



Direktorat Jenderal Pendidikan Tinggi, Riset, dan Teknologi
Kementerian Pendidikan, Kebudayaan, Riset, dan Teknologi
Republik Indonesia

Pelatihan **Microcredential** **CERTIFICATION** untuk **Associate Data Scientist**

1 November - 10 Desember 2021



Hands-On

Hands-On ini digunakan pada kegiatan Microcredential Associate Data Scientist 2021

Tugas Mandiri Pertemuan 14

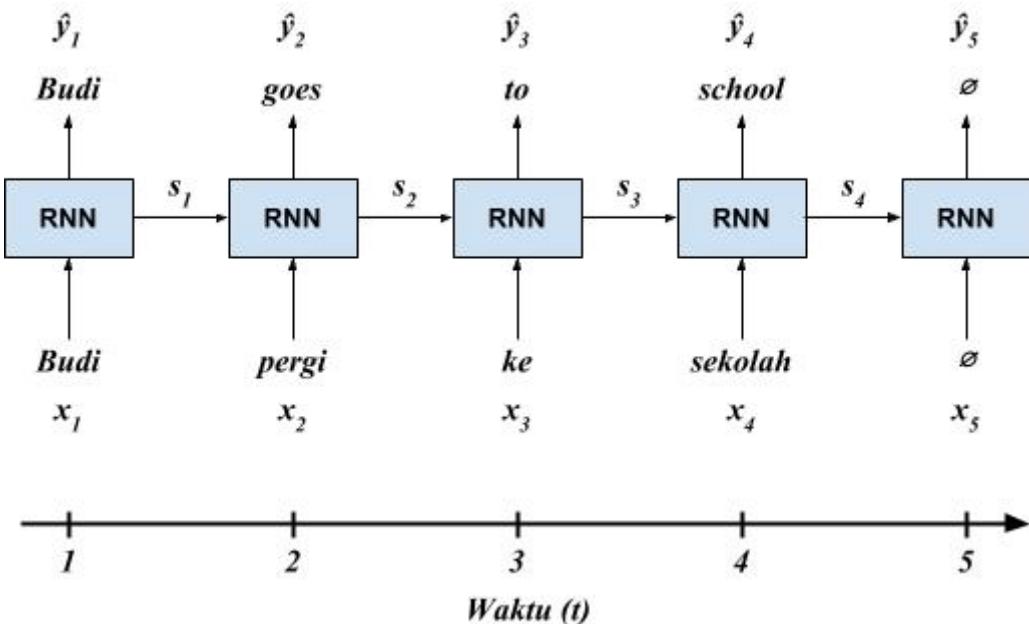
Pertemuan 14 (empatbelas) pada Microcredential Associate Data Scientist 2021 menyampaikan materi mengenai Membangun Model (RNN dan LSTM). silakan Anda kerjakan Latihan 1 s/d 5. Output yang anda lihat merupakan panduan yang dapat Anda ikuti dalam penulisan code :)

RNN

Jaringan saraf berulang atau recurrent neural network (RNN) adalah jenis arsitektur jaringan saraf tiruan yang pemrosesannya dipanggil berulang-ulang untuk memroses masukan yang biasanya adalah data sekuensial. RNN masuk dalam kategori deep learning karena data diproses melalui banyak lapis (layer). RNN telah mengalami kemajuan yang pesat dan telah merevolusi bidang-bidang seperti pemrosesan bahasa alami (NLP), pengenalan suara, sintesa musik, pemrosesan data finansial seri waktu, analisa deret DNA, analisa video, dan sebagainya.

RNN memroses input secara sekuensial, sampel per sampel. Dalam tiap pemrosesan, output yang dihasilkan tidak hanya merupakan fungsi dari sampel itu saja, tapi juga berdasarkan state internal yang merupakan hasil dari pemrosesan sampel-sampel sebelumnya (atau setelahnya, pada bidirectional RNN).

Berikut adalah ilustrasi bagaimana RNN bekerja. Misalnya kita membuat RNN untuk menerjemahkan bahasa Indonesia ke bahasa Inggris



Ilustrasi di atas kelihatan rumit, tapi sebenarnya cukup mudah dipahami.

- sumbu horizontal adalah waktu, direpresentasikan dengan simbol t. Dapat kita bayangkan pemrosesan berjalan dari kiri ke kanan. Selanjutnya kita sebut t adalah langkah waktu (time step).
- Keseluruhan input adalah kalimat, dalam hal ini:

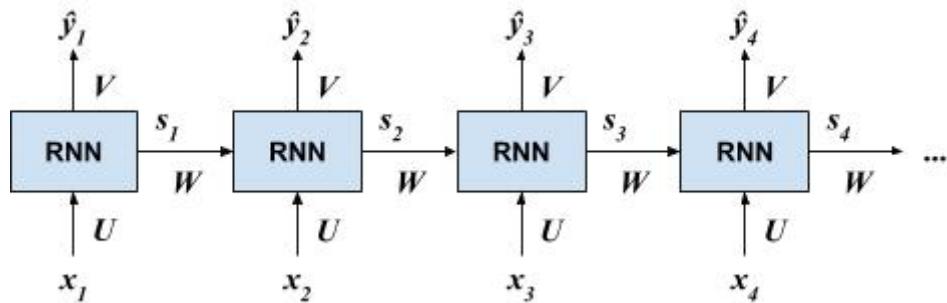
Budi pergi ke sekolah.

- Pemrosean input oleh RNN adalah kata demi kata. Input kata-kata ini disimbolkan dengan x_1, x_2, \dots, x_5 , atau secara umum x_t .
- Output adalah kalimat, dalam hal ini:

Budi goes to school.

- RNN memberikan output kata demi kata, dan ini kita simbolkan dengan $\hat{y}_1, \hat{y}_2, \dots, \hat{y}_5$, atau secara umum \hat{y}_t .
- Dalam tiap pemrosesan, RNN akan menyimpan state internal yaitu s_t , yang diberikan dari satu langkah waktu ke langkah waktu berikutnya. Inilah “memori” dari RNN.

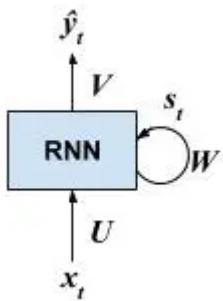
Dengan contoh di atas, kita bisa generalisasikan arsitektur RNN sebagai berikut:



Tambahan yang tidak terdapat di diagram sebelumnya adalah U, V, dan W, yang merupakan parameter-parameter yang dimiliki RNN. Kita akan bahas pemakaian parameter-parameter ini nanti.

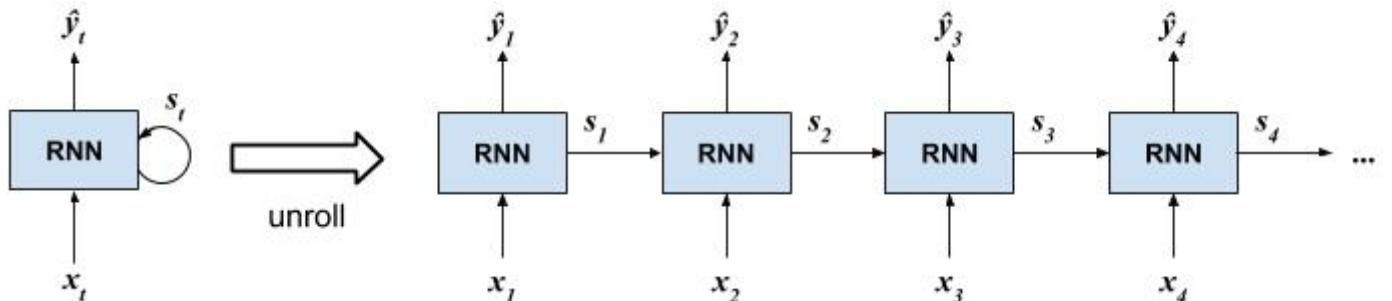
Penting untuk dipahami bahwa walaupun ada empat kotak RNN di gambar di atas, empat kotak itu mencerminkan satu modul RNN yang sama (satu instans model dengan parameter-parameter U, V, dan W yang sama). Penggambaran di atas hanya agar aspek sekuensialnya lebih tergambar.

Alternatif representasinya adalah seperti ini, agar lebih jelas bahwa hanya ada satu modul RNN:



Inilah sebabnya kenapa arsitektur ini disebut RNN. Kata recurrent (berulang) dalam RNN timbul karena RNN melakukan perhitungan yang sama secara berulang-ulang atas input yang kita berikan.

Sering juga kedua ilustrasi di atas digabungkan jadi satu sbb:



Sesuai dengan gambar di atas, ilustrasi di sebelah kanan adalah penjabaran (unrolled) dari versi berulang di sebelah kiri.

Latihan (1)

Melakukan import library yang dibutuhkan

In [51]:

```
# import Library pandas
import pandas as pd

# Import Library numpy
import numpy as np

# Import Library matplotlib untuk visualisasi
import matplotlib.pyplot as plt

# import Library for build model
from keras.layers import Dense,Dropout,SimpleRNN,LSTM
from keras.models import Sequential

# import Library untuk data preprocessing
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
```

Load Dataset

In [52]:

```
#Panggil file (load file bernama Stock.csv) dan simpan dalam dataframe
dataset ="Stock.csv"
data = pd.read_csv(dataset)
```

In [53]:

```
# tampilkan 5 baris data
data.head()
```

Out[53]:

	Date	Open	High	Low	Close	Volume	Name
0	2006-01-03	56.45	56.66	55.46	56.53	3716500	UTX
1	2006-01-04	56.80	56.80	55.84	56.19	3114500	UTX
2	2006-01-05	56.30	56.49	55.63	55.98	3118900	UTX
3	2006-01-06	56.45	56.67	56.10	56.16	2874300	UTX
4	2006-01-09	56.37	56.90	56.16	56.80	2467200	UTX

Review Data

In [54]:

```
# Melihat Informasi lebih detail mengenai struktur DataFrame dapat dilihat menggunakan fung
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3020 entries, 0 to 3019
Data columns (total 7 columns):
 #   Column  Non-Null Count  Dtype  
 ---  --   --   --   --   --   -- 
 0   Date    3020 non-null   object 
 1   Open    3019 non-null   float64
 2   High   3020 non-null   float64
 3   Low    3020 non-null   float64
 4   Close   3020 non-null   float64
 5   Volume  3020 non-null   int64  
 6   Name    3020 non-null   object 
dtypes: float64(4), int64(1), object(2)
memory usage: 165.3+ KB
```

In [55]:

```
# Kolom 'Low' yang akan kita gunakan dalam membangun model
# Slice kolom 'Low'
```

```
Low_data = data.iloc[:,3:4].values
```

In [56]:

```
# cek output Low_data
Low_data
```

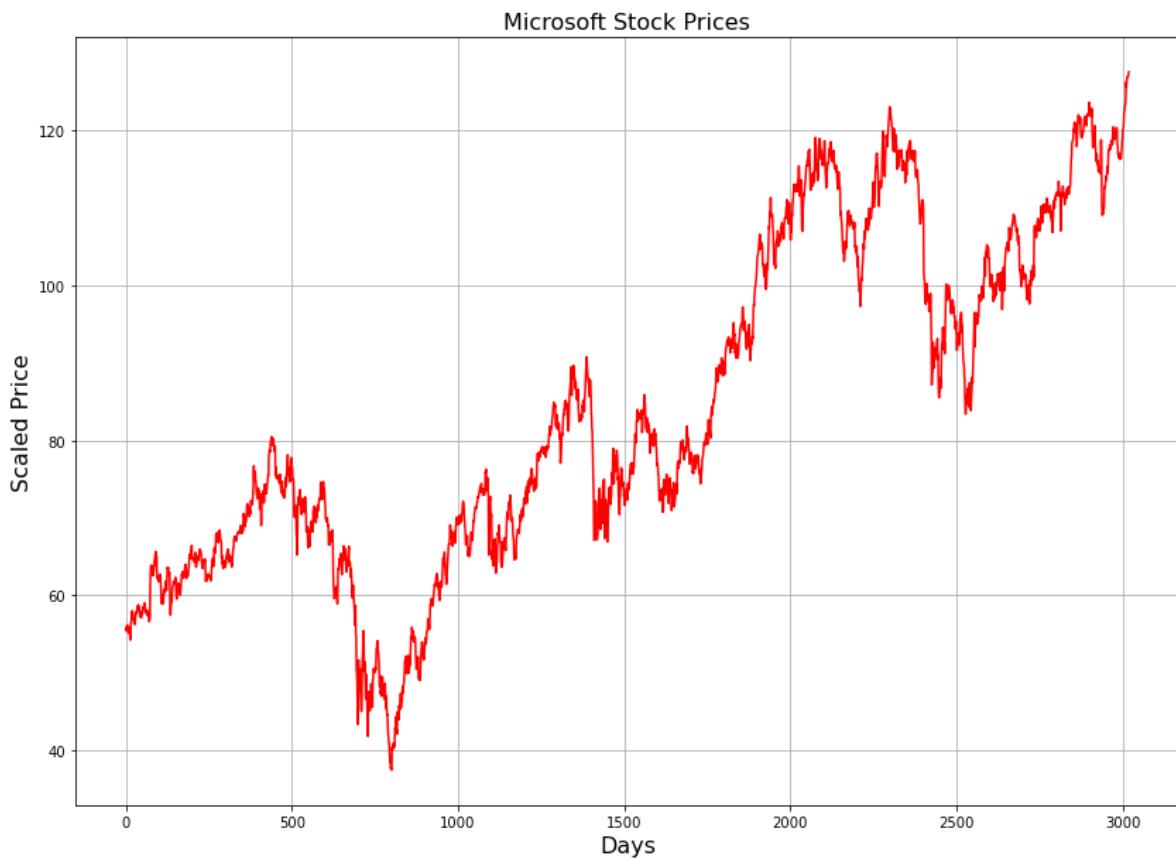
Out[56]:

```
array([[ 55.46],
       [ 55.84],
       [ 55.63],
       ...,
       [126.92],
       [127.29],
       [127.57]])
```

In [57]:

```
# Visualizing low_data
```

```
plt.figure(figsize=(14,10))
plt.plot(Low_data,c="red")
plt.title("Microsoft Stock Prices",fontsize=16)
plt.xlabel("Days",fontsize=16)
plt.ylabel("Scaled Price",fontsize=16)
plt.grid()
plt.show()
```



Latihan (2)

Data Preprocessing

In [58]:

```
# Menskalakan data antara 1 dan 0 (scaling) pada Low data

scaler = MinMaxScaler(feature_range=(0,1))
Low_scaled = scaler.fit_transform(Low_data)
```

In [59]:

```
# definisikan variabel step dan train

step_size = 21

train_x = []
train_y = []
```

In [60]:

```
# membuat fitur dan lists label

# for i in range(step_size,3019):
#     train_x.append(Low_scaled[i-step_size:i,0])
#     train_y.append(Low_scaled[i,0])

for i in range(step_size,3019):           # making feature and the label lists
    train_x.append(Low_scaled[i-step_size:i,0])
    train_y.append(Low_scaled[i,0])
```

In [61]:

```
# mengonversi list yang telah dibuat sebelumnya ke array

train_x = np.array(train_x)
train_y = np.array(train_y)
```

In [62]:

```
# cek dimensi data dengan function .shape

print(train_x.shape)
```

(2998, 21)

In [63]:

```
# 498 hari terakhir akan digunakan dalam pengujian
# 2500 hari pertama akan digunakan dalam pelatihan

test_x = train_x[2500:]
train_x = train_x[:2500]

test_y = train_y[2500:]
train_y = train_y[:2500]
```

In [64]:

```
# reshape data untuk dimasukkan kedalam Keras model  
  
train_x = np.reshape(train_x, (2500, step_size, 1))  
test_x = np.reshape(test_x, (498, step_size, 1))
```

In [66]:

```
# cek kembali dimensi data yang telah di reshape dengan function .shape  
  
print(train_x.shape)  
print(test_x.shape)  
  
(2500, 21, 1)  
(498, 21, 1)
```

Sekarang kita bisa mulai membuat model kita, dimulai dengan RNN

Latihan (3)

Build Model - RNN

In [67]:

```
# buat varibel penampung model RNN  
rnn_model = Sequential()
```

In [68]:

```
# Output dari SimpleRNN akan menjadi bentuk tensor 2D (batch_size, 40) dengan Dropout sebes  
rnn_model.add(SimpleRNN(40,activation="tanh",return_sequences=True, input_shape=(train_x.sh  
rnn_model.add(Dropout(0.15))  
  
rnn_model.add(SimpleRNN(40,activation="tanh",return_sequences=True))  
rnn_model.add(Dropout(0.15))  
  
rnn_model.add(SimpleRNN(40,activation="tanh",return_sequences=False))  
rnn_model.add(Dropout(0.15))  
  
# Add a Dense Layer with 1 units.  
rnn_model.add(Dense(1))
```

In [69]:

```
# menambahkan Loss function kedalam model RNN dengan tipe MSE  
  
rnn_model.compile(optimizer="adam",loss="MSE")
```

In [70]:

```
# fit the model RNN, dengan epoch 20 dan batch size 25
rnn_model.fit(train_x,train_y,epochs=20,batch_size=25)
```

```
Epoch 1/20
100/100 [=====] - 6s 27ms/step - loss: 0.2435
Epoch 2/20
100/100 [=====] - 2s 25ms/step - loss: 0.0694
Epoch 3/20
100/100 [=====] - 3s 27ms/step - loss: 0.0385
Epoch 4/20
100/100 [=====] - 3s 26ms/step - loss: 0.0224
Epoch 5/20
100/100 [=====] - 2s 21ms/step - loss: 0.0174
Epoch 6/20
100/100 [=====] - 2s 23ms/step - loss: 0.0135: 0s -
Epoch 7/20
100/100 [=====] - 2s 24ms/step - loss: 0.0115
Epoch 8/20
100/100 [=====] - 2s 22ms/step - loss: 0.0094
Epoch 9/20
100/100 [=====] - 2s 23ms/step - loss: 0.0081
Epoch 10/20
100/100 [=====] - 2s 23ms/step - loss: 0.0069
Epoch 11/20
100/100 [=====] - 2s 24ms/step - loss: 0.0063
Epoch 12/20
100/100 [=====] - 3s 27ms/step - loss: 0.0053
Epoch 13/20
100/100 [=====] - 3s 28ms/step - loss: 0.0051
Epoch 14/20
100/100 [=====] - 2s 25ms/step - loss: 0.0044
Epoch 15/20
100/100 [=====] - 2s 24ms/step - loss: 0.0041
Epoch 16/20
100/100 [=====] - 2s 22ms/step - loss: 0.0041
Epoch 17/20
100/100 [=====] - 2s 21ms/step - loss: 0.0037: - E
TA: 0s - loss: 0.00
Epoch 18/20
100/100 [=====] - 2s 23ms/step - loss: 0.0034
Epoch 19/20
100/100 [=====] - 2s 20ms/step - loss: 0.0028
Epoch 20/20
100/100 [=====] - 2s 21ms/step - loss: 0.0028
```

Out[70]:

```
<keras.callbacks.History at 0x2aca85342e0>
```

In [71]:

```
# Prediksi Model RNN
rnn_predictions = rnn_model.predict(test_x)

rnn_score = r2_score(test_y,rnn_predictions)
```

In [72]:

```
rnn_score
```

Out[72]:

```
0.8828160685945109
```

Latihan (4)

Build Model - LSTM

In [73]:

```
# buat varibel penampung model LSTM
lstm_model = Sequential()
```

In [74]:

```
# Add a LSTM Layer with 40 internal units. dengan Dropout sebesar 0.15
lstm_model.add(LSTM(40,activation="tanh",return_sequences=True, input_shape=(train_x.shape[0], train_x.shape[1]), dropout=0.15))

lstm_model.add(LSTM(40,activation="tanh",return_sequences=True))
lstm_model.add(Dropout(0.15))

lstm_model.add(LSTM(40,activation="tanh",return_sequences=False))
lstm_model.add(Dropout(0.15))

# Add a Dense layer with 1 units.
lstm_model.add(Dense(1))
```

In [75]:

```
# menambahkan Loss function kedalam model Lstm dengan tipe MSE
lstm_model.compile(optimizer="adam", loss="MSE")
```

In [76]:

```
# fit Lstm model, dengan epoch 20 dan batch size 25
```

```
lstm_model.fit(train_x,train_y,epochs=20,batch_size=25)
```

```
Epoch 1/20  
100/100 [=====] - 11s 41ms/step - loss: 0.0166  
Epoch 2/20  
100/100 [=====] - 4s 42ms/step - loss: 0.0036  
Epoch 3/20  
100/100 [=====] - 4s 39ms/step - loss: 0.0032  
Epoch 4/20  
100/100 [=====] - 4s 43ms/step - loss: 0.0032  
Epoch 5/20  
100/100 [=====] - 4s 41ms/step - loss: 0.0027  
Epoch 6/20  
100/100 [=====] - 4s 41ms/step - loss: 0.0026  
Epoch 7/20  
100/100 [=====] - 4s 36ms/step - loss: 0.0026  
Epoch 8/20  
100/100 [=====] - 4s 37ms/step - loss: 0.0025  
Epoch 9/20  
100/100 [=====] - 4s 38ms/step - loss: 0.0023  
Epoch 10/20  
100/100 [=====] - 4s 41ms/step - loss: 0.0021  
Epoch 11/20  
100/100 [=====] - 4s 38ms/step - loss: 0.0022  
Epoch 12/20  
100/100 [=====] - 4s 42ms/step - loss: 0.0022  
Epoch 13/20  
100/100 [=====] - 5s 46ms/step - loss: 0.0021  
Epoch 14/20  
100/100 [=====] - 4s 37ms/step - loss: 0.0019  
Epoch 15/20  
100/100 [=====] - 4s 37ms/step - loss: 0.0018  
Epoch 16/20  
100/100 [=====] - 4s 37ms/step - loss: 0.0018  
Epoch 17/20  
100/100 [=====] - 4s 37ms/step - loss: 0.0018  
Epoch 18/20  
100/100 [=====] - 4s 37ms/step - loss: 0.0018  
Epoch 19/20  
100/100 [=====] - 4s 40ms/step - loss: 0.0016  
Epoch 20/20  
100/100 [=====] - 4s 45ms/step - loss: 0.0016
```

Out[76]:

```
<keras.callbacks.History at 0x2acaad3df40>
```

In [77]:

```
# Prediksi Model LSTM
```

```
lstm_predictions = lstm_model.predict(test_x)
```

```
lstm_score = r2_score(test_y,lstm_predictions)
```

In [78]:

```
lstm_score
```

Out[78]:

```
0.8726014375678911
```

Latihan (5)

Evaluation

In [79]:

```
# Cetak nilai prediksi masing-masing model dengan menggunakan r^2 square
print("R^2 Score dari model RNN",rnn_score)
print("R^2 Score dari model LSTM",lstm_score)
```

```
R^2 Score dari model RNN 0.8828160685945109
```

```
R^2 Score dari model LSTM 0.8726014375678911
```

Visualisasi Perbandingan Hasil Model prediksi dengan data original

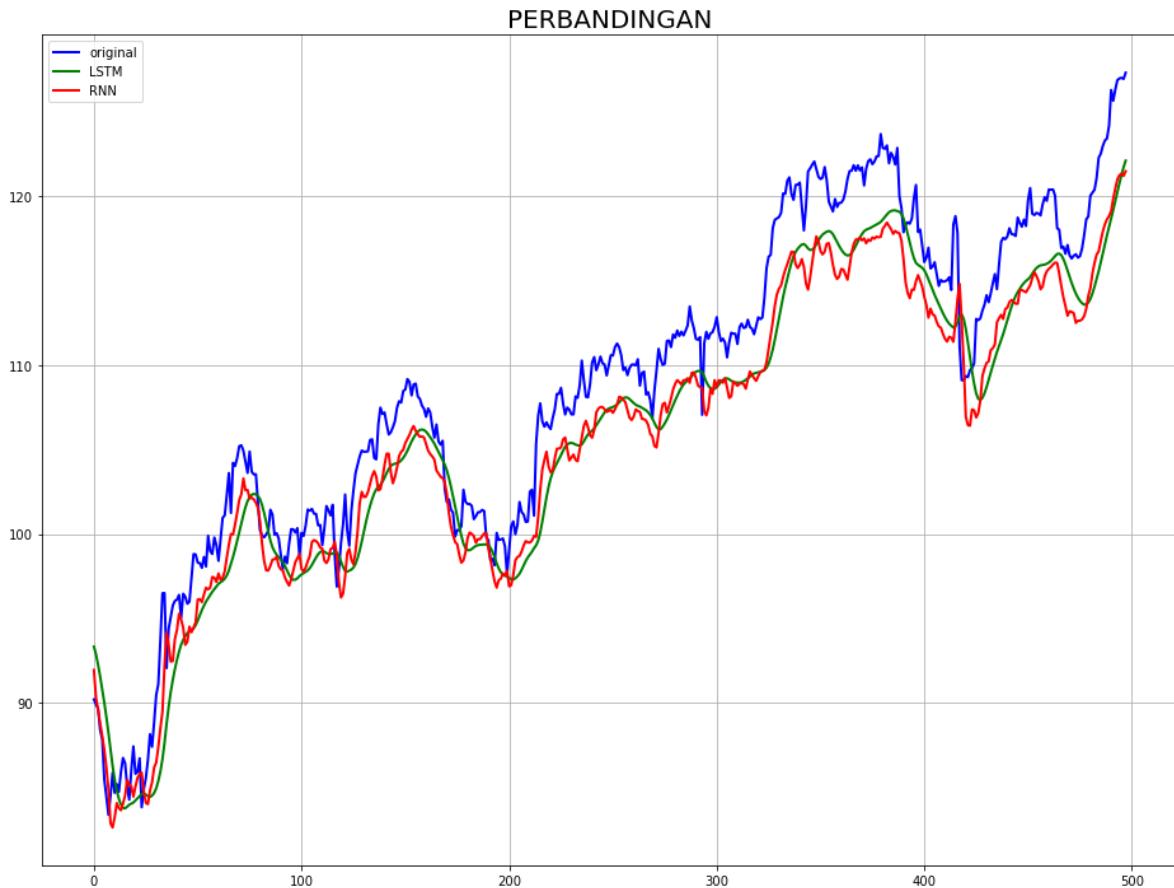
In [80]:

```
lstm_predictions = scaler.inverse_transform(lstm_predictions)
rnn_predictions = scaler.inverse_transform(rnn_predictions)
test_y = scaler.inverse_transform(test_y.reshape(-1,1))
```

In [81]:

```
plt.figure(figsize=(16,12))

plt.plot(test_y, c="blue", linewidth=2, label="original")
plt.plot(lstm_predictions, c="green", linewidth=2, label="LSTM")
plt.plot(rnn_predictions, c="red", linewidth=2, label="RNN")
plt.legend()
plt.title("PERBANDINGAN", fontsize=20)
plt.grid()
plt.show()
```



Berikan Kesimpulan Anda!

Dengan menggunakan model LTSM dan RNN seperti ini dapat membantu untuk membuat model prediksi stock yang hasilnya mirip dengan data original, dan membuat saya merasa bisa bekerja di Wall Street :D