**PREDIKSI HARGA SAHAM BBCA MENGGUNAKAN METODE RNN DAN CNN**

Farhan Rahardian

A11.2022.14096

A11.4413

1. Judul :Prediksi Harga Saham BBCA dengan Menggunakan Metode RNN dan CNN

Identitas : Farhan Rahardian / A11.2022.14096 / A11.4413

1. Ringkasan dan Permasalahan Projek

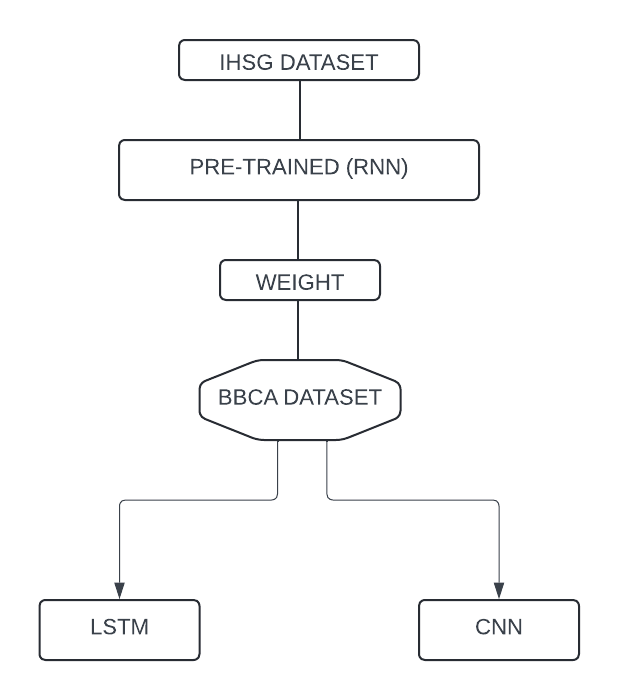
Saham adalah suatu instrumen keuangan yang mewakili kepemilikan perusahaan oleh seseorang individu. Setiap pemilik saham berhak mendapatkan sebagian keuntungan dari perusahaan. Akan tetapi perusahaan tidak selalu mendapatkan keuntungan, saat hal itu terjadi harga saham perusahaan tersebut akan turun. Projek ini bertujuan untuk membandingkan metode RNN dan CNN. Dataset yang digunakan pada pembuatan projek ini adalah track record harga saham BBCA dan IHSG dari tahun 2004 hingga tahun 2024. Data ini didapatkan dari website Kaggle.

Permasalahan : Bagaimana performa RNN dan CNN dalam memprediksi saham.

Tujuan

1. Memprediksi harga saham
2. Analisis saham BBCA
3. Mempermudah pemilik saham yang awam

Model / Alur



1. Penjelasan Dataset, EDA, dan Features data

Dataset yang saya gunakan dari Kaggle dengan nama dataset ihsg.csv .Dataset ini berisi informasi

Date : Terjadinya transaksi pada saham

Open : Harga pembukaan saham pada awal perdagangan pada hari tersebut

High : Harga tertinggi yang dicapai oleh saham pada hari tersebut

Low : Harga terendah yang dicapai oleh saham pada hari tersebut

Close : Harga penutupan saham pada akhir perdagangan

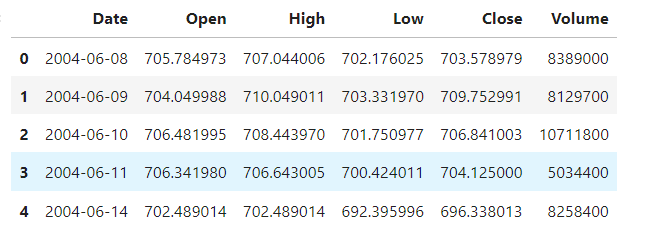
Volume : Jumlah saham yang diperdagangkan

**EDA**

Memuat dataset dan menampilkan informasi umum

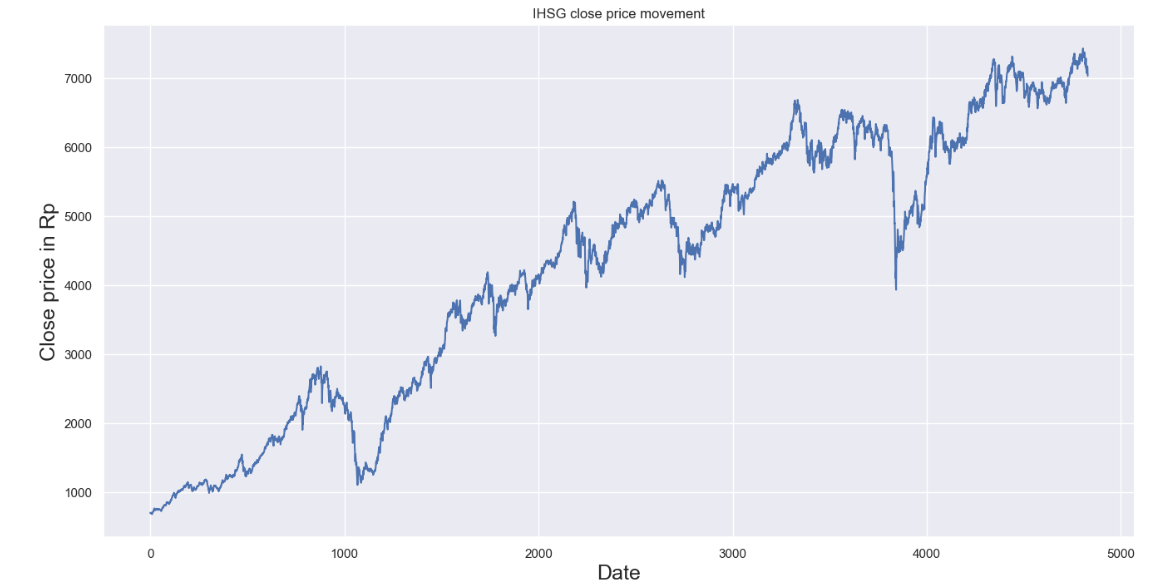
**import** pandas **as** pd

Memasukkan data



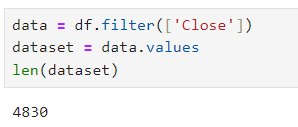
*Plot Figure*

Digunakan untuk menampilkan data harga saat pasar saham tutup setiap hari



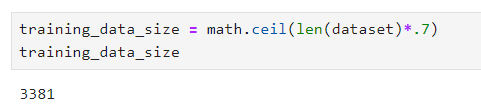
*Menghitung jumlah data*

Pada bagian ini, diambil data dari kolom ‘close’ dari DataFrame df, mengonversinya ke dalam bentuk array NumPy ,kemudian menghitung jumlah entri dalam array tersebut.



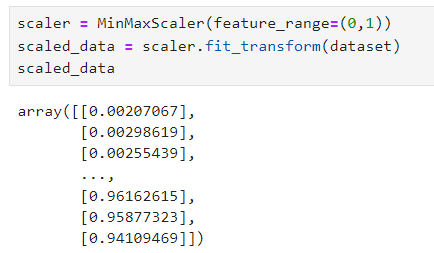
*Menghitung ukuran data*

Dibagian ini digunakan untuk menghitung ukuran data pelatihan (training data size) yang akan digunakan, dengan menggunakan 70% dari total data yang ada dalam dataset.



*Mengatur rentan*

Menyesuaikan data yang akan digunakan dengan rentang yang sudah diatur



*Mempersiapkan Data*

Bagian ini akan mempersiapkan data untuk model time series dengan menggunakan pendekatan sliding window, di mana setiap data latihan (x\_train) terdiri dari jendela 50 nilai sebelumnya dan targetnya (y\_train) adalah nilai berikutnya setelah jendela tersebut. Metode ini umum digunakan untuk mempersiapkan data time series untuk pemodelan dan prediksi berikutnya.

train\_data = scaled\_data[0:training\_data\_size,:]

x\_train = []

y\_train = []

for i in range(50, len(train\_data)):

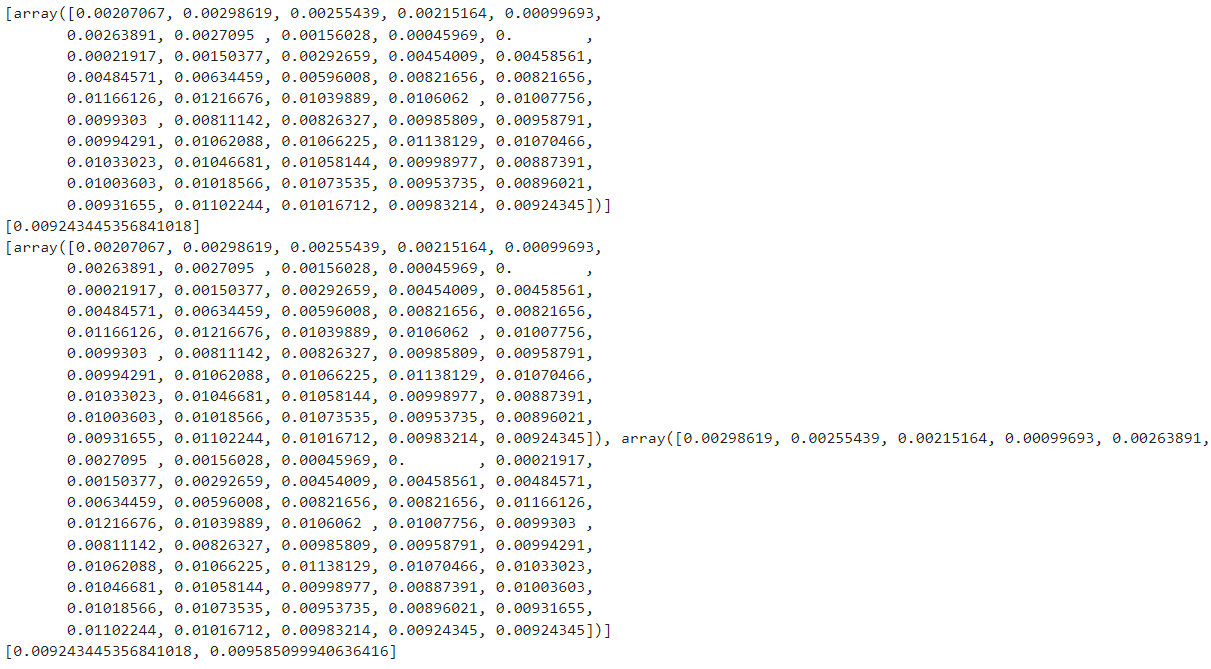
x\_train.append(train\_data[i-50:i,0])

y\_train.append(train\_data[i,0])

if i<=51:

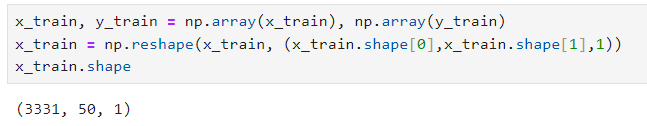
print(x\_train)

print(y\_train)



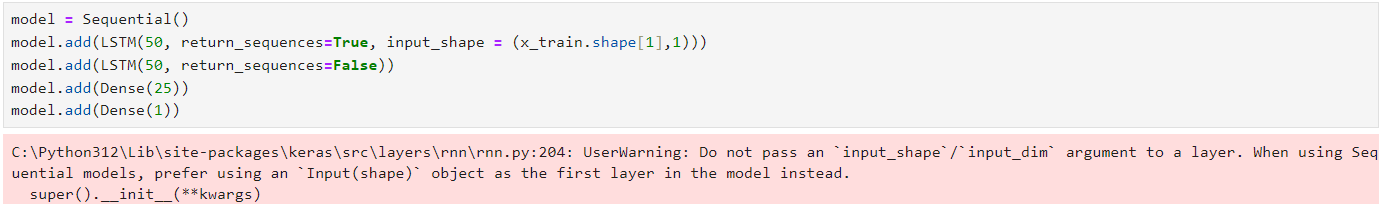
*Konversi array*

Melakukan konversi ke array NumPy dan mengubah bentuk x\_train menggunakan np.reshape() sehingga dapat dapat digunakan untuk di-train ke model



1. Proses Learning / Modeling

Pada bagian ini, akan dibuat model yang akan dilatih. Model ini memiliki dua layer LSTM untuk memproses sequence data input, diikuti oleh dua layer dense yang lebih sederhana untuk melakukan pemrosesan lebih lanjut dan menghasilkan output.



*Meng-compile model*

model.compile(optimizer='adam', loss='mean\_squared\_error')

*Men-train model*

model.fit(x\_train, y\_train, batch\_size=1, epochs=50)

Epoch 1/50

**3331/3331** ━━━━━━━━━━━━━━━━━━━━ **48s** 14ms/step - loss: 0.0034

Epoch 2/50

**3331/3331** ━━━━━━━━━━━━━━━━━━━━ **49s** 15ms/step - loss: 4.4627e-04

Epoch 3/50

**3331/3331** ━━━━━━━━━━━━━━━━━━━━ **47s** 14ms/step - loss: 2.1465e-04

Epoch 4/50

**3331/3331** ━━━━━━━━━━━━━━━━━━━━ **46s** 14ms/step - loss: 1.6831e-04

Epoch 5/50

**3331/3331** ━━━━━━━━━━━━━━━━━━━━ **47s** 14ms/step - loss: 1.8506e-04

Epoch 6/50

**3331/3331** ━━━━━━━━━━━━━━━━━━━━ **47s** 14ms/step - loss: 1.9013e-04

Epoch 7/50

**3331/3331** ━━━━━━━━━━━━━━━━━━━━ **47s** 14ms/step - loss: 1.1870e-04

Epoch 8/50

**3331/3331** ━━━━━━━━━━━━━━━━━━━━ **47s** 14ms/step - loss: 9.8173e-05

Epoch 9/50

**3331/3331** ━━━━━━━━━━━━━━━━━━━━ **47s** 14ms/step - loss: 9.9343e-05

Epoch 10/50

**3331/3331** ━━━━━━━━━━━━━━━━━━━━ **48s** 14ms/step - loss: 1.0837e-04

Epoch 11/50

**3331/3331** ━━━━━━━━━━━━━━━━━━━━ **45s** 13ms/step - loss: 9.3660e-05

Epoch 12/50

**3331/3331** ━━━━━━━━━━━━━━━━━━━━ **44s** 13ms/step - loss: 8.7212e-05

Epoch 13/50

**3331/3331** ━━━━━━━━━━━━━━━━━━━━ **45s** 14ms/step - loss: 9.3704e-05

Epoch 14/50

**3331/3331** ━━━━━━━━━━━━━━━━━━━━ **45s** 13ms/step - loss: 8.5397e-05

Epoch 15/50

**3331/3331** ━━━━━━━━━━━━━━━━━━━━ **44s** 13ms/step - loss: 9.6708e-05

Epoch 16/50

**3331/3331** ━━━━━━━━━━━━━━━━━━━━ **45s** 14ms/step - loss: 7.8246e-05

Epoch 17/50

**3331/3331** ━━━━━━━━━━━━━━━━━━━━ **47s** 14ms/step - loss: 9.5221e-05

Epoch 18/50

**3331/3331** ━━━━━━━━━━━━━━━━━━━━ **48s** 14ms/step - loss: 7.9358e-05

Epoch 19/50

**3331/3331** ━━━━━━━━━━━━━━━━━━━━ **46s** 14ms/step - loss: 7.6036e-05

Epoch 20/50

**3331/3331** ━━━━━━━━━━━━━━━━━━━━ **47s** 14ms/step - loss: 7.7395e-05

Epoch 21/50

**3331/3331** ━━━━━━━━━━━━━━━━━━━━ **47s** 14ms/step - loss: 8.2361e-05

Epoch 22/50

**3331/3331** ━━━━━━━━━━━━━━━━━━━━ **47s** 14ms/step - loss: 8.1662e-05

Epoch 23/50

**3331/3331** ━━━━━━━━━━━━━━━━━━━━ **46s** 14ms/step - loss: 9.2808e-05

Epoch 24/50

**3331/3331** ━━━━━━━━━━━━━━━━━━━━ **45s** 14ms/step - loss: 8.4159e-05

Epoch 25/50

**3331/3331** ━━━━━━━━━━━━━━━━━━━━ **44s** 13ms/step - loss: 8.3472e-05

Epoch 26/50

**3331/3331** ━━━━━━━━━━━━━━━━━━━━ **46s** 14ms/step - loss: 7.7394e-05

Epoch 27/50

**3331/3331** ━━━━━━━━━━━━━━━━━━━━ **46s** 14ms/step - loss: 8.1450e-05

Epoch 28/50

**3331/3331** ━━━━━━━━━━━━━━━━━━━━ **46s** 14ms/step - loss: 7.0173e-05

Epoch 29/50

**3331/3331** ━━━━━━━━━━━━━━━━━━━━ **46s** 14ms/step - loss: 6.7685e-05

Epoch 30/50

**3331/3331** ━━━━━━━━━━━━━━━━━━━━ **46s** 14ms/step - loss: 8.1875e-05

Epoch 31/50

**3331/3331** ━━━━━━━━━━━━━━━━━━━━ **49s** 15ms/step - loss: 6.4470e-05

Epoch 32/50

**3331/3331** ━━━━━━━━━━━━━━━━━━━━ **54s** 16ms/step - loss: 6.2396e-05

Epoch 33/50

**3331/3331** ━━━━━━━━━━━━━━━━━━━━ **54s** 16ms/step - loss: 6.3167e-05

Epoch 34/50

**3331/3331** ━━━━━━━━━━━━━━━━━━━━ **54s** 16ms/step - loss: 7.1608e-05

Epoch 35/50

**3331/3331** ━━━━━━━━━━━━━━━━━━━━ **53s** 16ms/step - loss: 6.8196e-05

Epoch 36/50

**3331/3331** ━━━━━━━━━━━━━━━━━━━━ **53s** 16ms/step - loss: 6.8202e-05

Epoch 37/50

**3331/3331** ━━━━━━━━━━━━━━━━━━━━ **54s** 16ms/step - loss: 7.7265e-05

Epoch 38/50

**3331/3331** ━━━━━━━━━━━━━━━━━━━━ **54s** 16ms/step - loss: 6.8223e-05

Epoch 39/50

**3331/3331** ━━━━━━━━━━━━━━━━━━━━ **54s** 16ms/step - loss: 8.1290e-05

Epoch 40/50

**3331/3331** ━━━━━━━━━━━━━━━━━━━━ **53s** 16ms/step - loss: 6.7125e-05

Epoch 41/50

**3331/3331** ━━━━━━━━━━━━━━━━━━━━ **54s** 16ms/step - loss: 7.0352e-05

Epoch 42/50

**3331/3331** ━━━━━━━━━━━━━━━━━━━━ **54s** 16ms/step - loss: 6.6585e-05

Epoch 43/50

**3331/3331** ━━━━━━━━━━━━━━━━━━━━ **54s** 16ms/step - loss: 6.7135e-05

Epoch 44/50

**3331/3331** ━━━━━━━━━━━━━━━━━━━━ **54s** 16ms/step - loss: 7.1876e-05

Epoch 45/50

**3331/3331** ━━━━━━━━━━━━━━━━━━━━ **54s** 16ms/step - loss: 6.9559e-05

Epoch 46/50

**3331/3331** ━━━━━━━━━━━━━━━━━━━━ **55s** 16ms/step - loss: 5.8630e-05

Epoch 47/50

**3331/3331** ━━━━━━━━━━━━━━━━━━━━ **54s** 16ms/step - loss: 5.9728e-05

Epoch 48/50

**3331/3331** ━━━━━━━━━━━━━━━━━━━━ **54s** 16ms/step - loss: 6.4148e-05

Epoch 49/50

**3331/3331** ━━━━━━━━━━━━━━━━━━━━ **55s** 16ms/step - loss: 6.3174e-05

Epoch 50/50

**3331/3331** ━━━━━━━━━━━━━━━━━━━━ **55s** 16ms/step - loss: 5.9374e-05

[11]:

<keras.src.callbacks.history.History at 0x2284d691b80>

*Mempersiapkan data*

test\_data = scaled\_data[training\_data\_size-50:, :]

x\_test = []

y\_test = dataset[training\_data\_size:, :]

for i in range(50, len(test\_data)):

x\_test.append(test\_data[i-50:i, 0])

Kode ini mengubah x\_test dari list Python menjadi array NumPy dan mengubah bentuknya sehingga cocok dengan input yang diharapkan oleh model LSTM, yaitu (jumlah sampel, panjang sampel, 1), di mana setiap sampel adalah jendela 50 nilai time series sebelumnya.

x\_test = np.array(x\_test)

x\_test = np.reshape(x\_test, (x\_test.shape[0], x\_test.shape[1],1))

Kode ini akan menggunakan model jaringan saraf (model) untuk melakukan prediksi kepada data uji (x\_test). Setelah mendapatkan prediksi, hasilnya akan diubah kembali ke skala semula dengan menggunakan metode inverse\_transform dari objek scaler, yang sebelumnya digunakan untuk normalisasi data, sehingga predictions berisi nilai prediksi dalam unit yang dapat dibandingkan dengan data asli.

predictions = model.predict(x\_test)

predictions = scaler.inverse\_transform(predictions)

**46/46** ━━━━━━━━━━━━━━━━━━━━ **2s** 28ms/step

Menghitung RMSE

Kode ini akan menghitung Root Mean Squared Error (RMSE) antara predictions dan y\_test dengan menghitung selisih kuadrat antara keduanya,

mengambil rata-rata dari selisih tersebut, dan menghitung akar kuadrat dari nilai rata-rata tersebut.

rmse = np.sqrt(np.mean(predictions - y\_test)\*\*2)

Pe\_train Kode ini akan membagi data menjadi data pelatihan (train) dan data validasi (valid), yang nantinya akan digunakan untuk memvisualisasikan perbandingan antara harga penutupan yang sebenarnya (Close) dengan harga sesuai dengan prediksi model (predictions) menggunakan grafik matplotlib. Ini membantu dalam memahami performa model LSTM dalam memprediksi harga saham berdasarkan data historis yang telah dilatih.

train = data[:training\_data\_size]

valid = data[training\_data\_size:]

valid['predictions'] = predictions

plt.figure(figsize=(16,8))

plt.title('Model LMST', fontsize=18)

plt.xlabel('Date', fontsize=12)

plt.ylabel('Close price in Rp', fontsize=12)

plt.plot(train['Close'])

plt.plot(valid[['Close', 'predictions']])

plt.legend(['Train', 'Val', 'predictions'], loc='lower right')

plt.show()



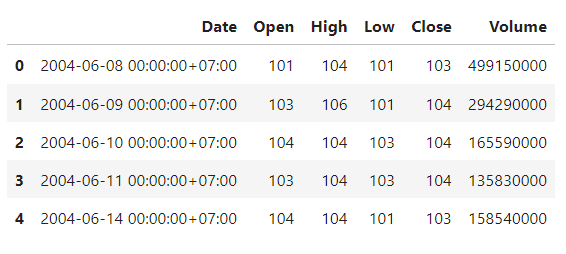
*Menyimpan total weight*

model.save\_weights('lstm\_model\_ihsg2\_weights.weights.h5')

*After pretrain*

Kode ini akan mengambil data harga penutupan saham BBCA, melakukan penskalaan data, dan mempersiapkan data yang akan di-train menggunakan model LSTM sebelumnya. Setelah melakukan fine-tuning model pada data baru, kode mengevaluasi performa model terhadap data uji, memprediksi harga penutupan, dan memvisualisasikan perbandingan antara data pelatihan, validasi, dan prediksi menggunakan grafik matplotlib.

*Data preprocessing*



plt.figure(figsize=(16,8))

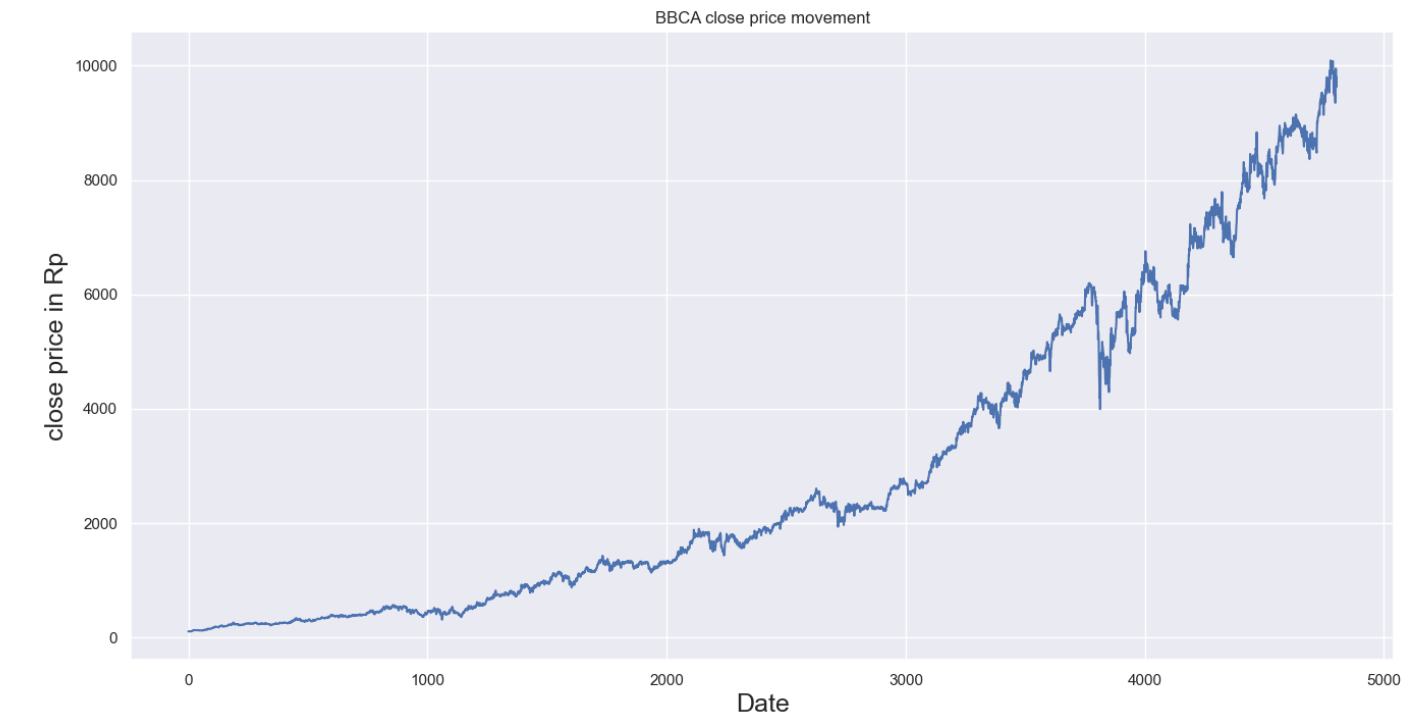
plt.title('BBCA close price movement')

plt.plot(df['Close'])

plt.xlabel('Date', fontsize=18)

plt.ylabel('close price in Rp', fontsize=18)

plt.show()





**Train Model**

***LSTM***

**2650/2650** ━━━━━━━━━━━━━━━━━━━━ **51s** 18ms/step - loss: 6.6012e-04 - val\_loss: 3.5068e-05

Epoch 2/10

**2650/2650** ━━━━━━━━━━━━━━━━━━━━ **45s** 17ms/step - loss: 2.5604e-04 - val\_loss: 1.2588e-04

Epoch 3/10

**2650/2650** ━━━━━━━━━━━━━━━━━━━━ **44s** 17ms/step - loss: 2.5715e-04 - val\_loss: 1.0427e-04

Epoch 4/10

**2650/2650** ━━━━━━━━━━━━━━━━━━━━ **44s** 17ms/step - loss: 1.8916e-04 - val\_loss: 3.8834e-05

Epoch 5/10

**2650/2650** ━━━━━━━━━━━━━━━━━━━━ **44s** 16ms/step - loss: 1.9645e-04 - val\_loss: 1.2851e-05

Epoch 6/10

**2650/2650** ━━━━━━━━━━━━━━━━━━━━ **44s** 16ms/step - loss: 1.3893e-04 - val\_loss: 8.8476e-05

Epoch 7/10

**2650/2650** ━━━━━━━━━━━━━━━━━━━━ **44s** 17ms/step - loss: 1.3939e-04 - val\_loss: 2.3461e-05

Epoch 8/10

**2650/2650** ━━━━━━━━━━━━━━━━━━━━ **44s** 17ms/step - loss: 1.4200e-04 - val\_loss: 8.6902e-05

Epoch 9/10

**2650/2650** ━━━━━━━━━━━━━━━━━━━━ **44s** 17ms/step - loss: 1.4129e-04 - val\_loss: 4.1891e-05

Epoch 10/10

**2650/2650** ━━━━━━━━━━━━━━━━━━━━ **46s** 17ms/step - loss: 1.5207e-04 - val\_loss: 1.3948e-04

Training fold 2...

Epoch 1/10

**2650/2650** ━━━━━━━━━━━━━━━━━━━━ **48s** 17ms/step - loss: 7.7980e-04 - val\_loss: 5.1657e-05

Epoch 2/10

**2650/2650** ━━━━━━━━━━━━━━━━━━━━ **44s** 17ms/step - loss: 2.3970e-04 - val\_loss: 1.2978e-05

Epoch 3/10

**2650/2650** ━━━━━━━━━━━━━━━━━━━━ **44s** 17ms/step - loss: 2.1525e-04 - val\_loss: 4.2271e-05

Epoch 4/10

**2650/2650** ━━━━━━━━━━━━━━━━━━━━ **45s** 17ms/step - loss: 1.8345e-04 - val\_loss: 1.9449e-04

Epoch 5/10

**2650/2650** ━━━━━━━━━━━━━━━━━━━━ **45s** 17ms/step - loss: 1.4515e-04 - val\_loss: 3.6253e-05

Epoch 6/10

**2650/2650** ━━━━━━━━━━━━━━━━━━━━ **45s** 17ms/step - loss: 1.6341e-04 - val\_loss: 5.8424e-05

Epoch 7/10

**2650/2650** ━━━━━━━━━━━━━━━━━━━━ **45s** 17ms/step - loss: 1.5446e-04 - val\_loss: 4.8283e-05

Epoch 8/10

**2650/2650** ━━━━━━━━━━━━━━━━━━━━ **44s** 17ms/step - loss: 1.6491e-04 - val\_loss: 8.3518e-05

Epoch 9/10

**2650/2650** ━━━━━━━━━━━━━━━━━━━━ **44s** 17ms/step - loss: 1.5225e-04 - val\_loss: 2.0394e-05

Epoch 10/10

**2650/2650** ━━━━━━━━━━━━━━━━━━━━ **45s** 17ms/step - loss: 1.3755e-04 - val\_loss: 8.7531e-05

Training fold 3...

Epoch 1/10

**2650/2650** ━━━━━━━━━━━━━━━━━━━━ **48s** 17ms/step - loss: 7.0179e-04 - val\_loss: 1.1121e-04

Epoch 2/10

**2650/2650** ━━━━━━━━━━━━━━━━━━━━ **44s** 17ms/step - loss: 2.4032e-04 - val\_loss: 1.8374e-04

Epoch 3/10

**2650/2650** ━━━━━━━━━━━━━━━━━━━━ **43s** 16ms/step - loss: 2.2170e-04 - val\_loss: 7.9055e-05

Epoch 4/10

**2650/2650** ━━━━━━━━━━━━━━━━━━━━ **43s** 16ms/step - loss: 1.8028e-04 - val\_loss: 1.8454e-05

Epoch 5/10

**2650/2650** ━━━━━━━━━━━━━━━━━━━━ **44s** 16ms/step - loss: 1.7246e-04 - val\_loss: 3.2605e-05

Epoch 6/10

**2650/2650** ━━━━━━━━━━━━━━━━━━━━ **44s** 16ms/step - loss: 1.6725e-04 - val\_loss: 4.2623e-05

Epoch 7/10

**2650/2650** ━━━━━━━━━━━━━━━━━━━━ **44s** 17ms/step - loss: 1.6007e-04 - val\_loss: 3.8995e-05

Epoch 8/10

**2650/2650** ━━━━━━━━━━━━━━━━━━━━ **44s** 16ms/step - loss: 1.2602e-04 - val\_loss: 2.0475e-04

Epoch 9/10

**2650/2650** ━━━━━━━━━━━━━━━━━━━━ **44s** 17ms/step - loss: 1.2694e-04 - val\_loss: 4.0334e-05

Epoch 10/10

**2650/2650** ━━━━━━━━━━━━━━━━━━━━ **44s** 16ms/step - loss: 1.2237e-04 - val\_loss: 6.6421e-05

Training fold 4...

Epoch 1/10

**2651/2651** ━━━━━━━━━━━━━━━━━━━━ **48s** 17ms/step - loss: 7.8545e-04 - val\_loss: 2.4937e-05

Epoch 2/10

**2651/2651** ━━━━━━━━━━━━━━━━━━━━ **45s** 17ms/step - loss: 3.1813e-04 - val\_loss: 1.6212e-04

Epoch 3/10

**2651/2651** ━━━━━━━━━━━━━━━━━━━━ **44s** 17ms/step - loss: 2.3522e-04 - val\_loss: 6.5272e-05

Epoch 4/10

**2651/2651** ━━━━━━━━━━━━━━━━━━━━ **46s** 17ms/step - loss: 1.8639e-04 - val\_loss: 4.7528e-05

Epoch 5/10

**2651/2651** ━━━━━━━━━━━━━━━━━━━━ **45s** 17ms/step - loss: 1.5757e-04 - val\_loss: 8.7968e-05

Epoch 6/10

**2651/2651** ━━━━━━━━━━━━━━━━━━━━ **46s** 17ms/step - loss: 1.9094e-04 - val\_loss: 1.7181e-05

Epoch 7/10

**2651/2651** ━━━━━━━━━━━━━━━━━━━━ **45s** 17ms/step - loss: 1.4647e-04 - val\_loss: 1.5481e-05

Epoch 8/10

**2651/2651** ━━━━━━━━━━━━━━━━━━━━ **45s** 17ms/step - loss: 1.6152e-04 - val\_loss: 2.4502e-05

Epoch 9/10

**2651/2651** ━━━━━━━━━━━━━━━━━━━━ **45s** 17ms/step - loss: 1.2210e-04 - val\_loss: 3.1814e-05

Epoch 10/10

**2651/2651** ━━━━━━━━━━━━━━━━━━━━ **45s** 17ms/step - loss: 1.2681e-04 - val\_loss: 1.0570e-04

Training fold 5...

Epoch 1/10

**2651/2651** ━━━━━━━━━━━━━━━━━━━━ **48s** 17ms/step - loss: 7.0546e-04 - val\_loss: 1.9155e-04

Epoch 2/10

**2651/2651** ━━━━━━━━━━━━━━━━━━━━ **45s** 17ms/step - loss: 2.7010e-04 - val\_loss: 1.7964e-04

Epoch 3/10

**2651/2651** ━━━━━━━━━━━━━━━━━━━━ **45s** 17ms/step - loss: 2.4603e-04 - val\_loss: 1.9514e-04

Epoch 4/10

**2651/2651** ━━━━━━━━━━━━━━━━━━━━ **45s** 17ms/step - loss: 1.8984e-04 - val\_loss: 3.8617e-05

Epoch 5/10

**2651/2651** ━━━━━━━━━━━━━━━━━━━━ **45s** 17ms/step - loss: 1.8680e-04 - val\_loss: 2.3826e-05

Epoch 6/10

**2651/2651** ━━━━━━━━━━━━━━━━━━━━ **45s** 17ms/step - loss: 1.7547e-04 - val\_loss: 3.4556e-05

Epoch 7/10

**2651/2651** ━━━━━━━━━━━━━━━━━━━━ **45s** 17ms/step - loss: 1.4052e-04 - val\_loss: 6.7548e-04

Epoch 8/10

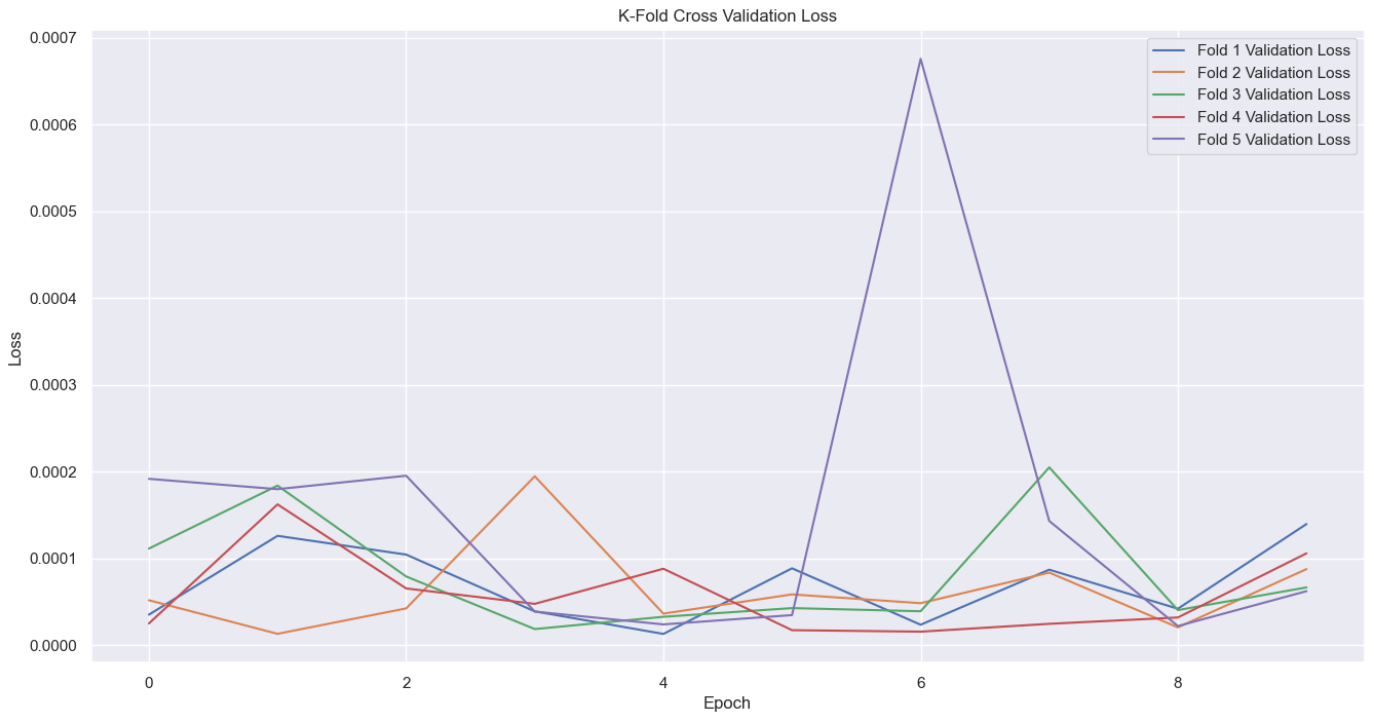
**2651/2651** ━━━━━━━━━━━━━━━━━━━━ **45s** 17ms/step - loss: 1.6854e-04 - val\_loss: 1.4299e-04

Epoch 9/10

**2651/2651** ━━━━━━━━━━━━━━━━━━━━ **45s** 17ms/step - loss: 1.3906e-04 - val\_loss: 2.1959e-05

Epoch 10/10

**2651/2651** ━━━━━━━━━━━━━━━━━━━━ **45s** 17ms/step - loss: 1.4019e-04 - val\_loss: 6.2120e-05



Epoch 1/10

**2650/2650** ━━━━━━━━━━━━━━━━━━━━ **47s** 17ms/step - loss: 3.8388e-04 - val\_loss: 1.1369e-04

Epoch 2/10

**2650/2650** ━━━━━━━━━━━━━━━━━━━━ **45s** 17ms/step - loss: 1.5558e-04 - val\_loss: 2.7523e-04

Epoch 3/10

**2650/2650** ━━━━━━━━━━━━━━━━━━━━ **44s** 17ms/step - loss: 7.9760e-05 - val\_loss: 1.6092e-04

Epoch 4/10

**2650/2650** ━━━━━━━━━━━━━━━━━━━━ **45s** 17ms/step - loss: 8.8141e-05 - val\_loss: 2.3294e-05

Epoch 5/10

**2650/2650** ━━━━━━━━━━━━━━━━━━━━ **44s** 17ms/step - loss: 7.4475e-05 - val\_loss: 2.2713e-05

Epoch 6/10

**2650/2650** ━━━━━━━━━━━━━━━━━━━━ **45s** 17ms/step - loss: 6.9998e-05 - val\_loss: 1.6780e-04

Epoch 7/10

**2650/2650** ━━━━━━━━━━━━━━━━━━━━ **44s** 17ms/step - loss: 6.2967e-05 - val\_loss: 1.1852e-04

Epoch 8/10

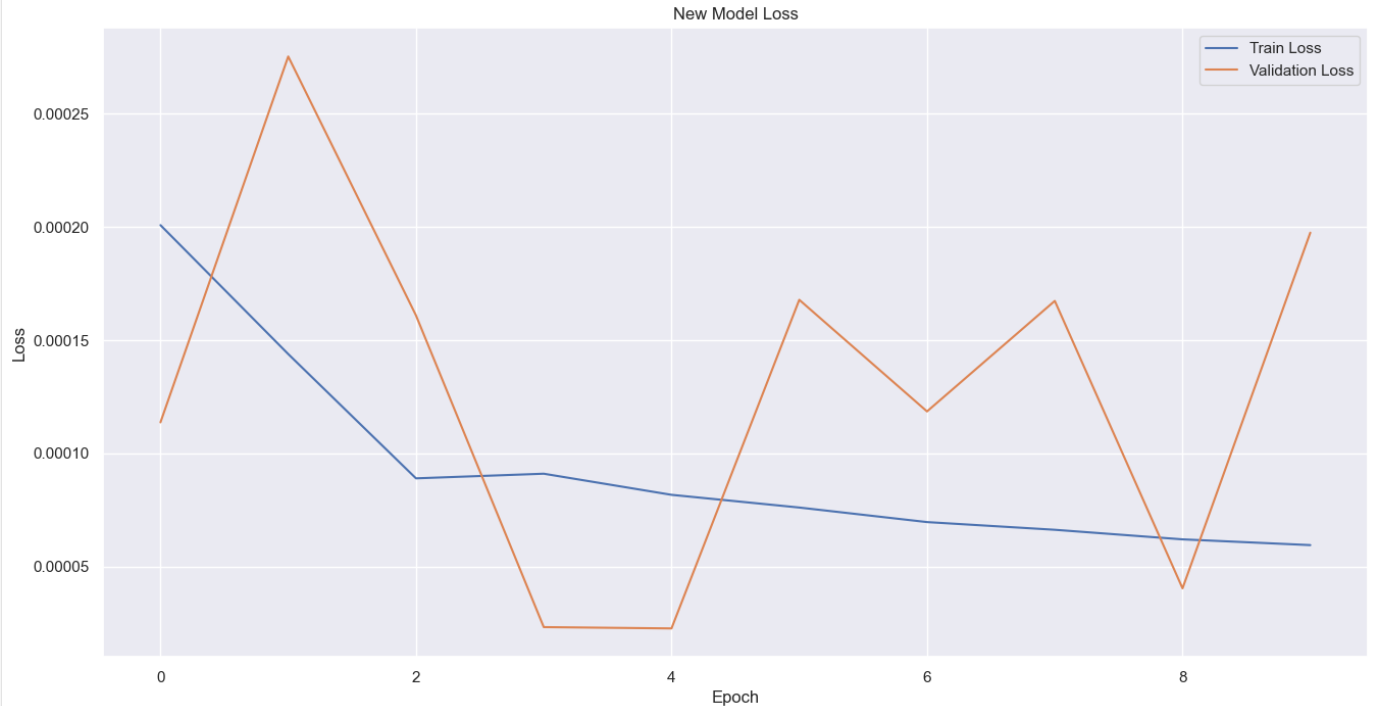
**2650/2650** ━━━━━━━━━━━━━━━━━━━━ **45s** 17ms/step - loss: 6.1781e-05 - val\_loss: 1.6732e-04

Epoch 9/10

**2650/2650** ━━━━━━━━━━━━━━━━━━━━ **45s** 17ms/step - loss: 6.5684e-05 - val\_loss: 4.0412e-05

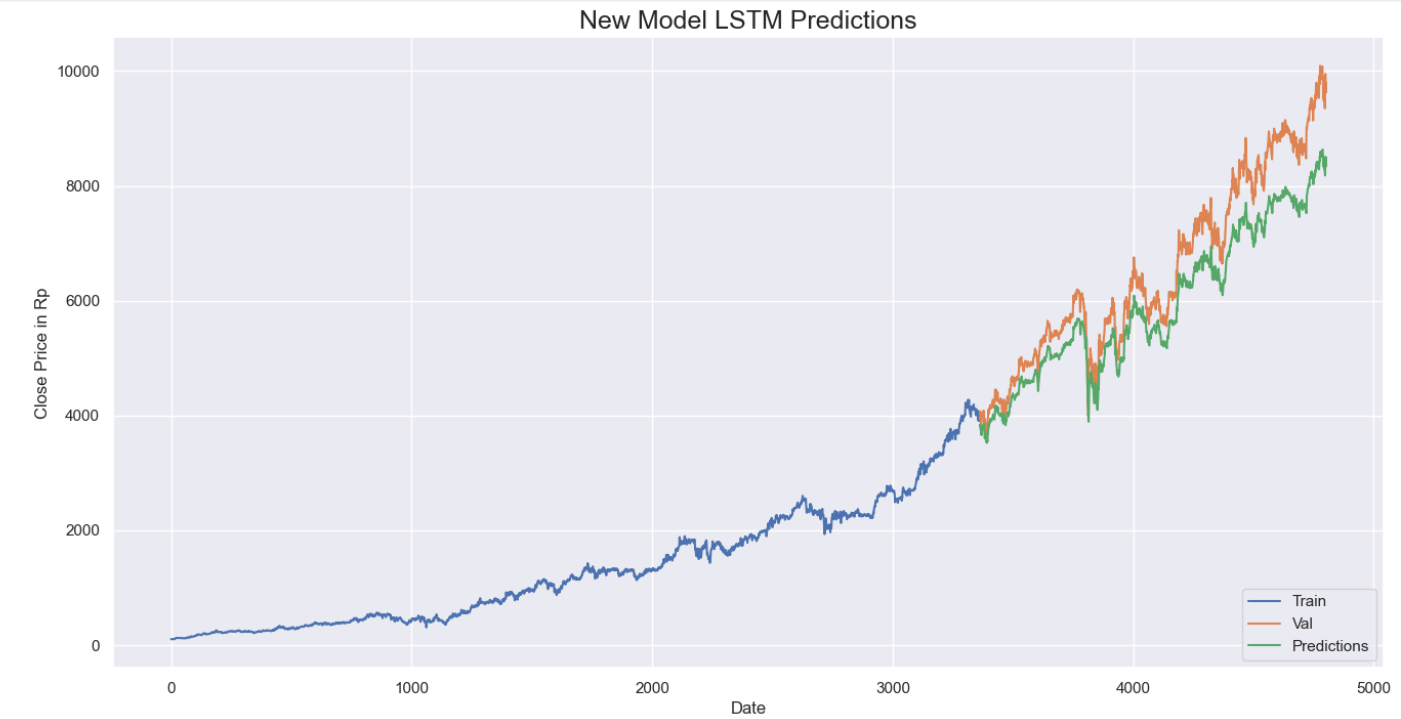
Epoch 10/10

**2650/2650** ━━━━━━━━━━━━━━━━━━━━ **45s** 17ms/step - loss: 5.5397e-05 - val\_loss: 1.9745e-04



**45/45** ━━━━━━━━━━━━━━━━━━━━ **1s** 10ms/step

Prediksi dari model LSTM



***CNN***

**60/60** ━━━━━━━━━━━━━━━━━━━━ **2s** 9ms/step - loss: 0.0104 - val\_loss: 4.2451e-04

Epoch 2/10

**60/60** ━━━━━━━━━━━━━━━━━━━━ **0s** 6ms/step - loss: 4.0925e-04 - val\_loss: 3.2941e-04

Epoch 3/10

**60/60** ━━━━━━━━━━━━━━━━━━━━ **0s** 6ms/step - loss: 3.5900e-04 - val\_loss: 2.9186e-04

Epoch 4/10

**60/60** ━━━━━━━━━━━━━━━━━━━━ **0s** 6ms/step - loss: 2.8467e-04 - val\_loss: 2.3070e-04

Epoch 5/10

**60/60** ━━━━━━━━━━━━━━━━━━━━ **0s** 7ms/step - loss: 4.3373e-04 - val\_loss: 2.0494e-04

Epoch 6/10

**60/60** ━━━━━━━━━━━━━━━━━━━━ **0s** 6ms/step - loss: 2.2747e-04 - val\_loss: 1.8250e-04

Epoch 7/10

**60/60** ━━━━━━━━━━━━━━━━━━━━ **0s** 7ms/step - loss: 1.9175e-04 - val\_loss: 1.8704e-04

Epoch 8/10

**60/60** ━━━━━━━━━━━━━━━━━━━━ **0s** 6ms/step - loss: 1.8548e-04 - val\_loss: 2.6267e-04

Epoch 9/10

**60/60** ━━━━━━━━━━━━━━━━━━━━ **0s** 6ms/step - loss: 2.0209e-04 - val\_loss: 3.2349e-04

Epoch 10/10

**60/60** ━━━━━━━━━━━━━━━━━━━━ **0s** 6ms/step - loss: 2.0203e-04 - val\_loss: 1.4645e-04

Training fold 2...

Epoch 1/10

**60/60** ━━━━━━━━━━━━━━━━━━━━ **2s** 11ms/step - loss: 0.0175 - val\_loss: 4.0645e-04

Epoch 2/10

**60/60** ━━━━━━━━━━━━━━━━━━━━ **0s** 6ms/step - loss: 3.6312e-04 - val\_loss: 3.6824e-04

Epoch 3/10

**60/60** ━━━━━━━━━━━━━━━━━━━━ **0s** 6ms/step - loss: 3.5335e-04 - val\_loss: 3.2957e-04

Epoch 4/10

**60/60** ━━━━━━━━━━━━━━━━━━━━ **0s** 6ms/step - loss: 2.7748e-04 - val\_loss: 3.9635e-04

Epoch 5/10

**60/60** ━━━━━━━━━━━━━━━━━━━━ **0s** 6ms/step - loss: 2.5942e-04 - val\_loss: 2.7014e-04

Epoch 6/10

**60/60** ━━━━━━━━━━━━━━━━━━━━ **0s** 7ms/step - loss: 2.2981e-04 - val\_loss: 2.7331e-04

Epoch 7/10

**60/60** ━━━━━━━━━━━━━━━━━━━━ **0s** 6ms/step - loss: 1.9556e-04 - val\_loss: 2.3010e-04

Epoch 8/10

**60/60** ━━━━━━━━━━━━━━━━━━━━ **0s** 5ms/step - loss: 1.9980e-04 - val\_loss: 2.1457e-04

Epoch 9/10

**60/60** ━━━━━━━━━━━━━━━━━━━━ **0s** 6ms/step - loss: 1.8236e-04 - val\_loss: 2.4954e-04

Epoch 10/10

**60/60** ━━━━━━━━━━━━━━━━━━━━ **0s** 6ms/step - loss: 1.4946e-04 - val\_loss: 2.1561e-04

Training fold 3...

Epoch 1/10

**60/60** ━━━━━━━━━━━━━━━━━━━━ **2s** 8ms/step - loss: 0.0328 - val\_loss: 4.3205e-04

Epoch 2/10

**60/60** ━━━━━━━━━━━━━━━━━━━━ **0s** 5ms/step - loss: 4.2182e-04 - val\_loss: 3.3682e-04

Epoch 3/10

**60/60** ━━━━━━━━━━━━━━━━━━━━ **0s** 5ms/step - loss: 3.2676e-04 - val\_loss: 2.8495e-04

Epoch 4/10

**60/60** ━━━━━━━━━━━━━━━━━━━━ **0s** 5ms/step - loss: 3.0557e-04 - val\_loss: 2.3979e-04

Epoch 5/10

**60/60** ━━━━━━━━━━━━━━━━━━━━ **0s** 5ms/step - loss: 2.3429e-04 - val\_loss: 2.0558e-04

Epoch 6/10

**60/60** ━━━━━━━━━━━━━━━━━━━━ **0s** 5ms/step - loss: 2.3258e-04 - val\_loss: 1.9137e-04

Epoch 7/10

**60/60** ━━━━━━━━━━━━━━━━━━━━ **0s** 5ms/step - loss: 2.1547e-04 - val\_loss: 2.2810e-04

Epoch 8/10

**60/60** ━━━━━━━━━━━━━━━━━━━━ **0s** 5ms/step - loss: 2.0205e-04 - val\_loss: 1.7750e-04

Epoch 9/10

**60/60** ━━━━━━━━━━━━━━━━━━━━ **0s** 6ms/step - loss: 1.9099e-04 - val\_loss: 1.4258e-04

Epoch 10/10

**60/60** ━━━━━━━━━━━━━━━━━━━━ **0s** 5ms/step - loss: 1.6263e-04 - val\_loss: 2.2799e-04

Training fold 4...

Epoch 1/10

**60/60** ━━━━━━━━━━━━━━━━━━━━ **1s** 7ms/step - loss: 0.0131 - val\_loss: 3.9419e-04

Epoch 2/10

**60/60** ━━━━━━━━━━━━━━━━━━━━ **0s** 4ms/step - loss: 4.4347e-04 - val\_loss: 3.3760e-04

Epoch 3/10

**60/60** ━━━━━━━━━━━━━━━━━━━━ **0s** 6ms/step - loss: 3.5596e-04 - val\_loss: 2.7451e-04

Epoch 4/10

**60/60** ━━━━━━━━━━━━━━━━━━━━ **0s** 6ms/step - loss: 3.7303e-04 - val\_loss: 2.2733e-04

Epoch 5/10

**60/60** ━━━━━━━━━━━━━━━━━━━━ **0s** 5ms/step - loss: 2.5719e-04 - val\_loss: 1.8770e-04

Epoch 6/10

**60/60** ━━━━━━━━━━━━━━━━━━━━ **0s** 5ms/step - loss: 2.9101e-04 - val\_loss: 1.9986e-04

Epoch 7/10

**60/60** ━━━━━━━━━━━━━━━━━━━━ **0s** 5ms/step - loss: 2.4874e-04 - val\_loss: 2.0382e-04

Epoch 8/10

**60/60** ━━━━━━━━━━━━━━━━━━━━ **0s** 5ms/step - loss: 1.8825e-04 - val\_loss: 1.6345e-04

Epoch 9/10

**60/60** ━━━━━━━━━━━━━━━━━━━━ **0s** 5ms/step - loss: 2.0899e-04 - val\_loss: 1.5113e-04

Epoch 10/10

**60/60** ━━━━━━━━━━━━━━━━━━━━ **0s** 5ms/step - loss: 1.8478e-04 - val\_loss: 1.6340e-04

Training fold 5...

Epoch 1/10

**60/60** ━━━━━━━━━━━━━━━━━━━━ **1s** 7ms/step - loss: 0.0185 - val\_loss: 3.8678e-04

Epoch 2/10

**60/60** ━━━━━━━━━━━━━━━━━━━━ **0s** 5ms/step - loss: 3.4268e-04 - val\_loss: 3.3958e-04

Epoch 3/10

**60/60** ━━━━━━━━━━━━━━━━━━━━ **0s** 5ms/step - loss: 3.4609e-04 - val\_loss: 3.4186e-04

Epoch 4/10

**60/60** ━━━━━━━━━━━━━━━━━━━━ **0s** 5ms/step - loss: 3.1811e-04 - val\_loss: 2.9636e-04

Epoch 5/10

**60/60** ━━━━━━━━━━━━━━━━━━━━ **0s** 5ms/step - loss: 2.6225e-04 - val\_loss: 2.6557e-04

Epoch 6/10

**60/60** ━━━━━━━━━━━━━━━━━━━━ **0s** 5ms/step - loss: 2.7085e-04 - val\_loss: 2.4964e-04

Epoch 7/10

**60/60** ━━━━━━━━━━━━━━━━━━━━ **0s** 5ms/step - loss: 2.5200e-04 - val\_loss: 2.0133e-04

Epoch 8/10

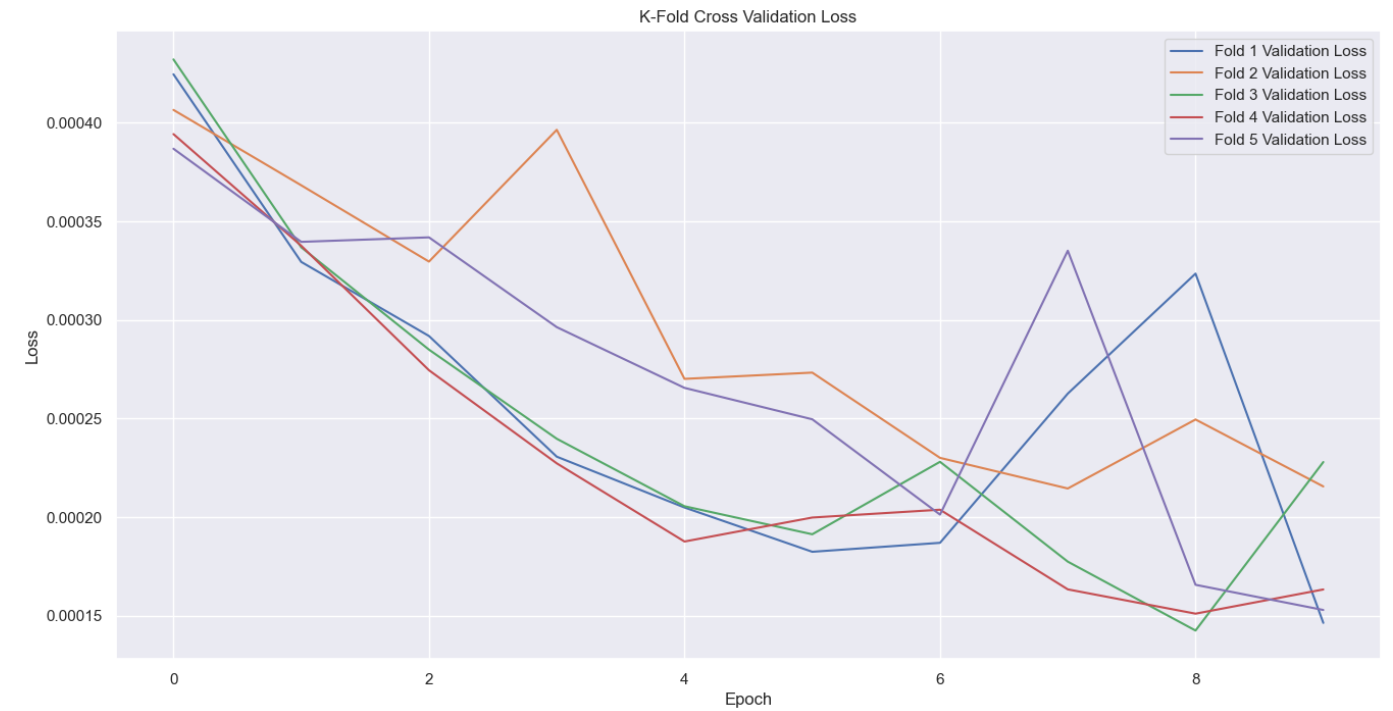
**60/60** ━━━━━━━━━━━━━━━━━━━━ **0s** 5ms/step - loss: 2.1944e-04 - val\_loss: 3.3509e-04

Epoch 9/10

**60/60** ━━━━━━━━━━━━━━━━━━━━ **0s** 5ms/step - loss: 2.6426e-04 - val\_loss: 1.6576e-04

Epoch 10/10

**60/60** ━━━━━━━━━━━━━━━━━━━━ **0s** 5ms/step - loss: 2.0114e-04 - val\_loss: 1.5300e-04



Epoch 1/10

**50/50** ━━━━━━━━━━━━━━━━━━━━ **1s** 8ms/step - loss: 0.0039 - val\_loss: 0.0034

Epoch 2/10

**50/50** ━━━━━━━━━━━━━━━━━━━━ **0s** 6ms/step - loss: 5.9941e-05 - val\_loss: 0.0015

Epoch 3/10

**50/50** ━━━━━━━━━━━━━━━━━━━━ **0s** 5ms/step - loss: 4.9012e-05 - val\_loss: 0.0017

Epoch 4/10

**50/50** ━━━━━━━━━━━━━━━━━━━━ **0s** 5ms/step - loss: 4.5274e-05 - val\_loss: 0.0013

Epoch 5/10

**50/50** ━━━━━━━━━━━━━━━━━━━━ **0s** 5ms/step - loss: 4.2528e-05 - val\_loss: 0.0017

Epoch 6/10

**50/50** ━━━━━━━━━━━━━━━━━━━━ **0s** 6ms/step - loss: 4.7634e-05 - val\_loss: 0.0013

Epoch 7/10

**50/50** ━━━━━━━━━━━━━━━━━━━━ **0s** 6ms/step - loss: 3.3159e-05 - val\_loss: 8.4506e-04

Epoch 8/10

**50/50** ━━━━━━━━━━━━━━━━━━━━ **0s** 6ms/step - loss: 3.2894e-05 - val\_loss: 0.0013

Epoch 9/10

**50/50** ━━━━━━━━━━━━━━━━━━━━ **0s** 6ms/step - loss: 3.2530e-05 - val\_loss: 0.0018

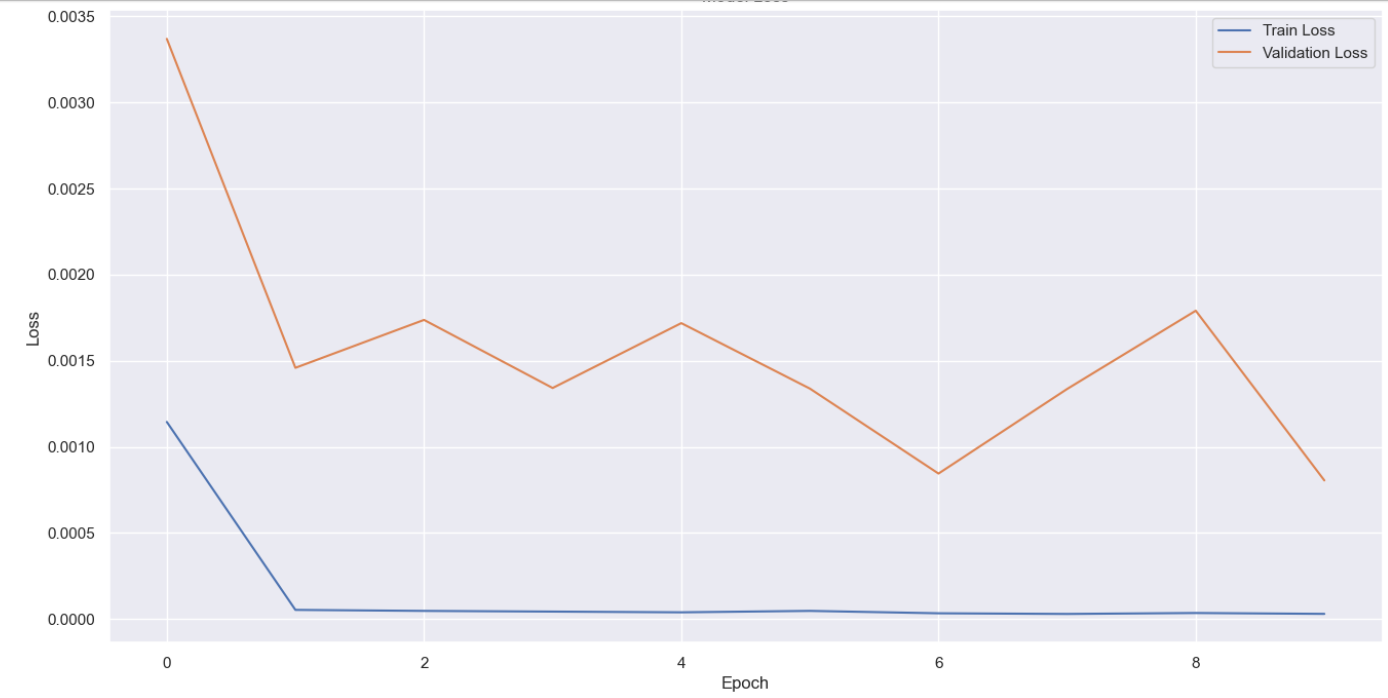
Epoch 10/10

**50/50** ━━━━━━━━━━━━━━━━━━━━ **0s** 5ms/step - loss: 3.2914e-05 - val\_loss: 8.0530e-04

**100/100** ━━━━━━━━━━━━━━━━━━━━ **0s** 2ms/step

**49/49** ━━━━━━━━━━━━━━━━━━━━ **0s** 2ms/step





**49/49** ━━━━━━━━━━━━━━━━━━━━ **0s** 3ms/step

New Test Mean Squared Error (MSE): 80352.81248249924

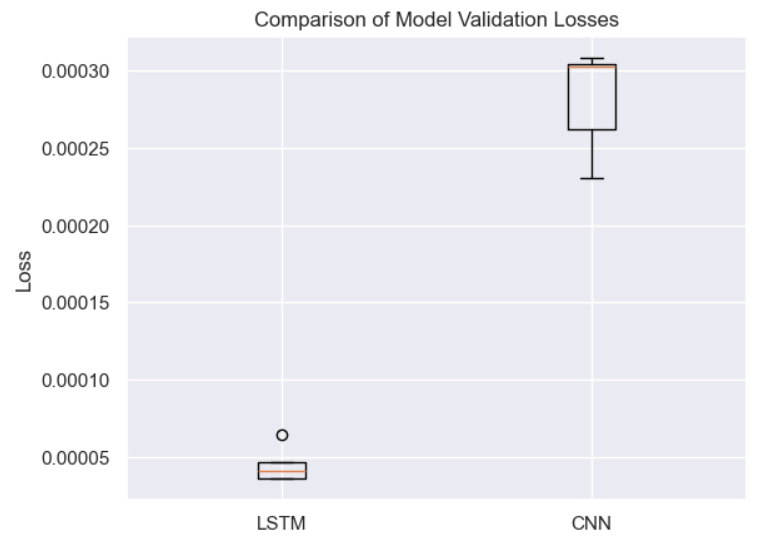
New Test Root Mean Squared Error (RMSE): 283.4657165910884

New Test Mean Absolute Error (MAE): 211.03682951028355

New Test R-squared (R²): 0.9721719693095076

T-Test

Visualisasi perbandingan antara LSTM dan CNN



1. Performa Model

LSTM - Mean: 4.4748e-05,

Std Dev: 1.0442563669904051e-05

CNN - Mean: 0.00028188000000000003,

Std Dev: 3.047539335267061e-05

T-statistic: -15.064969412784944,

P-value: 0.00011314295317299837

Berdasarkan pada uji T, dapat diambil kesilmpulan bahwa ada perbedaan signifikan dari dua model tersebut.

1. Hasil dan Kesimpulan

Pada model ini kita menggunakan Indeks Harga Saham Gabungan

dalam pre-trained model dan Saham BBCA dengan menggunakan

pendekatan RNN dan CNN.

Model yang telah kami buat menunjukkan minimnya loss sehingga

dapat diimplementasikan untuk menjadi acuan para calon investor

yang mungkin masih awam tentang saham.

Kesimpulan : Berdasarkan percobaan implementasi yang telah kita lakukan mungkin masih jauh dari kata sempurna. Untuk model yang telah kita buat, berdasarkan hasil train yang paling cocok untuk memprediksi saham BBCA yaitu menggunakan pendekatan LSTM (Long Short Term Memory) karena memiliki loss yang paling sedikit daripada metode CNN (Convolutional Neural Network).