

Nama: Rafael Nicholas Tanaja

NIM: 2540118656

```
In [1]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import tensorflow as tf

from tensorflow import keras
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras import layers
from tensorflow.keras.layers import LSTM, Dense, GRU, Bidirectional
from sklearn.metrics import mean_absolute_error, mean_squared_error
from tensorflow.keras.wrappers.scikit_learn import KerasClassifier
from sklearn.model_selection import GridSearchCV, ParameterGrid
from tensorflow.keras.optimizers import SGD, Adam, RMSprop
from sklearn.preprocessing import MinMaxScaler
from tqdm import tqdm

import warnings
warnings.filterwarnings('ignore')
```

Data Preprocessing

```
In [2]: amd = pd.read_csv('AMD.csv')
aapl = pd.read_csv('AAPL.csv')
```

```
In [3]: amd['Date'] = pd.to_datetime(amd['Date'])
aapl['Date'] = pd.to_datetime(aapl['Date'])
```

```
In [4]: amdsqrt = amd.copy()

amdsqrt['Open'] = np.sqrt(amd['Open'])
amdsqrt['Close'] = np.sqrt(amd['Close'])
amdsqrt['High'] = np.sqrt(amd['High'])
amdsqrt['Low'] = np.sqrt(amd['Low'])
amdsqrt['Adj Close'] = np.sqrt(amd['Adj Close'])
amdsqrt['Volume'] = np.sqrt(amd['Volume'])
```

```
In [5]: aaplsqrt = aapl.copy()

aaplsqrt['Open'] = np.sqrt(aapl['Open'])
aaplsqrt['Close'] = np.sqrt(aapl['Close'])
aaplsqrt['High'] = np.sqrt(aapl['High'])
aaplsqrt['Low'] = np.sqrt(aapl['Low'])
aaplsqrt['Adj Close'] = np.sqrt(aapl['Adj Close'])
aaplsqrt['Volume'] = np.sqrt(aapl['Volume'])
```

```
In [6]: amdsqrt = amdsqrt.drop(['Open', 'High', 'Low', 'Adj Close', 'Volume'], axis = 1)
aaplsqrt = aaplsqrt.drop(['Open', 'High', 'Low', 'Adj Close', 'Volume'], axis = 1)
```

```
In [7]: amdsqrt.head(5)
```

Out[7]:

	Date	Close
0	1980-03-17	1.773650
1	1980-03-18	1.741049
2	1980-03-19	1.744037
3	1980-03-20	1.735055
4	1980-03-21	1.707825

```
In [8]: amdsqrt_close = amdsqrt['Close'].to_numpy()
aaplsqrt_close = aaplsqrt['Close'].to_numpy()
```

```
In [9]: amdsqrt_close.shape
```

Out[9]: (10098,)

```
In [10]: amdsqrt_close = amdsqrt_close.reshape(-1,1)
aaplsqrt_close = aaplsqrt_close.reshape(-1,1)

scale = MinMaxScaler(feature_range=(0,1))
amdsqrt_close = scale.fit_transform(amdsqrt_close)
aaplsqrt_close = scale.fit_transform(aaplsqrt_close)
```

Window and Split Data

```
In [11]: windowsize = 5
horizon = 5
```

```
In [12]: def labelingwindow(x, horizon):
return x[:, :-horizon], x[:, -horizon:]
```

```
In [13]: def makewindows(x, window_size, horizon):  
    step = np.expand_dims(np.arange(window_size + horizon), axis=0)  
  
    index = step + np.expand_dims(np.arange(len(x) - (window_size+horizon-1)), axis=0).T  
  
    arr = x[index]  
  
    input, output = labelingwindow(arr, horizon)  
  
    return input, output
```

```
In [14]: amdinput, amdoutput = makewindows(amdsqrt_close, window_size, horizon)  
  
    aaplinput, aaploutput = makewindows(aaplsqrt_close, window_size, horizon)
```

```
In [15]: for i in range(5):  
         print(f"Window: {amdinput[i-3]} \n Label: {amdoutput[i-3]}")
```

```
Window: [[0.81206069]
```

```
[0.77818329]
```

```
[0.78688559]
```

```
[0.784283 ]
```

```
[0.80916001]]
```

```
Label: [[0.86314808]
```

```
[0.84472206]
```

```
[0.87775248]
```

```
[0.8672758 ]
```

```
[0.88182441]]
```

```
Window: [[0.77818329]
```

```
[0.78688559]
```

```
[0.784283 ]
```

```
[0.80916001]
```

```
[0.86314808]]
```

```
Label: [[0.84472206]
```

```
[0.87775248]
```

```
[0.8672758 ]
```

```
[0.88182441]
```

```
[0.85461253]]
```

```
Window: [[0.78688559]
```

```
[0.784283 ]
```

```
[0.80916001]
```

```
[0.86314808]
```

```
[0.84472206]]
```

```
Label: [[0.87775248]
```

```
[0.8672758 ]
```

```
[0.88182441]
```

```
[0.85461253]
```

```
[0.83331948]]
```

```
Window: [[0.07823647]
```

```
[0.07314401]
```

```
[0.07361089]
```

```
[0.07220783]
```

```
[0.06795435]]
```

```
Label: [[0.05626521]
```

```
[0.05325825]
```

```
[0.04557904]
```

```
[0.04191158]
```

```
[0.05021498]]
```

```
Window: [[0.07314401]
```

```
[0.07361089]
```

```
[0.07220783]
```

```
[0.06795435]
```

```
[0.05626521]]
```

```
Label: [[0.05325825]
```

```
[0.04557904]
```

```
[0.04191158]
```

```
[0.05021498]
```

```
[0.05021498]]
```

```
In [16]: for i in range(5):  
         print(f"Window: {aaplinput[i-3]} \n Label: {aaploutput[i-3]}")
```

```
Window: [[0.87605324]
```

```
         [0.86495459]
```

```
         [0.86153815]
```

```
         [0.83293178]
```

```
         [0.82376859]]
```

```
Label: [[0.8653334 ]
```

```
         [0.86287738]
```

```
         [0.8859423 ]
```

```
         [0.86688298]
```

```
         [0.87952138]]
```

```
Window: [[0.86495459]
```

```
         [0.86153815]
```

```
         [0.83293178]
```

```
         [0.82376859]
```

```
         [0.8653334 ]]
```

```
Label: [[0.86287738]
```

```
         [0.8859423 ]
```

```
         [0.86688298]
```

```
         [0.87952138]
```

```
         [0.87859783]]
```

```
Window: [[0.86153815]
```

```
         [0.83293178]
```

```
         [0.82376859]
```

```
         [0.8653334 ]
```

```
         [0.86287738]]
```

```
Label: [[0.8859423 ]
```

```
         [0.86688298]
```

```
         [0.87952138]
```

```
         [0.87859783]
```

```
         [0.85450117]]
```

```
Window: [[0.01548906]
```

```
         [0.01441558]
```

```
         [0.0129372 ]
```

```
         [0.01340529]
```

```
         [0.0139596 ]]
```

```
Label: [[0.01513442]
```

```
         [0.01610235]
```

```
         [0.01696297]
```

```
         [0.01805614]
```

```
         [0.02000478]]
```

```
Window: [[0.01441558]
```

```
         [0.0129372 ]
```

```
         [0.01340529]
```

```
         [0.0139596 ]
```

```
         [0.01513442]]
```

```
Label: [[0.01610235]
```

```
         [0.01696297]
```

```
         [0.01805614]
```

```
         [0.02000478]
```

```
         [0.02032143]]
```

```
In [17]: def datasplit(input, output):  
    train_size = int(0.8 * len(input))  
    test_size = train_size + int(0.1 * len(input))  
  
    X_train, y_train = input[:train_size], output[:train_size]  
  
    X_test, y_test = input[train_size:test_size], output[train_size:test_size]  
  
    X_val, y_val = input[test_size:], output[test_size:]  
  
    return X_train, y_train, X_test, y_test, X_val, y_val
```

AMD

```
In [18]: amd_Xtrain, amd_ytrain, amd_Xtest, amd_ytest, amd_Xval, amd_yval = datasplit(amdinput, amdoutput)
```

```
In [19]: print("AMD Split:")  
print(len(amd_Xtrain))  
print(len(amd_ytrain))  
  
print(len(amd_Xtest))  
print(len(amd_ytest))  
  
print(len(amd_Xval))  
print(len(amd_yval))
```

AMD Split:

8071
8071
1008
1008
1010
1010

```
In [20]: amd_ytrain.shape
```

```
Out[20]: (8071, 5, 1)
```

```
In [21]: # amd_ytrain = amd_ytrain[:, 0, :]  
# amd_yval = amd_yval[:, 0, :]  
# amd_ytest = amd_ytest[:, 0, :]
```

```
In [22]: print(amd_ytrain.shape)  
print(amd_yval.shape)  
print(amd_ytest.shape)
```

(8071, 5, 1)
(1010, 5, 1)
(1008, 5, 1)

In [23]: `amd_ytrain`

```
Out[23]: array([[0.05626521],
                [0.05325825],
                [0.04557904],
                [0.04191158],
                [0.05021498]],

               [[0.05325825],
                [0.04557904],
                [0.04191158],
                [0.05021498],
                [0.05021498]],

               [[0.04557904],
                [0.04191158],
                [0.05021498],
                [0.05021498],
                [0.05972913]],

               ...,

               [[0.24984829],
                [0.24848662],
                [0.24794081],
                [0.24465176],
                [0.2441012 ]],

               [[0.24848662],
                [0.24794081],
                [0.24465176],
                [0.2441012 ],
                [0.24382566]],

               [[0.24794081],
                [0.24465176],
                [0.2441012 ],
                [0.24382566],
                [0.2454763 ]])
```

AAPL

In [24]: `aapl_Xtrain, aapl_ytrain, aapl_Xtest, aapl_ytest, aapl_Xval, aapl_yval = datasplit(aaplinput`

```
In [25]: print("AAPL Split:")
print(len(aapl_Xtrain))
print(len(aapl_ytrain))

print(len(aapl_Xtest))
print(len(aapl_ytest))

print(len(aapl_Xval))
print(len(aapl_yval))
```

```
AAPL Split:
7920
7920
990
990
990
990
```

```
In [26]: # aapl_ytrain = aapl_ytrain[:, 0, :]
# aapl_yval = aapl_yval[:, 0, :]
# aapl_ytest = aapl_ytest[:, 0, :]
```

```
In [27]: print(aapl_ytrain.shape)
print(aapl_yval.shape)
print(aapl_ytest.shape)
```

```
(7920, 5, 1)
(990, 5, 1)
(990, 5, 1)
```

Dataset telah di windowing dengan window size = 5 dan horizon = 5, kolom yang diambil hanyalah kolom price dikarenakan feature lainnya tidak terlalu dibutuhkan didalam proses ini.

Transformer Baseline Model

```
In [28]: def positional_encoding(input_dim, d_model):
angle_rates = 1 / np.power(10000, (2 * (np.arange(d_model) // 2)) / np.float32(d_model))
positional_encoding = np.zeros((input_dim, d_model))

for position in range(input_dim):
    for i in range(d_model):
        if i % 2 == 0:
            positional_encoding[position, i] = np.sin(position * angle_rates[i])
        else:
            positional_encoding[position, i] = np.cos(position * angle_rates[i])

return tf.cast(positional_encoding, dtype=tf.float32)
```

```
In [29]: def transformerencoder(inputs, head_size, num_heads, ff_dim, dropout=0):
length = inputs.shape[1]
d_model = inputs.shape[2]

x = inputs

encoding = positional_encoding(length, d_model)
x = x + encoding

x = layers.MultiHeadAttention(key_dim=head_size, num_heads=num_heads, dropout=dropout)(x)

x = layers.Dropout(dropout)(x)
res = x + inputs

x = layers.Conv1D(filters=ff_dim, kernel_size=1, activation = "relu")(x)

x = layers.Dropout(dropout)(x)

return x + res
```



```
In [30]: def buildmodel(
        input_shape,
        head_size,
        num_heads,
        ff_dim,
        num_transformer_blocks,
        mlp_units,
        dropout=0,
        mlp_dropout=0,
    ):
        inputs = keras.Input(shape=input_shape)
        x = inputs

        for _ in range(num_transformer_blocks):
            x = transformerencoder(x, head_size, num_heads, ff_dim, dropout)

        x = layers.GlobalAveragePooling1D(data_format="channels_first")(x)
        for dim in mlp_units:
            x = layers.Dense(dim, activation="elu")(x)
            x = layers.Dropout(mlp_dropout)(x)
        outputs = layers.Dense(horizon, activation="linear")(x)
        return keras.Model(inputs, outputs)

def lrscheduler(epoch, lr, warmup_epochs=30, decay_epochs=100, initial_lr=1e-6, base_lr=1e-3):
    if epoch <= warmup_epochs:
        pct = epoch / warmup_epochs
        return ((base_lr - initial_lr) * pct) + initial_lr

    if epoch > warmup_epochs and epoch < warmup_epochs+decay_epochs:
        pct = 1 - ((epoch - warmup_epochs) / decay_epochs)
        return ((base_lr - min_lr) * pct) + min_lr

    return min_lr
```

```
In [31]: callbacks = [keras.callbacks.EarlyStopping(patience=10, restore_best_weights=True), keras.ca
```

```
In [32]: input_shape_amd = amd_Xtrain.shape[1:]
        print(input_shape_amd)
```

```
(5, 1)
```

```
In [33]: input_shape_aapl = aapl_Xtrain.shape[1:]
        print(input_shape_aapl)
```

```
(5, 1)
```

```
In [45]: def evaluatemodel(model, name, data, X_test, y_test):  
    testpred = model.predict(X_test)  
    testpred = testpred.flatten()  
  
    y_test = y_test.flatten()  
  
    testrmse = np.sqrt(np.mean((testpred - y_test)) ** 2)  
    testmae = mean_absolute_error(y_test, testpred)  
  
    print(f'\n{name} - Test:')  
    print('RMSE: %.8f ' % (testrmse))  
    print('MAE: %.8f ' % (testmae))  
  
    plt.figure(figsize=(20, 10))  
    plt.plot(testpred,color='red', label='Prediction Test Data')  
    plt.plot(y_test,color='green', label='y_test')  
    plt.title(f'{name} - Test')  
    plt.xlabel('Number of Days')  
    plt.ylabel('Close Price')  
    plt.legend(loc='upper left')  
    plt.show()
```

In [35]: `from tensorflow.keras import layers`

```
model = buildmodel(
    input_shape_amd,
    head_size=46,
    num_heads=60,
    ff_dim=55,
    num_transformer_blocks=5,
    mlp_units=[256],
    mlp_dropout=0.4,
    dropout=0.14
)

model.compile(
    loss="mean_squared_error",
    optimizer=keras.optimizers.Adam(learning_rate=1e-4),
    metrics=["mean_squared_error"],
)

history = model.fit(
    amd_Xtrain,
    amd_ytrain,
    validation_split=0.2,
    epochs=20,
    batch_size=64,
    callbacks=callbacks
)
```

```
101/101 [=====] - 18s 177ms/step - loss: 0.0025 - mean_squared_e
rror: 0.0025 - val_loss: 5.5496e-04 - val_mean_squared_error: 5.5496e-04 - lr: 3.3400e-04
Epoch 12/20
101/101 [=====] - 18s 178ms/step - loss: 0.0023 - mean_squared_e
rror: 0.0023 - val_loss: 0.0010 - val_mean_squared_error: 0.0010 - lr: 3.6730e-04
Epoch 13/20
101/101 [=====] - 18s 180ms/step - loss: 0.0020 - mean_squared_e
rror: 0.0020 - val_loss: 4.6277e-04 - val_mean_squared_error: 4.6277e-04 - lr: 4.0060e-04
Epoch 14/20
101/101 [=====] - 18s 179ms/step - loss: 0.0019 - mean_squared_e
rror: 0.0019 - val_loss: 0.0015 - val_mean_squared_error: 0.0015 - lr: 4.3390e-04
Epoch 15/20
101/101 [=====] - 18s 178ms/step - loss: 0.0021 - mean_squared_e
rror: 0.0021 - val_loss: 0.0012 - val_mean_squared_error: 0.0012 - lr: 4.6720e-04
Epoch 16/20
101/101 [=====] - 18s 179ms/step - loss: 0.0018 - mean_squared_e
rror: 0.0018 - val_loss: 0.0016 - val_mean_squared_error: 0.0016 - lr: 5.0050e-04
Epoch 17/20
101/101 [=====] - 18s 179ms/step - loss: 0.0026 - mean_squared_e
rror: 0.0026 - val_loss: 3.4837e-04 - val_mean_squared_error: 3.4837e-04 - lr: 5.3380e-04
```

```
In [36]: model2 = buildmodel(
    input_shape_aapl,
    head_size=46,
    num_heads=60,
    ff_dim=55,
    num_transformer_blocks=5,
    mlp_units=[256],
    mlp_dropout=0.4,
    dropout=0.14,
)

model2.compile(
    loss="mean_squared_error",
    optimizer=keras.optimizers.Adam(learning_rate=1e-4),
    metrics=["mean_squared_error"],
)

history = model2.fit(
    aapl_Xtrain,
    aapl_ytrain,
    validation_split=0.2,
    epochs=20,
    batch_size=64,
    callbacks=callbacks
)
```

```
Epoch 10/20
99/99 [=====] - 20s 204ms/step - loss: 1.1236e-04 - mean_squared_error: 1.1236e-04 - val_loss: 2.8840e-04 - val_mean_squared_error: 2.8840e-04 - lr: 5.0050e-04
Epoch 17/20
99/99 [=====] - 21s 217ms/step - loss: 1.1072e-04 - mean_squared_error: 1.1072e-04 - val_loss: 9.1685e-04 - val_mean_squared_error: 9.1685e-04 - lr: 5.3380e-04
Epoch 18/20
99/99 [=====] - 20s 207ms/step - loss: 1.2719e-04 - mean_squared_error: 1.2719e-04 - val_loss: 3.2733e-04 - val_mean_squared_error: 3.2733e-04 - lr: 5.6710e-04
Epoch 19/20
99/99 [=====] - 18s 180ms/step - loss: 9.6154e-05 - mean_squared_error: 9.6154e-05 - val_loss: 6.1412e-05 - val_mean_squared_error: 6.1412e-05 - lr: 6.0040e-04
Epoch 20/20
99/99 [=====] - 18s 177ms/step - loss: 9.4096e-05 - mean_squared_error: 9.4096e-05 - val_loss: 9.9800e-05 - val_mean_squared_error: 9.9800e-05 - lr: 6.3370e-04
```

Evaluate Baseline Model

```
In [37]: evaluatemodel(model, "Evaluation on Baseline Model AMD", amsqrt_close, amd_Xtest, amd_ytest
```

```
32/32 [=====] - 1s 34ms/step
```

Evaluation on Baseline Model AMD - Test:

RMSE: 0.00581307

MAE: 0.00884268



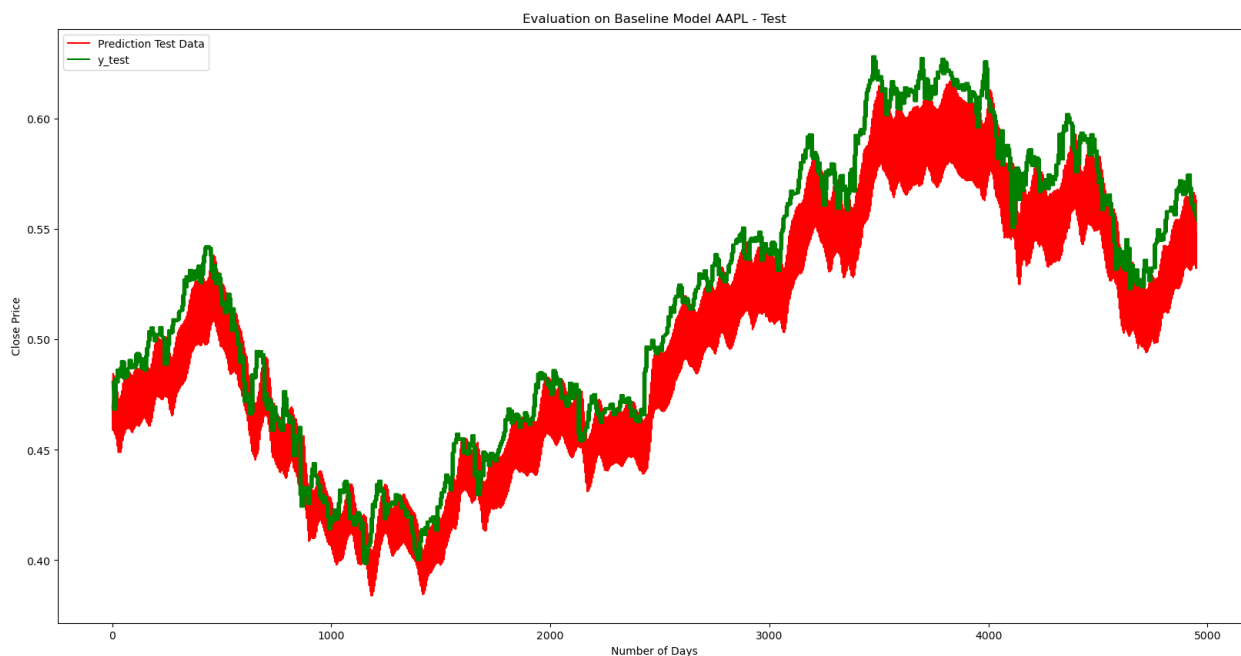
In [38]: `evaluatemodel(model2, "Evaluation on Baseline Model AAPL", aaplqrt_close, aapl_xtest, aapl_ytest)`

31/31 [=====] - 1s 32ms/step

Evaluation on Baseline Model AAPL - Test:

RMSE: 0.01837979

MAE: 0.01971346



Modified Model

```
In [39]: def modifiedtransformerencoder(inputs, head_size, num_heads, ff_dim, dropout=0):
    length = inputs.shape[1]
    embed = inputs.shape[2]

    x = layers.LayerNormalization(epsilon=1e-6)(inputs)

    x = layers.MultiHeadAttention(key_dim=head_size, num_heads=num_heads, dropout=dropout)(x)

    x = layers.Dropout(dropout)(x)
    res = x + inputs

    x = layers.LayerNormalization(epsilon=1e-6)(res)
    encoding = positional_encoding(length, embed)

    x = layers.Conv1D(ff_dim, kernel_size=1, activation = "relu")(x)

    x = layers.Dropout(dropout)(x)

    x = layers.Conv1D(inputs.shape[-1], kernel_size=1)(x)

    return x + res
```

Pada modified transformer, saya menambahkan Layer Normalization dan Convolutional layer. Hal ini dilakukan untuk mengatasi underfitting.

```
In [46]: def buildmodel(
        input_shape,
        head_size,
        num_heads,
        ff_dim,
        num_transformer_blocks,
        mlp_units,
        dropout=0,
        mlp_dropout=0,
    ):
        inputs = keras.Input(shape=input_shape)
        x = inputs

        for _ in range(num_transformer_blocks):
            x = modifiedtransformerencoder(x, head_size, num_heads, ff_dim, dropout)

        x = layers.GlobalAveragePooling1D(data_format="channels_first")(x)
        for dim in mlp_units:
            x = layers.Dense(dim, activation="elu")(x)
            x = layers.Dropout(mlp_dropout)(x)
        outputs = layers.Dense(5, activation="linear")(x)
        return keras.Model(inputs, outputs)
```

AMD

```
In [47]: model1 = buildmodel(
    input_shape_amd,
    head_size=46,
    num_heads=60,
    ff_dim=55,
    num_transformer_blocks=5,
    mlp_units=[128],
    mlp_dropout=0.4,
    dropout=0.2,
)

model1.compile(
    loss="mean_squared_error",
    optimizer=keras.optimizers.Adam(learning_rate=1e-4),
    metrics=["mean_squared_error"],
)

history = model1.fit(
    amd_Xtrain,
    amd_ytrain,
    validation_split=0.2,
    epochs=20,
    batch_size=64,
    callbacks=callbacks
)

101/101 [=====] - 14s 139ms/step - loss: 0.0021 - mean_squared_e
rror: 0.0021 - val_loss: 3.4987e-04 - val_mean_squared_error: 3.4987e-04 - lr: 6.3370e-04
Epoch 20/20
```

Hyperparameter pada mlp_units dari 256 -> 128 agar mengurangi kemungkinan overfitting, dikarenakan sudah menambahkan layer pada transformer model.

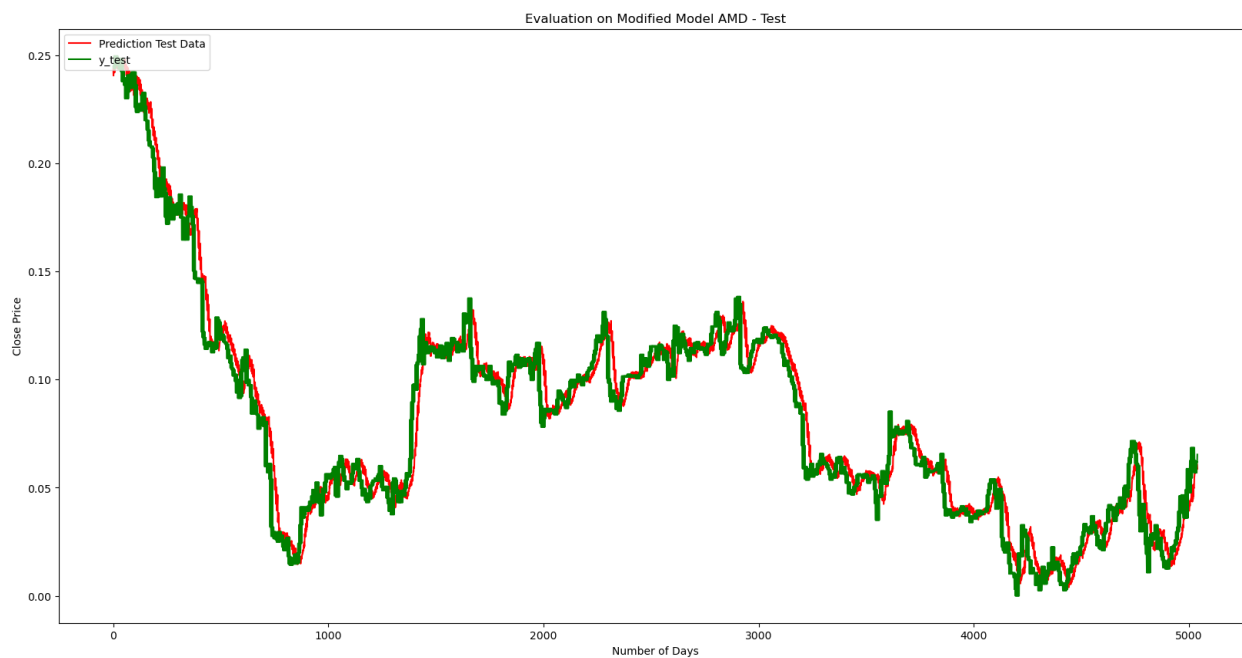

```
In [48]: evaluatemodel(model1, "Evaluation on Modified Model AMD", amdsqrt_close, amd_Xtest, amd_ytes
```

```
32/32 [=====] - 1s 27ms/step
```

Evaluation on Modified Model AMD - Test:

RMSE: 0.00163976

MAE: 0.00710936



AAPL

```

In [49]: model2 = buildmodel(
            input_shape_aapl,
            head_size=24,
            num_heads=32,
            ff_dim=45,
            num_transformer_blocks=5,
            mlp_units=[128],
            mlp_dropout=0.2,
            dropout=0.07,
        )

model2.compile(
    loss="mean_squared_error",
    optimizer=keras.optimizers.Adam(learning_rate=1e-4),
    metrics=["mean_squared_error"],
)

history = model2.fit(
    aapl_Xtrain,
    aapl_ytrain,
    validation_split=0.2,
    epochs=20,
    batch_size=64,
    callbacks=callbacks
)

```

1. 0.0051 - val_loss: 0.1001 - val_mean_squared_error: 0.1001 - lr: 1.0000e-00
 Epoch 2/20
 99/99 [=====] - 4s 38ms/step - loss: 0.0018 - mean_squared_error: 0.0018 - val_loss: 0.0900 - val_mean_squared_error: 0.0900 - lr: 3.4300e-05
 Epoch 3/20
 99/99 [=====] - 4s 36ms/step - loss: 0.0011 - mean_squared_error: 0.0011 - val_loss: 0.0706 - val_mean_squared_error: 0.0706 - lr: 6.7600e-05
 Epoch 4/20
 99/99 [=====] - 4s 36ms/step - loss: 6.3469e-04 - mean_squared_error: 6.3469e-04 - val_loss: 0.0384 - val_mean_squared_error: 0.0384 - lr: 1.0090e-04
 Epoch 5/20
 99/99 [=====] - 4s 37ms/step - loss: 3.1723e-04 - mean_squared_error: 3.1723e-04 - val_loss: 0.0149 - val_mean_squared_error: 0.0149 - lr: 1.3420e-04
 Epoch 6/20
 99/99 [=====] - 4s 37ms/step - loss: 1.3313e-04 - mean_squared_error: 1.3313e-04 - val_loss: 0.0040 - val_mean_squared_error: 0.0040 - lr: 1.6750e-04
 Epoch 7/20
 99/99 [=====] - 4s 36ms/step - loss: 6.4603e-05 - mean_squared_error: 6.4603e-05 - val_loss: 8.2146e-04 - val_mean_squared_error: 8.2146e-04 - lr: 2.0080e-04
 Epoch 8/20
 99/99 [=====] - 4s 36ms/step - loss: 3.1723e-05 - mean_squared_error: 3.1723e-05 - val_loss: 8.2146e-04 - val_mean_squared_error: 8.2146e-04 - lr: 2.0080e-04

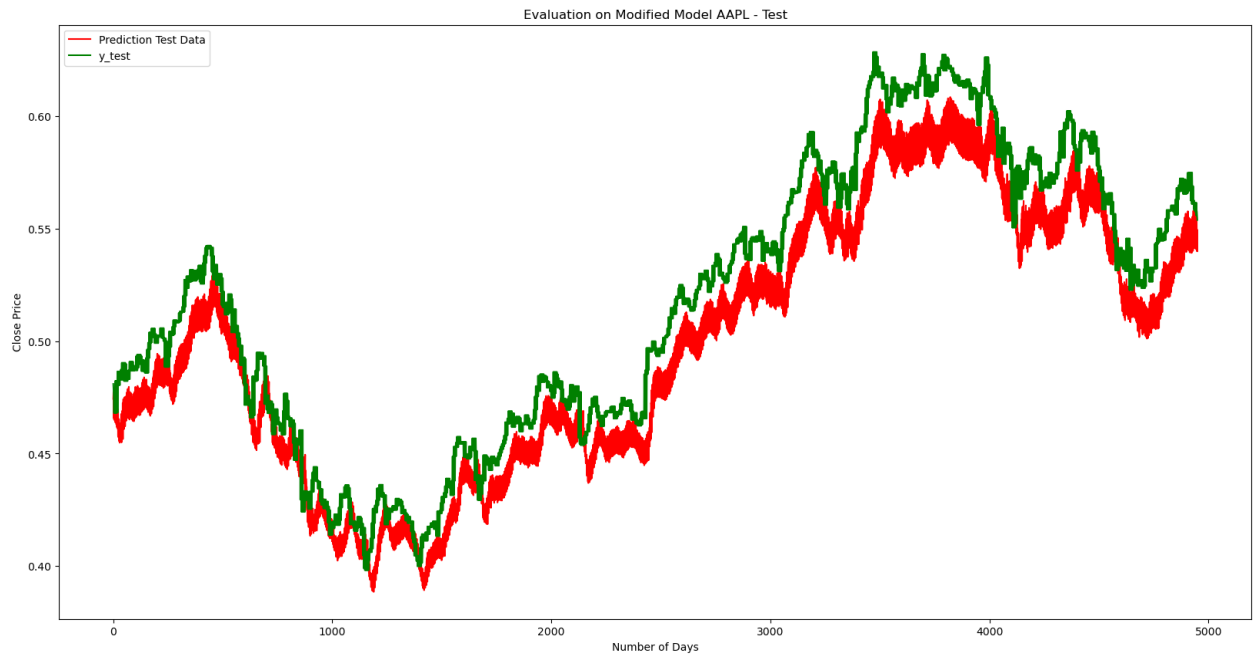
```
In [50]: evaluatemodel(model2, "Evaluation on Modified Model AAPL", aaplqrt_close, aapl_xtest, aapl_
```

```
31/31 [=====] - 1s 9ms/step
```

Evaluation on Modified Model AAPL - Test:

RMSE: 0.01722986

MAE: 0.01780906



Conclusion

Berikut merupakan hasil dari evaluasi baseline:

Evaluation on Baseline Model AMD - Test:

- RMSE: 0.00581307
- MAE: 0.00884268

Evaluation on Baseline Model AAPL - Test:

- RMSE: 0.01837979
- MAE: 0.01971346

Berikut merupakan hasil dari modified baseline:

Evaluation on Modified Model AMD - Test:

- RMSE: 0.00163976
- MAE: 0.00710936

Evaluation on Modified Model AAPL - Test:

- RMSE: 0.01722986
- MAE: 0.01780906

Model modified pada kedua saham memiliki performa yang lebih baik daripada model baseline, hal ini dapat dilihat dari nilai RMSE dan MAE yang lebih kecil pada Modified Model.

