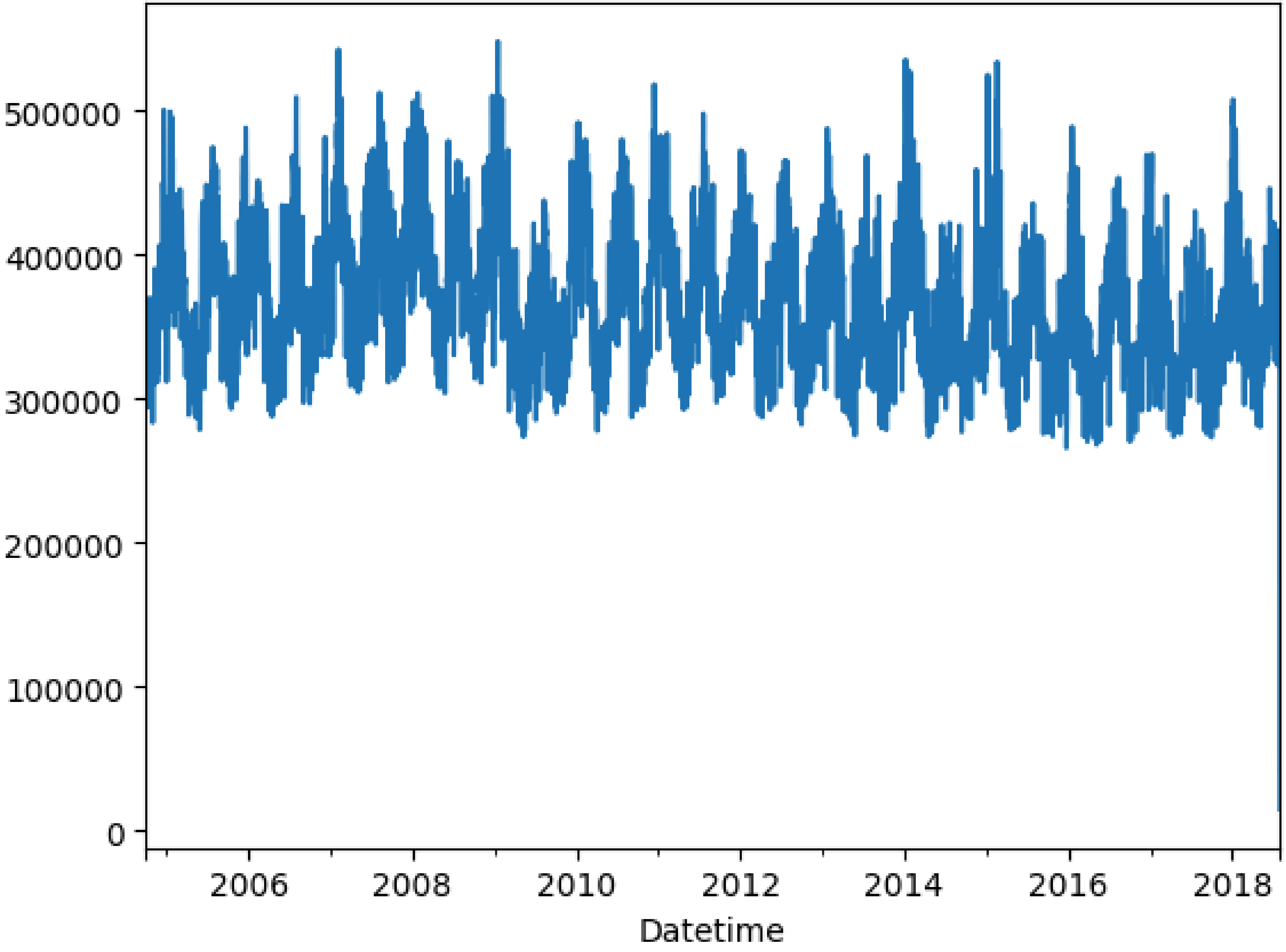
ELECTRIC LOVE

Ahmad Ardra Damarjatri

71486

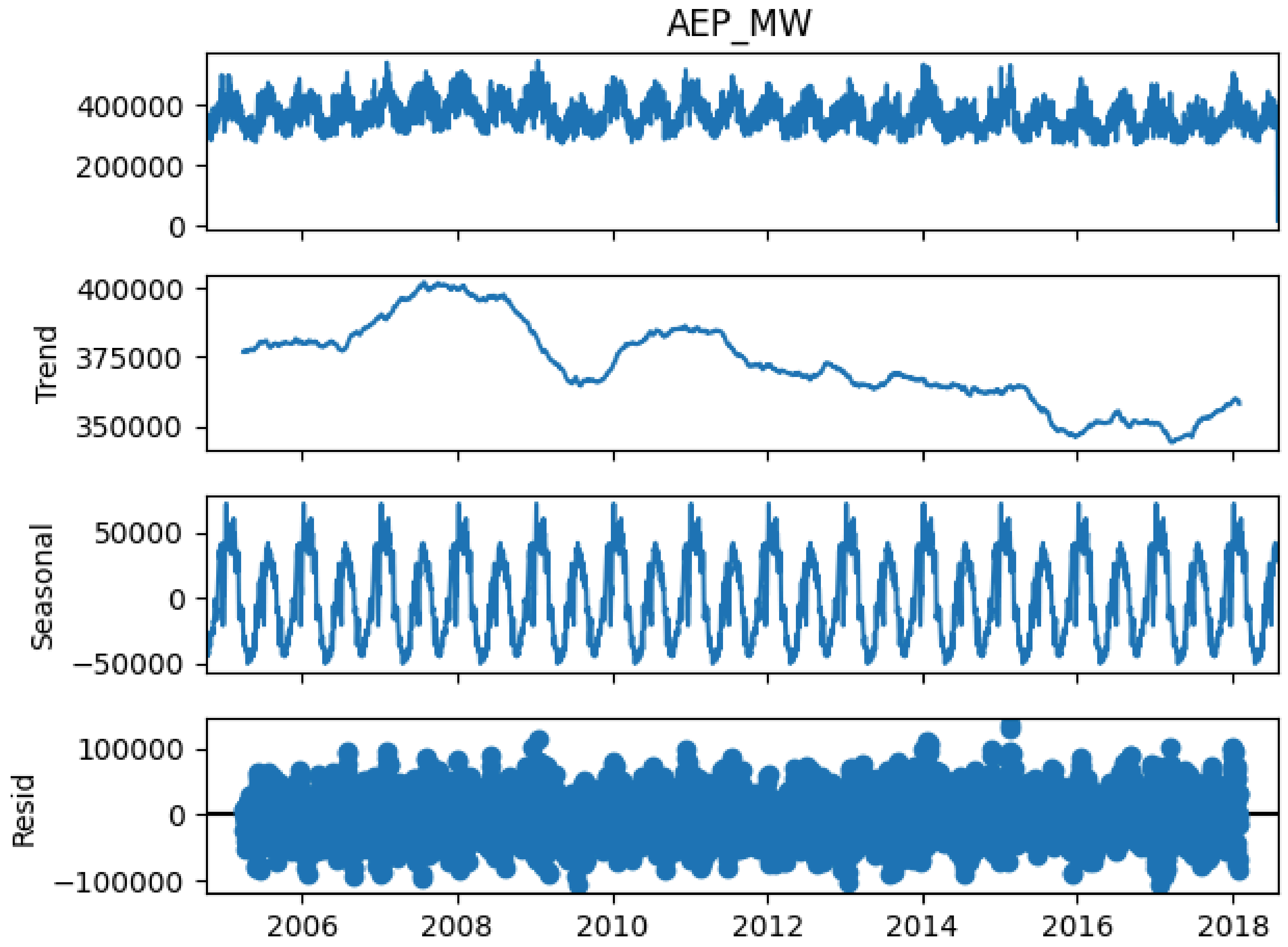
UNIVERSITAS TIMMERMAN

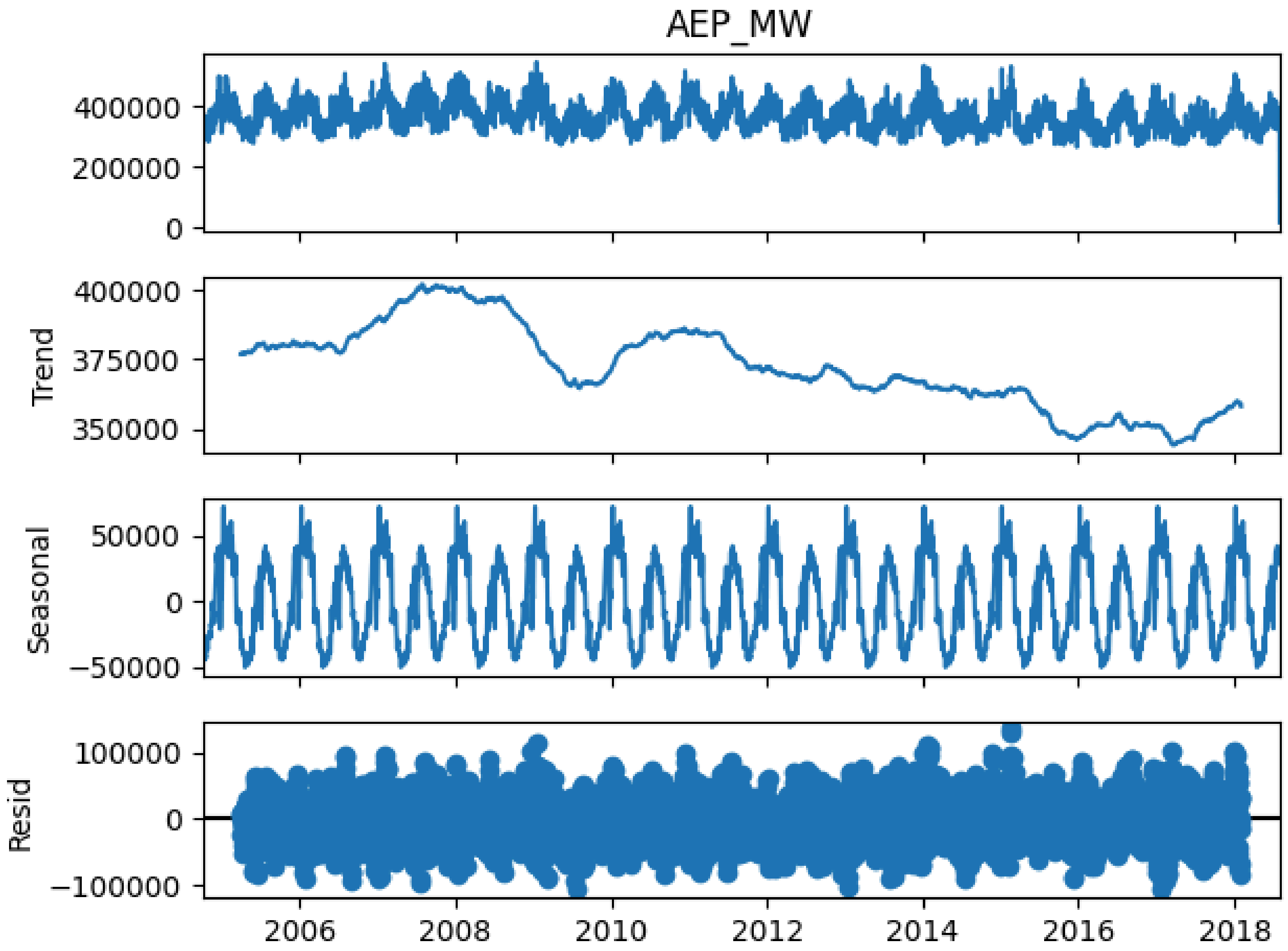
UNIVERSITAS
TIMMERMAN



UNIVERSITAS
TIMMERMAN

Periode None





Periode 365

SEMINAR PROPOSAL | 03

	Values	Metric
0	-7.279596e+00	Test Statistics
1	1.512584e-10	p-value
2	3.000000e+01	No. of lags used
3	5.024000e+03	Number of observations used
4	-3.431652e+00	critical value (1%)
5	-2.862115e+00	critical value (5%)
6	-2.567076e+00	critical value (10%)

Linear Regression

```python
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import numpy as np

# 1. Tambah fitur lag
df_baseline = df.copy()
df_baseline['lag_1'] = df_baseline['AEP_MW'].shift(1)
df_baseline['lag_7'] = df_baseline['AEP_MW'].shift(7)
df_baseline['lag_180'] = df_baseline['AEP_MW'].shift(180)
df_baseline['lag_365'] = df_baseline['AEP_MW'].shift(365)

# 2. Drop baris NaN
df_baseline = df_baseline.dropna()

# 3. Split train/test
train = df_baseline[df_baseline.index < '2014-01-01']
test = df_baseline[df_baseline.index >= '2014-01-01']

X_train = train[['lag_1', 'lag_7','lag_180', 'lag_365']]
y_train = train['AEP_MW']
X_test = test[['lag_1', 'lag_7', 'lag_180','lag_365']]
y_test = test['AEP_MW']

# 4. Linear Regression
model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

# 5. RMSE
rmse_lr = np.sqrt(mean_squared_error(y_test, y_pred))
print(f"✅ Baseline Linear Regression RMSE: {rmse_lr:.4f}")
```

✓ 0.0s

Baseline Linear Regression RMSE: 25335.7495

UNIVERSITAS
TIMMERMAN

GRU Model

GRU RMSE: 23404.7094

BASELINE MODEL

```python
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import GRU, Dropout, Dense

# Buat model GRU
model = Sequential()
model.add(GRU(64, input_shape=(seq_length, X.shape[2])))
model.add(Dropout(0.2))
model.add(Dense(1))

# Compile model
model.compile(optimizer='adam', loss='mse')

# Training
model.fit(X_train, y_train, epochs=20, batch_size=32, validation_split=0.1)
```

```python
# --- Siapkan fitur musiman ---
df['dayofyear'] = df.index.dayofyear
df['sin_day'] = np.sin(2 * np.pi * df['dayofyear'] / 365)
df['cos_day'] = np.cos(2 * np.pi * df['dayofyear'] / 365)

features = ['AEP_MW', 'sin_day', 'cos_day']
data = df[features]

# --- Split & scaling ---
cutoff_date = '2014-01-01'
cutoff_index = df.index.get_loc(cutoff_date)

# Fit scaler hanya ke data sebelum 2014
scaler = MinMaxScaler()
scaler.fit(data.iloc[:cutoff_index])
data_scaled = scaler.transform(data)

# --- Sequence maker ---
def create_sequences(data, seq_length):
    X, y = [], []
    for i in range(len(data) - seq_length):
        X.append(data[i:i+seq_length])
        y.append(data[i+seq_length][0])  # AEP_MW index ke-0
    return np.array(X), np.array(y)

seq_length = 180
X, y = create_sequences(data_scaled, seq_length)

X_train, X_test = X[:cutoff_index - seq_length], X[cutoff_index - seq_length:]
y_train, y_test = y[:cutoff_index - seq_length], y[cutoff_index - seq_length:]
```

LSTM Model

LSTM (WITH SINCOS) RMSE: 23065.33

LSTM (WITHOUT SINCOS) RMSE: 21020.7720

```python
# --- Feature engineering ---
df_feat = df.copy()
for lag in [1, 2, 3, 7, 14, 30]:
    df_feat[f'lag_{lag}'] = df_feat['AEP_MW'].shift(lag)

# Tambah rolling mean
df_feat['rolling_mean_7'] = df_feat['AEP_MW'].rolling(window=7).mean()
df_feat['rolling_mean_30'] = df_feat['AEP_MW'].rolling(window=30).mean()

# Tambah fitur waktu
df_feat['dayofweek'] = df_feat.index.dayofweek
df_feat['month'] = df_feat.index.month
df_feat['dayofyear'] = df_feat.index.dayofyear

df_feat.dropna(inplace=True)

# --- Split train/test ---
train = df_feat[df_feat.index < '2014-01-01']
test = df_feat[df_feat.index >= '2014-01-01']

X_train = train.drop(columns=['AEP_MW'])
y_train = train['AEP_MW']
X_test = test.drop(columns=['AEP_MW'])
y_test = test['AEP_MW']

# --- Train model XGBoost ---
model = xgb.XGBRegressor(n_estimators=100, learning_rate=0.1)
model.fit(X_train, y_train)

# --- Predict & evaluate ---
y_pred = model.predict(X_test)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print("RMSE:", rmse)

# --- Plot hasil ---
plt.figure(figsize=(12,6))
plt.plot(y_test.values, label='Actual')
plt.plot(y_pred, label='Predicted')
plt.title('XGBoost Forecast AEP_MW')
plt.legend()
plt.show()
```

XGBOOST Model

XGBoost RMSE: 19953.5706

UNIVERSITAS
TIMMERMAN

Baseline Model

Baseline Linear Regression RMSE: 25335.7495

GRU RMSE: 23404.7094

LSTM (WITH SINCOS) RMSE: 23065.33

LSTM (WITHOUT SINCOS) RMSE: 21020.7720

XGBoost RMSE: 19953.5706

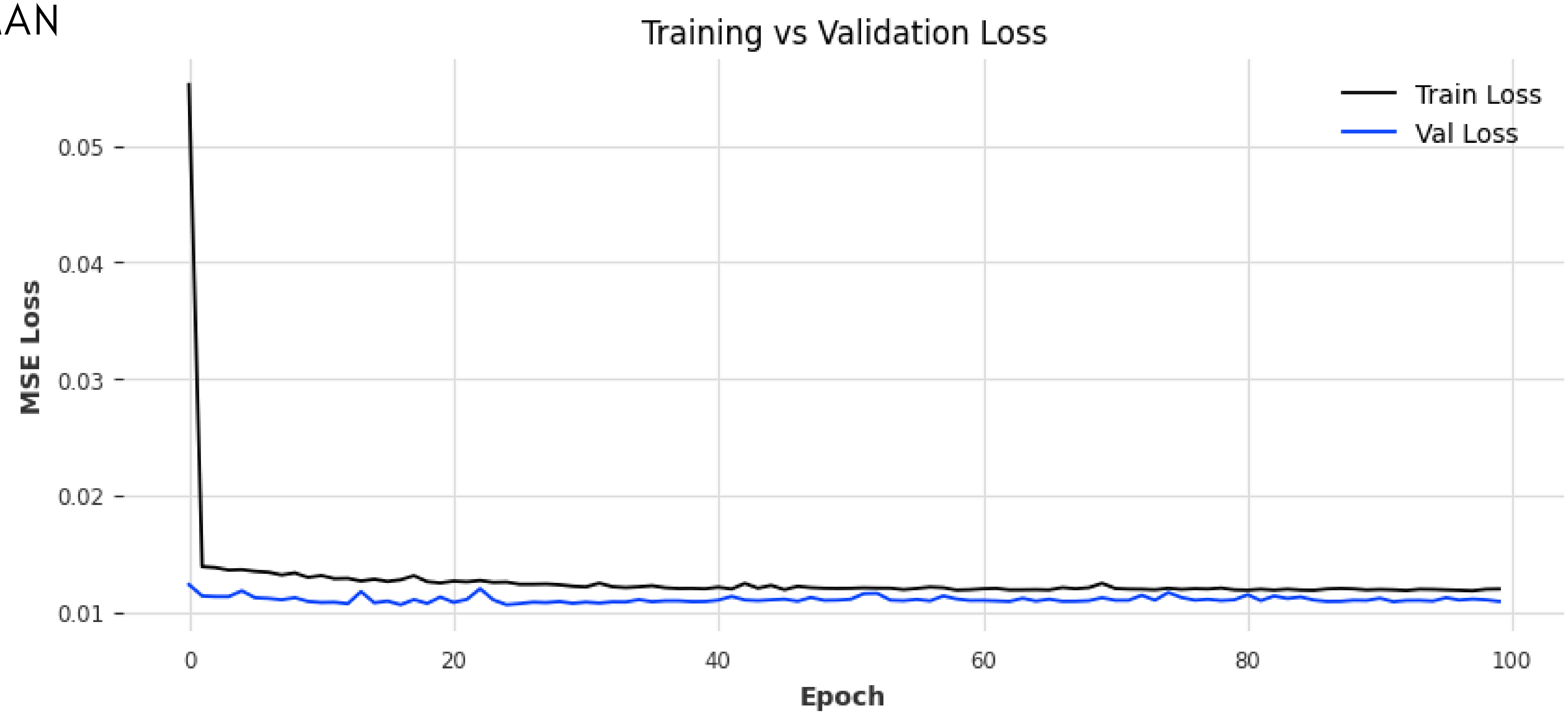

BY THE POWER
OF FRIENDSHIP!
imgflip.com

```python
df_feat = df.copy()
for lag in [1, 2, 3, 7, 14, 30]:
    df_feat[f'lag_{lag}'] = df_feat['AEP_MW'].shift(lag)
df_feat['rolling_mean_7'] = df_feat['AEP_MW'].rolling(7).mean()
df_feat['rolling_mean_30'] = df_feat['AEP_MW'].rolling(30).mean()
df_feat['dayofweek'] = df_feat.index.dayofweek
df_feat['month'] = df_feat.index.month
df_feat['dayofyear'] = df_feat.index.dayofyear
df_feat.dropna(inplace=True)

train = df_feat[df_feat.index < '2014-01-01']
test = df_feat[df_feat.index >= '2014-01-01']

X_train = train.drop(columns=['AEP_MW'])
y_train = train['AEP_MW']
X_test = test.drop(columns=['AEP_MW'])
y_test = test['AEP_MW']

model_xgb = xgb.XGBRegressor(
    n_estimators=500,
    learning_rate=0.02,
    max_depth=6,
    subsample=0.8,
    colsample_bytree=0.8,
    random_state=42
)
model_xgb.fit(X_train, y_train)
xgb_pred = model_xgb.predict(X_test)

# --- Step 2: Residual ---
residual = y_test.values - xgb_pred

# --- Step 3: LSTM untuk Residual ---
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense

# Scale residual
scaler = MinMaxScaler()
residual_scaled = scaler.fit_transform(residual.reshape(-1,1))
```

+ Code + Markdown | Run All Restart Clear All Outputs | View

```python
# Buat sequence
def create_sequences(data, seq_length):
    X, y = [], []
    for i in range(len(data) - seq_length):
        X.append(data[i:i+seq_length])
        y.append(data[i+seq_length])
    return np.array(X), np.array(y)

seq_length = 30
X_lstm, y_lstm = create_sequences(residual_scaled, seq_length)

# Bangun model
model_lstm = Sequential()
model_lstm.add(LSTM(64, activation='relu', input_shape=(seq_length, 1
model_lstm.add(Dense(1))
model_lstm.compile(optimizer='adam', loss='mse')
history = model_lstm.fit(
    X_lstm,
    y_lstm,
    epochs=100,
    batch_size=32,
    validation_split=0.1,    # << TAMBAHAN VALIDATION
    verbose=1,
)
model_lstm.fit(
    X_lstm,
    y_lstm,
    epochs=100,
    batch_size=32,
    validation_split=0.1,    # << TAMBAHAN VALIDATION
    verbose=1,
)
# Prediksi residual
lstm_pred_scaled = model_lstm.predict(X_lstm)
lstm_pred = scaler.inverse_transform(lstm_pred_scaled)

# --- Step 4: Gabungkan hasil ---
xgb_pred = xgb_pred[seq_length:]  # samakan panjang
y_test_final = y_test.values[seq_length:]
final_pred = xgb_pred + lstm_pred.flatten()

rmse_hybrid = np.sqrt(mean_squared_error(y_test_final, final_pred))
print("Hybrid RMSE:", rmse_hybrid)

# --- Visualisasi ---
plt.figure(figsize=(12,6))
plt.plot(y_test_final, label='Actual')
plt.plot(final_pred, label='Hybrid Prediction')
plt.title('Hybrid XGBoost + LSTM Forecast')
plt.legend()
```

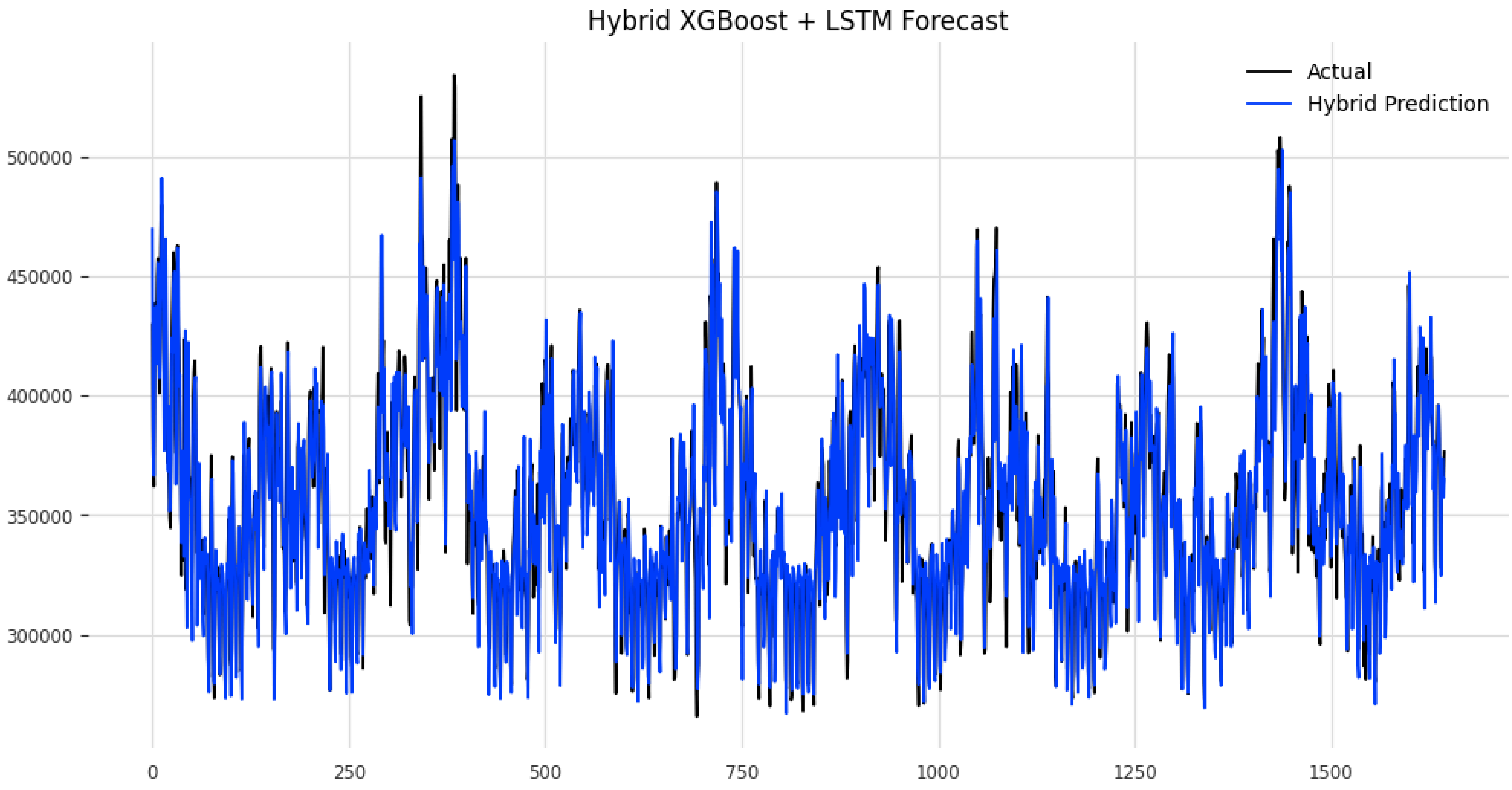
UNIVERSITAS
TIMMERMAN



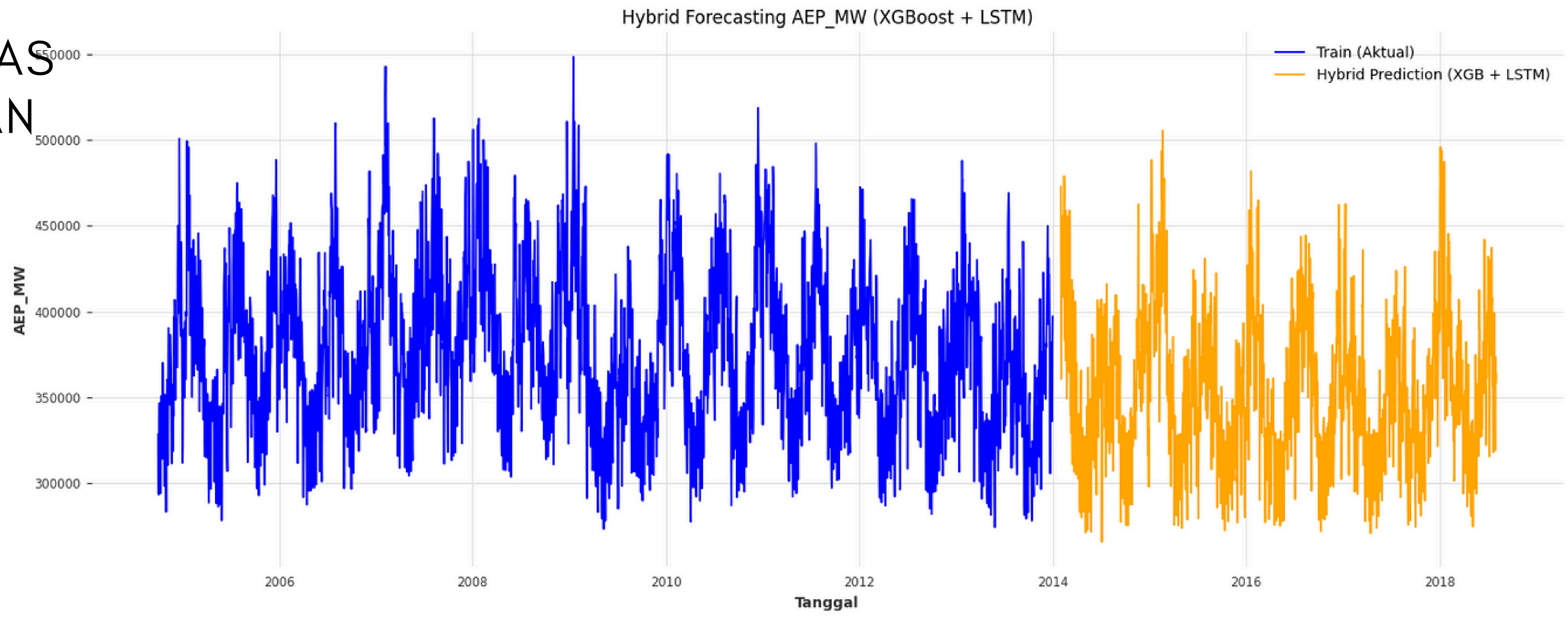

```
Epoch 100/100
47/47 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0115 - val_loss: 0.0114
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step
Hybrid RMSE: 15791.65437953225
```

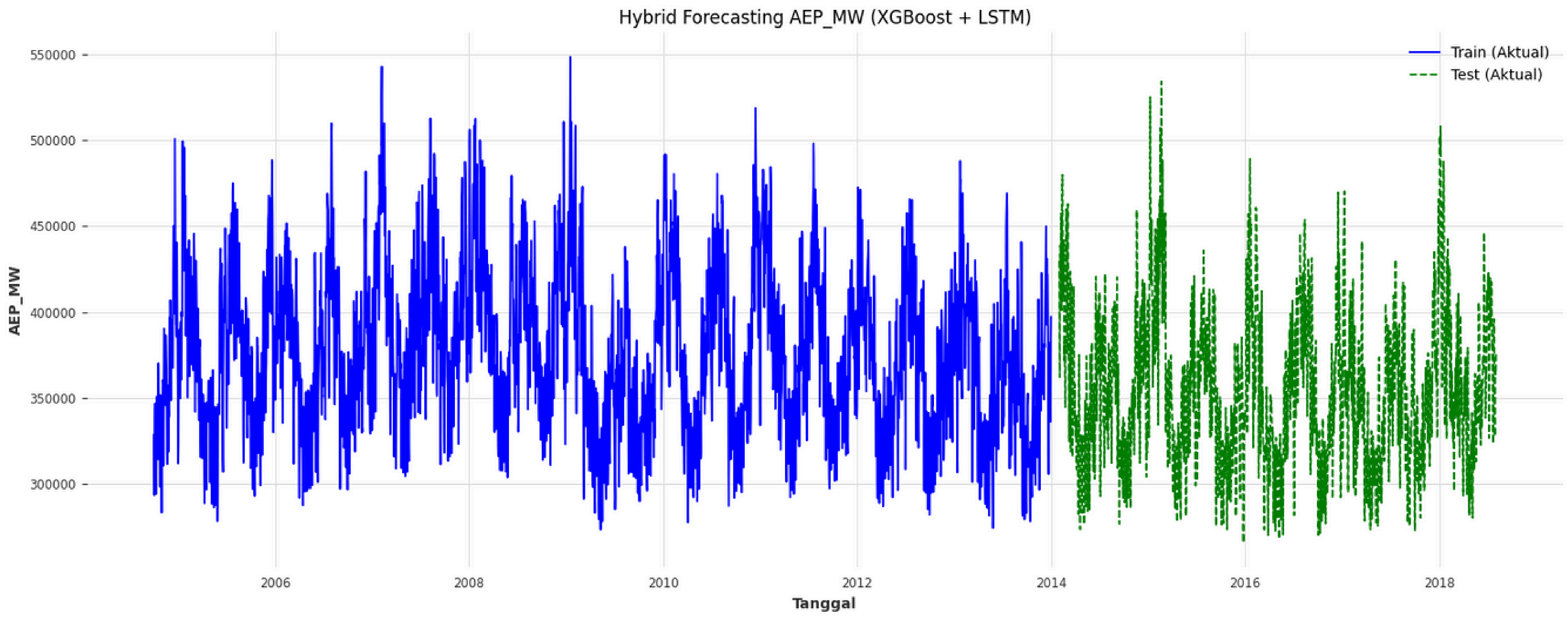
Relative RMSE: 4.42%

UNIVERSITAS TIMMERMAN



UNIVERSITAS TIMMERMAN





Terima Kasih