DAFTAR PUSTAKA

Amanda, Y. R., Aini, M. N., Miyoze, M., & Nugroho, D. O. W. (2022). Prediksi Gempa Bumi di Indonesia Menggunakan R-Shiny. *Jurnal Sains dan Seni ITS, 11*(3), D315–D321. https://doi.org/10.12962/j23373520.v11i3.62562

Ashar, M. I., Nurwidiyanto, M. I., & Danusaputro, H. (2017). Pemodelan bawah permukaan zona subduksi Daerah Selatan Jawa Barat berdasarkan data anomali medan gravitasi. *Youngster Physics Journal*, *6*(4), 382–387.

BPBD. (2018). Pengertian Gempa Bumi, Jenis-Jenis, Penyebab, Akibat, dan Cara Menghadapi Gempa Bumi – BPBD KOTA BANDA ACEH. Diambil 21 Mei 2023, dari https://bpbd.bandaacehkota.go.id/2018/08/05/pengertian-gempa-bumi-jenis-jenis-penyebab-akibat-dan-cara-menghadapi-gempa-bumi/

Chevalier, G. (2018). *English: The Long Short-Term Memory (LSTM) cell can process data sequentially and keep its hidden state through time.* Diambil dari https://commons.wikimedia.org/wiki/File:The_LSTM_cell.png

Cui, X., Li, Z., & Hu, Y. (2022). *Similarity of shallow and deep earthquakes in seismic moment release* (preprint). In Review. https://doi.org/10.21203/rs.3.rs-1877440/v1

Daryono. (2022). Aktivitas Gempabumi Di Indonesia Tahun 2022. Pusat Gempabumi dan Tsunami - Kedeputian Bidang Geofisika, Badan Meteorologi Klimatologi dan Geofisika (BMGK). Diambil dari https://www.its.ac.id/tgeofisika/wp-content/uploads/sites/33/2022/12/REFLEKSI-GEMPA-2022-DARYONO-BMKG.pdf

Davtalab, R. (2023). What's the acceptable value of Root Mean Square Error (RMSE), Sum of Squares due to error (SSE) and Adjusted R-square? Diambil dari https://www.researchgate.net/post/Whats-the-acceptable-value-of-Root-Mean-Square-Error-RMSE-Sum-of-Squares-due-to-error-SSE-and-Adjusted-R-square/63cf02d06e7af9e6ee0db6d4/citation/download

Earthquake Magnitude Scale. (2023). Diambil 26 Agustus 2023, dari https://www.mtu.edu/geo/community/seismology/learn/earthquake-measure/magnitude/

Faradiba, N. (2022). Jenis-jenis Sesar Beserta Penjelasannya. Diambil 8 September 2023, dari https://www.kompas.com/sains/read/2022/02/02/113100823/jenis-jenis-sesar-beserta-penjelasannya

Firdaus, R. F. (2022). Prediksi Curah Hujan Menggunakan Metode Long Short Term Memory (Studi Kasus : Kota Bandung). *Universitas Islam Indonesia*.

Hajar, S., Novany, A. A., Windarto, A. P., Wanto, A., & Irawan, E. (2020). Penerapan K-Means Clustering Pada Ekspor Minyak Kelapa Sawit Menurut Negara Tujuan.

Hasan, R. A. (2019). NASA: Gempa Langka di Palu Bergerak Sangat Cepat, 14.760 km per jam. Diambil dari https://www.liputan6.com/global/read/3889470/nasa-gempa-langka-di-palu-bergerak-sangat-cepat-14760-km-per-jam?page=2

Husein, S. (2016). Bencana Gempabumi. https://doi.org/10.13140/RG.2.1.1112.6808

Irawan, L., Hasibuan, L. H., & Fauzi, F. (2020). Analisa Prediksi Efek Kerusakan Gempa Dari Magnitudo (Skala Richter) Dengan Metode Algoritma Id3 Menggunakan Aplikasi Data Mining Orange. *Jurnal Teknologi Informasi: Jurnal Keilmuan dan Aplikasi Bidang Teknik Informatika*, *14*(2), 189–201. https://doi.org/10.47111/jti.v14i2.1079

Ishomyl F.A, M., Waluyo, & Mustafa, L. D. (2020). Implementasi Wireless Sensor Network Pada Simulasi Peringatan Gempa Bumi Menggunakan Sensor SW-420. *Jurnal JARTEL, 10*.

Khairi, A., Awaluddin, M., & Sudarsono, B. (2020). Analisis Deformasi Seismik Sesar Matano Menggunakan Gnss Dan Interferometrik SAR.

Khumaidi, A., & Nirmala, I. A. (2022). *Algoritma Long Short Term Memory dengan Hyperparameter Tuning* (Cetakan Pertama). Yogyakarta: Deepublish.

Kusmiran, A., Minarti, Massinai, M. F. I., Zarkasi, A., Maharani, A. A., & Desiani, R. (2022). Klasifikasi Kedalaman Kejadian Gempa Menggunakan Algoritma K-Means Clustering: Studi Kasus Kejadian Gempa Di Sulawesi. *JFT: Jurnal Fisika dan Terapannya*, *9*(2), 79–88. https://doi.org/10.24252/jft.v9i2.29198

Manulang, J., santoso, A. J., & Emanuel, A. W. R. (2020). Prediksi Kunjungan Wisatawan Taman Nasional Gunung Merbabu dengan Time Series Forecasting dan LSTM. *Jurnal Buana Informatika*, *11*. Diambil dari https://e-journal.uajy.ac.id/26642/

Maramis, C., Pasau, G., & Tamuntua, G. H. (2020). Analisis Percepatan Tanah Maksimum Akibat Adanya Gempa Bumi di Lengan Utara Pulau Sulawesi Menggunakan Metode Fukushima Tanaka.

Mulyo, A. (2018). *Geologi Dasar* (Cetakan Pertama). Bandung: CV PUSTAKA SETIA.

Naufal, M. A. (2022). Simulasi Pengaruh Patahan Terkubur Terhadap Penentuan Lokasi Patahan Berdasarkan Second Vertical Derivatif Data Gravitasi. *Proceeding Seminar Nasional*.

Noor, D. (2012). *Pengantar Geologi* (Kedua). Bogor: Universitas Pakuan.

Praptosuwiryo, T. N. (2021). Botanical inventory and rarity of the fern Genus Pteris in the karst forests of Bantimurung - Bulusaraung National Park, Sulawesi – Indonesia. *IOP Conference Series: Earth and Environmental Science*, *762*(1), 012016. https://doi.org/10.1088/1755-1315/762/1/012016

Qori, P. A., Oktafani, D. S., & Kharisudin, I. (2022). Analisis Peramalan dengan Long Short Term Memory pada Data Kasus Covid-19 di Provinsi Jawa Tengah. *PRISMA 5*, *5*.

Ridla, M. A., Azise, N., & Rahman, M. (2023). Perbandingan Model Time Series Forecasting Dalam Memprediksi Jumlah Kedatangan Wisatawan Dan Penumpang Airport. *SIMKOM*, *8*(1), 1–14. https://doi.org/10.51717/simkom.v8i1.103

Sabaruddin, A. (2023). Laporan Progress Penegasan Zona Rawan Bencana Sesar Palukoro Pasca Gempa Palu 28 September 2018. *Pusat Studi Gempa Nasional*. Diambil dari https://ciptakarya.pu.go.id/admin/assets/upload/galeri/gempa/2023/02/16/2

54118_10_Laporan%20Progress%20Penegasan%20Zona%20Rawan%20Bencana%20Sesar%20Palukoro%20Pasca%20Gempa%20Palu%2028%20September%202018.pdf

Saputra, I., & Kristiyanti, D. A. (2022). *Machine Learning Untuk Pemula*. Bandung: Informatika Bandung.

Shalih, O., Adi, A. W., Wiguna, S., & Shabrina, F. Z. (2023). *RISIKO BENCANA INDONESIA: Memahami Risiko Sistemik di Indonesia* (Pertama). Jakarta: Pusat Data, Informasi, dan Komunikasi Kebencanaan BNPB.

Sofi, K., Sunge, A. S., Riady, S. R., & Kamalia, A. Z. (2021). Perbandingan Algoritma Linear Regression, Lstm, Dan Gru Dalam Memprediksi Harga Saham Dengan Model Time Series. *SEMINASTIKA*, *3*(1), 39–46. https://doi.org/10.47002/seminastika.v3i1.275

Tang, L., Ma, Y., Wang, L., Zhang, W., Zheng, L., & Wen, H. (2021). Application of Long Short-Term Memory Neural Network and Prophet Algorithm in Slope Displacement Prediction. *Application of Long Short-Term Memory Neural Network and Prophet Algorithm in Slope Displacement Prediction*, 48–66. https://doi.org/10.4417/IJGCH-06-04-04

Wati, N. P. S., & Pramartha, C. (2022). Penerapan Long Short Term Memory dalam Mengklasifikasi Jenis Ujaran Kebencian pada Tweet Bahasa Indonesia, *1*.

Widayati, T., Chatra, M. A., & Daengs, A. (2023). *Perkembangan & Transformasi Perekonomian Indonesia Abad 21 Terkini*. SONPEDIA.

Wiranda, L., & Sadikin, M. (2019). Penerapan Long Short Term Memory Pada Data Time Series Untuk Memprediksi Penjualan Produk Pt. Metiska Farma. *Jurnal Nasional Pendidikan Teknik Informatika*, *8*.

LAMPIRAN

Lampiran 1 Contoh Dataset

Datetime	*Latitude*	*Longitude*	*Depth*	*Mag*
1/2/2008	0.343	125.794	35.5	4.4
1/3/2008	-5.921	122.662	10	5.4
1/6/2008	-2.163	120.83	35	4.5
1/6/2008	1.184	125.374	35	3.9
1/8/2008	0.521	123.35	268.7	4.6
1/12/2008	-2.255	120.587	25	4.1
1/14/2008	0.189	122.032	214.7	4.3
1/14/2008	-2.483	120.843	58.3	4.2
1/17/2008	0.143	122.004	213.3	4.2
1/21/2008	0.014	123.788	144.7	4.8
1/21/2008	-1.059	120.663	48.5	4.2
1/23/2008	-2.321	120.651	34.9	4.7
1/24/2008	0.044	124.42	128.6	4.8
1/24/2008	0.499	123.599	271.9	4
1/24/2008	-0.226	120.223	30.9	4.3
1/24/2008	-0.114	124.001	107.9	5
1/27/2008	0.064	123.542	190.4	4.2
1/30/2008	-0.191	124.977	35.2	4.6
…	…	…	…	…
9/25/2023	0.245398	122.1239	172	2.782713
9/25/2023	-0.59692	121.8538	10	3.509643
9/25/2023	-2.35739	120.9949	8	2.998734
9/26/2023	-2.33573	121.2058	5	2.827855
9/26/2023	1.836048	120.6643	17	3.512597
9/26/2023	-0.02377	121.2014	13	3.200471
9/28/2023	0.076113	122.7128	77	2.771405
9/28/2023	1.086288	124.0268	16	3.289771
9/29/2023	-0.03993	124.4941	34	3.581592
9/29/2023	-0.87638	122.0569	12	3.834407
9/29/2023	-1.31625	120.4701	10	3.383128
9/29/2023	0.624118	120.3255	10	3.086072
9/29/2023	0.602672	120.359	10	3.459435
9/29/2023	-2.91173	122.3018	10	3.257329
9/30/2023	1.051974	121.7028	25	2.542568
9/30/2023	-1.05985	120.8527	5	3.735689
9/30/2023	-0.85507	122.0741	7	3.269487
9/30/2023	-0.03521	122.9579	154	2.924073
9/30/2023	-0.06102	123.4312	91	2.679644

Lampiran 2 Source Code Penelitian

Potongan Kode *Import, Input* dan *Preprocessing* Data

Import Modul

```
!pip install folium
import torch
import torch.nn as nn
import torch.optim as optim
import numpy as np
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error, mean_absolute_error,
mean_absolute_percentage_error, r2_score
from folium import plugins
from IPython.display import HTML
from IPython.display import display
# from google.colab import files
```

Input Data

```
# jupyter notebook
# Path file CSV
csv_file_path = '../Data Gempa 2008 - 2023.csv'

# Baca file CSV sebagai DataFrame
data = pd.read_csv(csv_file_path, delimiter=',', parse_dates=['datetime'])

# drive
# data = pd.read_csv('/content/drive/MyDrive/Topik Gempa/Data Gempa 2008 - 2023.csv',
delimiter=',', parse_dates=['datetime'])
data
```

Preprocessing Data

```
# Data Cleaning
print("Pengecekan Data Kosong")
print(data.isnull().sum())
data = data.dropna()
print("Setelah Data Cleaning")
print(data.isnull().sum())

# Menambahkan kondisi filter
```

```
mag_threshold = 2.5
data = data[data['mag'] >= mag_threshold]

# Tampilkan DataFrame
data

# Hitung berapa banyak data duplikat
duplicate_count_before = data.duplicated(subset=['datetime', 'latitude', 'longitude',
'depth', 'mag']).sum()

# Tampilkan jumlah data duplikat
print("Jumlah data duplikat:", duplicate_count_before)

# Hapus data duplikat
print("Penghapusan data duplikat")
data = data[~data.duplicated(subset=['datetime','latitude', 'longitude', 'depth',
'mag'], keep='first')]

# Hitung berapa banyak data duplikat
duplicate_count_after = data.duplicated(subset=['datetime', 'latitude', 'longitude',
'depth', 'mag']).sum()

# Tampilkan jumlah data duplikat
print("Jumlah data duplikat setelah dihapus:", duplicate_count_after)

# Buat dataframe baru agar tidak menghapus data depth asli
new_data = pd.DataFrame(data, columns=['datetime', 'latitude', 'longitude', 'depth',
'mag'])
new_data

# Pengelompokan Data depth
new_data.set_index('datetime', inplace=True)

# Batas-batas pengelompokan
bin_borders = [0, 30, 60, 180, 300, 525, 1000]

# Buat label untuk setiap pengelompokan
labels=[1, 2, 3, 4, 5, 6]

# Pengelompokan depth
new_data['depth'] = pd.cut(new_data['depth'], bins=bin_borders, labels=labels,
right=False)

new_data
```

Potongan Kode Klasterisasi

```
num_clusters = 6
kmeans = KMeans(n_clusters=num_clusters, random_state=42, n_init=10)
new_data['region'] = kmeans.fit_predict(new_data[['latitude', 'longitude']])
```

Potongan Kode Normalisasi Data

```
# Normalisasi
scaler = MinMaxScaler(feature_range=(0,1))
scaled_data = scaler.fit_transform(new_data[['latitude', 'longitude', 'depth',
'mag']].values)
```

Potongan Kode Pembagian Data

```
# Pembagian data latih dan uji
train_size = int(0.7 * len(scaled_data))
train_data = scaled_data[:train_size]
test_data = scaled_data[train_size:]
print(len(train_data), len(test_data))
```

Potongan Kode Pembuatan Dataset *LSTM*

```
# Fungsi untuk membuat dataset lSTM
def create_dataset(data, lookback):
    X, y = [], []
    for i in range(len(data)-lookback):
        X.append(data[i:i+lookback])
        y.append(data[i+lookback])
    X = np.array(X)
    y = np.array(y)
    return torch.tensor(X).float(), torch.tensor(y).float()

lookback_pelatihan = 10  # Jumlah langkah waktu
X_train, y_train = create_dataset(train_data, lookback_pelatihan)
X_test, y_test = create_dataset(test_data, lookback_pelatihan)
```

Potongan Kode Pembuatan Model *LSTM*

```
# Mendefinisikan Model
class GempaModel(nn.Module):
    def __init__(self, input_size, hidden_size, output_size):
        super(GempaModel, self).__init__()
        self.lstm1 = nn.LSTM(input_size, hidden_size, batch_first=True)
        self.lstm2 = nn.LSTM(hidden_size, hidden_size, batch_first=True)
        self.dropout2 = nn.Dropout(0.4)
```

```
        self.batchnorm1 = nn.BatchNorm1d(hidden_size)
        self.lstm3 = nn.LSTM(hidden_size, hidden_size, batch_first=True)
        self.lstm4 = nn.LSTM(hidden_size, hidden_size, batch_first=True)
        self.dropout4 = nn.Dropout(0.4)
        self.batchnorm2 = nn.BatchNorm1d(hidden_size)
        self.fc = nn.Linear(hidden_size, output_size)
        self.relu = nn.ReLU()

    def forward(self, x):
        out, _ = self.lstm1(x)
        out, _ = self.lstm2(out)
        out = self.dropout2(out)
        out = self.batchnorm1(out.permute(0, 2, 1)).permute(0, 2, 1)
        out, _ = self.lstm3(out)
        out, _ = self.lstm4(out)
        out = self.dropout4(out)
        out = self.batchnorm2(out.permute(0, 2, 1)).permute(0, 2, 1)
        out = self.fc(out[:, -1, :])
        out = self.relu(out)
        return out

input_size =4  # Jumlah fitur masukan (latitude, longitude, depth, mag)
hidden_size =64  # Jumlah Unit LSTM
output_size =4 # Jumlah fitur keluaran (latitude, longitude, depth, mag)

model = GempaModel(input_size, hidden_size, output_size)
```

Potongan Kode Pelatihan Model

```
# Loss function dan optimizer
criterion = nn.MSELoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)

# Jumlah epoch dan batch_size
num_epochs = 5000
batch_size = 64
iterasi = 1
layer = 4
drop_out = 4

# fungsi untuk menyimpan nama kombinasi parameter percobaan
kombinasi_parameter = lookback_pelatihan,hidden_size,num_epochs,layer,drop_out,iterasi

%%time

best_loss = float('inf')  # Inisialisasi dengan nilai yang besar
```

```
best_model_path = f'best_model{kombinasi_parameter}.pth'  # Jalur untuk menyimpan model terbaik

train_losses = []  # List untuk menyimpan training losses

early_stopping_patience = 10  # Jumlah epoch tanpa peningkatan yang akan ditoleransi
early_stopping_counter = 0  # Hitung epoch tanpa peningkatan

model.train()
for epoch in range(num_epochs):
    indices = torch.randperm(X_train.size(0))
    epoch_loss = 0.0  # Inisiasi loss untuk epoch saat ini
    for i in range(0, X_train.size(0), batch_size):
        batch_indices = indices[i:i+batch_size]
        batch_X = X_train[batch_indices]
        batch_y = y_train[batch_indices]

        optimizer.zero_grad()
        outputs = model(batch_X)
        loss = criterion(outputs, batch_y)
        loss.backward()
        optimizer.step()

        epoch_loss += loss.item()

    # Kalkulasi nilai loss rata-rata
    epoch_loss /= len(X_train) / batch_size
    train_losses.append(epoch_loss)

    if (epoch + 1) % 10 == 0:
        print(f'Epoch: {epoch+1}/{num_epochs}, Loss: {epoch_loss:.5f}')

        # Memeriksa apakah nilai loss lebih baik dari nilai loss sebelumnya
        if epoch_loss < best_loss:
            print("Saving the best model...")
            best_loss = epoch_loss
            torch.save(model.state_dict(), best_model_path)
            early_stopping_counter = 0  # Reset counter >
        else:
            early_stopping_counter += 1  # Tidak ada peningkatan

        # Early stopping check
        if early_stopping_counter >= early_stopping_patience:
            print(f'Early stopping at epoch {epoch+1}...')
            break  # Hentikan pelatihan
```

```
print(f"Training complete")
print(f'Best model{kombinasi_parameter} saved')
```

Potongan Kode Hasil Pelatihan

```
# Membuat DataFrame untuk nilai loss
train_loss_df = pd.DataFrame({'Epoch': range(1, len(train_losses)+1), 'Loss':
train_losses})

train_loss_df

# Ploting nilai loss
plt.plot(range(1, len(train_losses)+1), train_losses, label='Train Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title(f'Training Loss Curve {kombinasi_parameter}')
plt.legend()

# Menyimpan gambar ke dalam file
plt.savefig(f'train_loss_curve_{kombinasi_parameter}.png', bbox_inches='tight')
print("Gambar telah disimpan.")

plt.show()
```

Potongan Kode Evaluasi Model

```
%%time
# Evaluasi
with torch.no_grad():
    model.eval()
    train_preds = model(X_train)
    test_preds = model(X_test)

# Inverse scaling of predictions
train_preds = scaler.inverse_transform(train_preds.numpy())
test_preds = scaler.inverse_transform(test_preds.numpy())

# Convert the true values to their original scale
y_train = scaler.inverse_transform(y_train.numpy())
y_test = scaler.inverse_transform(y_test.numpy())
```

Potongan Kode *Visualisasi* Hasil Prakiraan

Pembuatan Dataframe aktual dan hasil prakiraan

```
# Dataframe baru data aktual
aktual_train = pd.DataFrame(y_train, columns=['latitude', 'longitude', 'depth', 'mag'],
index=new_data.index[:len(y_train)])
aktual_test = pd.DataFrame(y_test, columns=['latitude', 'longitude', 'depth', 'mag'],
index=new_data.index[-len(y_test):])
aktual = pd.concat([aktual_train, aktual_test])
aktual

# Dataframe baru hasil prediksi
prediksi_train = pd.DataFrame(train_preds, columns=['latitude', 'longitude', 'depth',
'mag'], index=new_data.index[:len(train_preds)])
prediksi_test = pd.DataFrame(test_preds, columns=['latitude', 'longitude', 'depth',
'mag'], index=new_data.index[-len(test_preds):])

# Membulatkan nilai hasil prediksi
prediksi_train['depth'] = prediksi_train['depth'].round().astype(int).clip(0, 6)
prediksi_test['depth'] = prediksi_test['depth'].round().astype(int).clip(0, 6)
```

Visualisasi Hasil

Latitude

```
# Perbandingan data aktual dan prediksi
plt.figure(figsize=(50, 10))
plt.plot(aktual.index, aktual['latitude'], label='Original')
plt.plot(prediksi_train.index, prediksi_train['latitude'], label='Train Predictions')
plt.plot(prediksi_test.index, prediksi_test['latitude'], label='Test Predictions')
plt.xlabel('Datetime')
plt.ylabel('Latitude')
plt.title('Perbandingan Data Original dan Prediksi (Latitude)')
plt.legend()

# Menyimpan gambar ke dalam file
plt.savefig(f'latitude{kombinasi_parameter}.png', bbox_inches='tight')
print("Gambar telah disimpan.")

plt.show()
```

Longitude

```
# Perbandingan data aktual dan prediksi
plt.figure(figsize=(50, 10))
plt.plot(aktual.index, aktual['longitude'], label='Original')
```

```
plt.plot(prediksi_train.index, prediksi_train['longitude'], label='Train Predictions')
plt.plot(prediksi_test.index, prediksi_test['longitude'], label='Test Predictions')
plt.xlabel('Datetime')
plt.ylabel('Longitude')
plt.title('Perbandingan Data Original dan Prediksi (Longitude)')
plt.legend()

# Menyimpan gambar ke dalam file
plt.savefig(f'longitude{kombinasi_parameter}.png', bbox_inches='tight')
print("Gambar telah disimpan.")

plt.show()
```

Depth

```
# Perbandingan data aktual dan prediksi
plt.figure(figsize=(50, 10))
plt.plot(aktual.index, aktual['depth'], label='Original')
plt.plot(prediksi_train.index, prediksi_train['depth'], label='Train Predictions')
plt.plot(prediksi_test.index, prediksi_test['depth'], label='Test Predictions')
plt.xlabel('Datetime')
plt.ylabel('Depth')
plt.title('Perbandingan Data Original dan Prediksi (Depth)')
plt.legend()

# Menyimpan gambar ke dalam file
plt.savefig(f'depth{kombinasi_parameter}.png', bbox_inches='tight')
print("Gambar telah disimpan.")

plt.show()
```

Mag

```
# Perbandingan data aktual dan prediksi
plt.figure(figsize=(50, 10))
plt.plot(aktual.index, aktual['mag'], label='Original')
plt.plot(prediksi_train.index, prediksi_train['mag'], label='Train Predictions')
plt.plot(prediksi_test.index, prediksi_test['mag'], label='Test Predictions')
plt.xlabel('Datetime')
plt.ylabel('Mag')
plt.title('Perbandingan Data Original dan Prediksi (Mag)')
plt.legend()

# Menyimpan gambar ke dalam file
plt.savefig(f'mag{kombinasi_parameter}.png', bbox_inches='tight')
```

```
print("Gambar telah disimpan.")

plt.show()
```

Plotting Pada Peta

Data Aktual

```
import folium

# Inisialisasi peta
center_latitude = aktual_test['latitude'].iloc[0]  # Mengambil nilai pertama dari kolom 'latitude'
center_longitude = aktual_test['longitude'].iloc[0]  # Mengambil nilai pertama dari kolom 'longitude'

# Inisialisasi peta dengan lokasi pusat yang sudah ditentukan
aktual_test_ploting = folium.Map(location=[center_latitude, center_longitude], zoom_start=5)

# Loop melalui setiap baris dalam aktual_test
for index, row in aktual_test.iterrows():
    latitude = row['latitude']
    longitude = row['longitude']
    depth = row['depth']
    mag = row['mag']
    date = index.strftime('%Y-%m-%d')  # Konversi tanggal ke format string

    color = '#FFFF00'  # Default color
    radius = 3

    if 2.5 <= mag <= 5.4:
        color = '#FFFF00'
        radius = 3
    elif 5.5 <= mag <= 6.0:
        color = '#FFD200'
        radius = 4
    elif 6.1 <= mag <= 6.9:
        color = '#FFA500'
        radius = 5
    elif 7.0 <= mag <= 7.9:
        color = '#FF5300'
        radius = 6
    elif 8.0 <= mag <= 9.9:
        color = '#FF0000'
        radius = 7
```

```
    # Bua marker berbentuk titik (CircleMarker) pada peta dengan warna yang sesuai
    folium.CircleMarker(
        location=[latitude, longitude],
        radius=radius,  # Ukuran radius titik
        color=color,  # Warna titik
        fill=True,
        fill_color=color,  # Warna isi titik
        fill_opacity=1,
        popup=f"Date: {date}<br> Latitude: {latitude}<br> Longitude:
{longitude}<br>Depth: {depth}<br>Mag: {mag}"
    ).add_to(aktual_test_ploting)

# Tampilkan peta
aktual_test_ploting
```

Data Prakiraan

```
import folium

center_latitude2 = prediksi_test['latitude'].iloc[0]  # Mengambil nilai pertama dari
kolom 'latitude'
center_longitude2 = prediksi_test['longitude'].iloc[0]  # Mengambil nilai pertama dari
kolom 'longitude'

# Inisialisasi peta dengan lokasi pusat yang sudah ditentukan
prediksi_test_ploting = folium.Map(location=[center_latitude2, center_longitude2],
zoom_start=5)

# Loop melalui setiap baris dalam aktual_test
for index, row in prediksi_test.iterrows():
    latitude = row['latitude']
    longitude = row['longitude']
    depth = row['depth']
    mag = row['mag']
    date = index.strftime('%Y-%m-%d')  # Konversi tanggal ke format string

    color = '#FFFF00'  # Default color
    radius = 3

    if 2.5 <= mag <= 5.4:
        color = '#FFFF00'
        radius = 3
    elif 5.5 <= mag <= 6.0:
        color = '#FFD200'
```

```
        radius = 4
    elif 6.1 <= mag <= 6.9:
        color = '#FFA500'
        radius = 5
    elif 7.0 <= mag <= 7.9:
        color = '#FF5300'
        radius = 6
    elif 8.0 <= mag <= 9.9:
        color = '#FF0000'
        radius = 7

    # Buat marker berbentuk titik (CircleMarker) pada peta dengan warna yang sesuai
    folium.CircleMarker(
        location=[latitude, longitude],
        radius=radius,  # Ukuran radius titik
        color=color,  # Warna titik
        fill=True,
        fill_color=color,  # Warna isi titik
        fill_opacity=1,
        popup=f"Date: {date}<br> Latitude: {latitude}<br> Longitude:
{longitude}<br>Depth: {depth}<br>Mag: {mag}"
    ).add_to(prediksi_test_ploting)

# Tampilkan peta
prediksi_test_ploting
```

Potongan Kode Evaluasi Matriks

```
eval_metrics_train = {
    'latitude': {'mse': None, 'rmse': None, 'mae': None},
    'longitude': {'mse': None, 'rmse': None, 'mae': None},
    'depth': {'mse': None, 'rmse': None, 'mae': None},
    'mag': {'mse': None, 'rmse': None, 'mae': None}
}

eval_metrics_test = {
    'latitude': {'mse': None, 'rmse': None, 'mae': None},
    'longitude': {'mse': None, 'rmse': None, 'mae': None},
    'depth': {'mse': None, 'rmse': None, 'mae': None},
    'mag': {'mse': None, 'rmse': None, 'mae': None}
}

for col in ['latitude', 'longitude', 'depth', 'mag']:
    train_col_idx = list(new_data.columns).index(col)
```

```
    test_col_idx = list(new_data.columns).index(col)

    # Evaluate on train predictions
    train_mse = mean_squared_error(y_train[:, train_col_idx], train_preds[:,
train_col_idx])
    train_rmse = np.sqrt(train_mse)
    train_mae = mean_absolute_error(y_train[:, train_col_idx], train_preds[:,
train_col_idx])

    eval_metrics_train[col] = {'mse': train_mse, 'rmse': train_rmse, 'mae': train_mae}

    # Evaluate on test predictions
    test_mse = mean_squared_error(y_test[:, test_col_idx], test_preds[:, test_col_idx])
    test_rmse = np.sqrt(test_mse)
    test_mae = mean_absolute_error(y_test[:, test_col_idx], test_preds[:,
test_col_idx])

    eval_metrics_test[col] = {'mse': test_mse, 'rmse': test_rmse, 'mae': test_mae}

# Buat DataFrames dari metrik evaluasi
train_metrics_df = pd.DataFrame(eval_metrics_train)
test_metrics_df = pd.DataFrame(eval_metrics_test)

# Calculate other evaluation metrics
train_mse_all = mean_squared_error(y_train, train_preds)
train_rmse_all = np.sqrt(train_mse_all)
train_mae_all = mean_absolute_error(y_train, train_preds)

test_mse_all = mean_squared_error(y_test, test_preds)
test_rmse_all = np.sqrt(test_mse_all)
test_mae_all = mean_absolute_error(y_test, test_preds)

# Create a DataFrame for evaluation metrics
evaluation_metrics_df = pd.DataFrame({
    'Metric': ['MSE', 'RMSE', 'MAE',],
    'Train': [train_mse_all, train_rmse_all, train_mae_all],
    'Test': [test_mse_all, test_rmse_all, test_mae_all]
})

# Inisialisasi DataFrame untuk semua evaluasi
all_metrics_df = pd.DataFrame(columns=['Column', 'Train MSE', 'Train RMSE', 'Train
MAE', 'Test MSE', 'Test RMSE', 'Test MAE'])

# List untuk menyimpan DataFrame sementara
dfs = []
```

```
# Loop melalui kolom-kolom
for col in eval_metrics_train:
    # Ambil metrik evaluasi dari dictionary
    train_mse = eval_metrics_train[col]['mse']
    train_rmse = eval_metrics_train[col]['rmse']
    train_mae = eval_metrics_train[col]['mae']

    test_mse = eval_metrics_test[col]['mse']
    test_rmse = eval_metrics_test[col]['rmse']
    test_mae = eval_metrics_test[col]['mae']

    # Tambahkan DataFrame sementara ke list
    df_temp = pd.DataFrame({
        'Column': col,
        'Train MSE': train_mse,
        'Train RMSE': train_rmse,
        'Train MAE': train_mae,
        'Test MSE': test_mse,
        'Test RMSE': test_rmse,
        'Test MAE': test_mae
    }, index=[0])

    dfs.append(df_temp)

# Gabungkan DataFrames menggunakan pandas.concat
all_metrics_df = pd.concat(dfs, ignore_index=True)

# Menambahkan hasil evaluasi keseluruhan ke DataFrame
all_metrics_df.loc[len(all_metrics_df)] = ['all', train_mse_all, train_rmse_all,
train_mae_all, test_mse_all, test_rmse_all, test_mae_all]

all_metrics_df.to_csv(f'all_metrics_df{kombinasi_parameter}.csv', index=True)

# Tampilkan DataFrame dengan semua metrik evaluasi termasuk hasil evaluasi keseluruhan
print(f"All Evaluation Metrics:{kombinasi_parameter}")
display(all_metrics_df)
```

Perbandingan Data Aktual dan Prakiraan

```
prediksi = pd.concat([prediksi_train, prediksi_test]).add_suffix('_predict')

# Menyimpan DataFrame ke dalam file CSV
prediksi.to_csv(f'prediksi{kombinasi_parameter}.csv', index=True)
prediksi

# Mengatur ulang indeks data asli dan prediksi
```

```
aktual_rest = aktual.reset_index(drop=True)
prediksi_reset = prediksi.reset_index(drop=True)

# Mengambil kolom datetime dari dataframe data
datetime = aktual.index

# Menggabungkan data asli dan prediksi
df_comparison = pd.concat([aktual_rest, prediksi_reset], axis=1)
df_comparison.columns = ['latitude', 'longitude', 'depth', 'mag', 'latitude_predict',
'longitude_predict', 'depth_predict', 'mag_predict']

# Memasukkan kolom datetime ke dalam df_comparison
df_comparison.insert(0, 'datetime', datetime)

# Menampilkan DataFrame hasil perbandingan
df_comparison

# Menyimpan DataFrame ke dalam file CSV
df_comparison.to_csv(f'df_comparison{kombinasi_parameter}.csv', index=True)
```

Potongan Kode *Blind Prediction*

```
model.eval()

lookback_prediksi = len(scaled_data)

# Inisialisasi data input awal
initial_data = scaled_data[-lookback_prediksi:]

# Inisialisasi data untuk hasil prediksi
predicted_data = []

# Iterasi
for day in range(844):
    # Prediksi satu langkah ke depan
    X_pred = initial_data[-lookback_prediksi:]
    X_pred = np.expand_dims(X_pred, axis=0)

    with torch.no_grad():
        pred = model(torch.tensor(X_pred).float())
        pred = pred.numpy()

    pred = scaler.inverse_transform(pred)

    # Tambahkan hasil prediksi (latitude, longitude, depth, dan mag)
    predicted_data.append([ *pred.squeeze()])
```

```
    # Tambahkan hasil prediksi ke data input
    initial_data = np.concatenate([initial_data, pred], axis=0)

# Konversi hasil prediksi menjadi DataFrame
columns = ['latitude', 'longitude', 'depth', 'mag']
prediksi_pred = pd.DataFrame(predicted_data, columns=columns)

prediksi_pred['depth'] = prediksi_pred['depth'].round().astype(int).clip(0, 6)
```

Potongan Kode *Visualisasi* Hasil Prakiraan

Latitude

```
# Ambil 500 data terakhir dari DataFrame new_data
last_data_longitude = new_data['latitude'][-500:]

# Plot
plt.figure(figsize=(30, 8))

# Plot data aktual
plt.plot(range(len(last_data_longitude)), last_data_longitude, label='Aktual')

# Plot hasil prediksi, dimulai dari akhir data aktual
plt.plot(range(len(last_data_longitude), len(last_data_longitude) +
len(prediksi_pred)), prediksi_pred['latitude'], label='Prakiraan')

# Tambahkan garis putus-putus merah untuk menandai batas
plt.axvline(x=len(last_data_longitude), color='red', linestyle='--')

plt.xlabel('Data Index')
plt.ylabel('Longitude')
plt.title('Perbandingan Nilai Aktual dan Hasil Prediksi (Longitude)')
plt.legend()

# # Menyimpan gambar ke dalam file
plt.savefig(f'latitude_bllind_predict{kombinasi_parameter}.png', bbox_inches='tight')
# print("Gambar telah disimpan.")

plt.show()
```

Longitude

```
# Ambil 500 data terakhir dari DataFrame new_data
```

```
last_data_longitude = new_data['longitude'][-500:]

# Plot
plt.figure(figsize=(30, 8))

# Plot data aktual
plt.plot(range(len(last_data_longitude)), last_data_longitude, label='Aktual')

# Plot hasil prediksi, dimulai dari akhir data aktual
plt.plot(range(len(last_data_longitude), len(last_data_longitude) +
len(prediksi_pred)), prediksi_pred['longitude'], label='Prakiraan')

# Tambahkan garis putus-putus merah untuk menandai batas
plt.axvline(x=len(last_data_longitude), color='red', linestyle='--')

plt.xlabel('Data Index')
plt.ylabel('Longitude')
plt.title('Perbandingan Nilai Aktual dan Hasil Prediksi (Longitude)')
plt.legend()

# # Menyimpan gambar ke dalam file
plt.savefig(f'longitude_bllind_predict{kombinasi_parameter}.png', bbox_inches='tight')
# print("Gambar telah disimpan.")

plt.show()
```

Depth

```
# Ambil 500 data terakhir dari DataFrame new_data
last_data_depth = new_data['depth'][-500:]

# Plotting the predictions
plt.figure(figsize=(30, 8))

# Plot data aktual
plt.plot(range(len(last_data_depth)), last_data_depth, label='Aktual')

# Plot hasil prediksi, dimulai dari akhir data aktual
plt.plot(range(len(last_data_depth), len(last_data_depth) + len(prediksi_pred)),
prediksi_pred['depth'], label='Prakiraan')

# Tambahkan garis putus-putus merah untuk menandai batas
plt.axvline(x=len(last_data_depth), color='red', linestyle='--')

plt.xlabel('Data Index')
```

```
plt.ylabel('Depth')
plt.title('Perbandingan Nilai Aktual dan Hasil Prediksi (Depth)')
plt.legend()

# # Menyimpan gambar ke dalam file
plt.savefig(f'depth_bllind_predict{kombinasi_parameter}.png', bbox_inches='tight')
# print("Gambar telah disimpan.")

plt.show()
```

Mag

```
# Ambil 500 data terakhir dari DataFrame new_data
last_data_mag = new_data['mag'][-500:]

# Plot
plt.figure(figsize=(30, 8))

# Plot data aktual
plt.plot(range(len(last_data_mag)), last_data_mag, label='Aktual')

# Plot hasil prediksi, dimulai dari akhir data aktual
plt.plot(range(len(last_data_mag), len(last_data_mag) + len(prediksi_pred)),
prediksi_pred['mag'], label='Prakiraan')

# Tambahkan garis putus-putus merah untuk menandai batas
plt.axvline(x=len(last_data_mag), color='red', linestyle='--')

plt.xlabel('Data Index')
plt.ylabel('Mag')
plt.title('Perbandingan Nilai Aktual dan Hasil Prediksi (Mag)')
plt.legend()

# Menyimpan gambar ke dalam file
plt.savefig(f'mag_bllind_predict{kombinasi_parameter}.png', bbox_inches='tight')
# print("Gambar telah disimpan.")

plt.show()
```

Ploting Peta

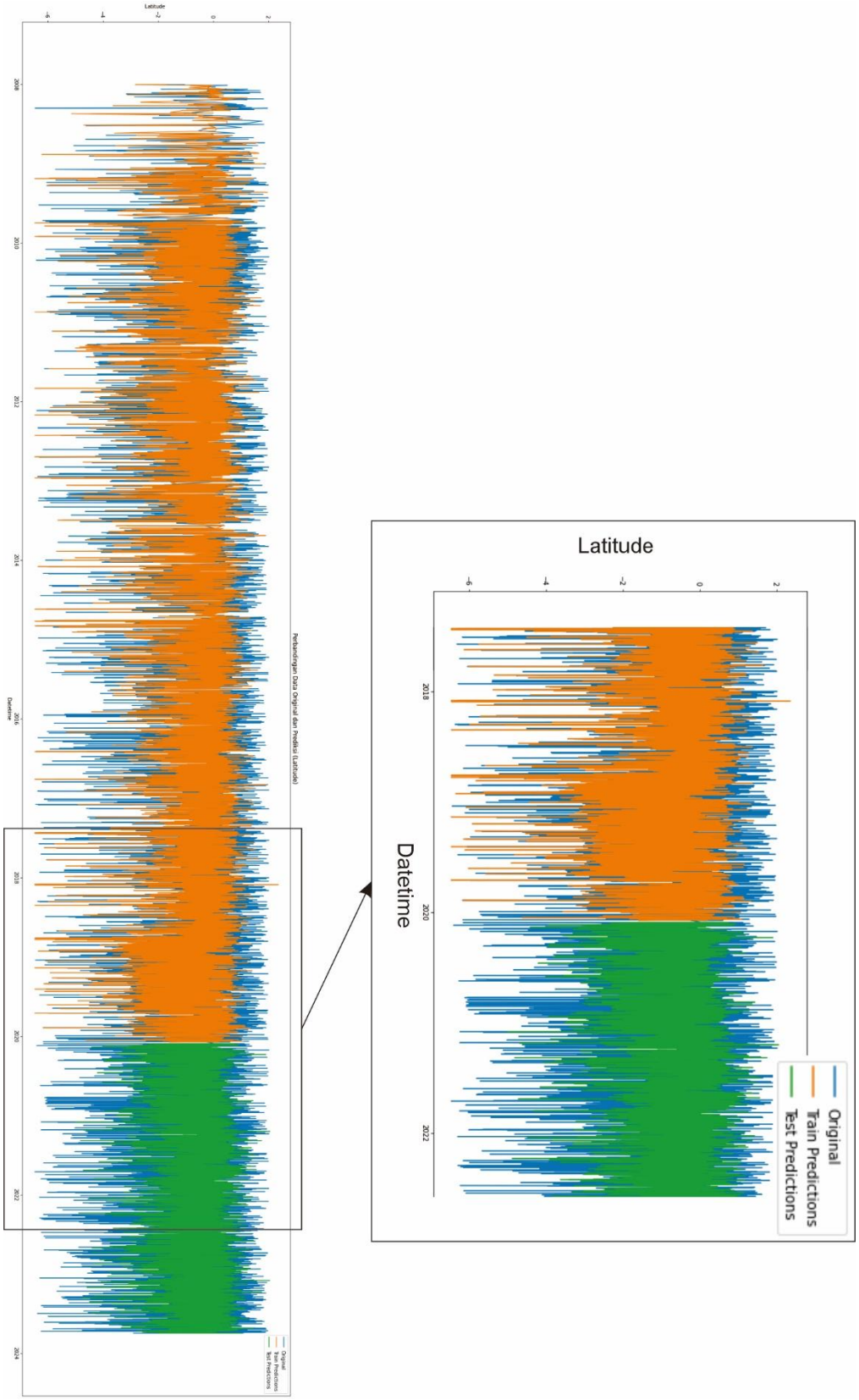
```
import folium
```

```
center_latitude1 = prediksi_pred['latitude'].iloc[0]  # Mengambil nilai pertama dari
kolom 'latitude'
center_longitude1 = prediksi_pred['longitude'].iloc[0]  # Mengambil nilai pertama dari
kolom 'longitude'

# Inisialisasi peta
blind_prediction = folium.Map(location=[center_latitude1, center_longitude1],
zoom_start=5)

# Loop melalui setiap baris dalam prediksi_pred
for index, row in prediksi_pred.iterrows():
    latitude = row['latitude']
    longitude = row['longitude']
    depth = row['depth']
    mag = row['mag']

    color = '#FFFF00'  # Default color
    radius = 3

    if 2.5 <= mag <= 5.4:
        color = '#FFFF00'
        radius = 3
    elif 5.5 <= mag <= 6.0:
        color = '#FFD200'
        radius = 4
    elif 6.1 <= mag <= 6.9:
        color = '#FFA500'
        radius = 5
    elif 7.0 <= mag <= 7.9:
        color = '#FF5300'
        radius = 6
    elif 8.0 <= mag <= 9.9:
        color = '#FF0000'
        radius = 7

    # Buat marker berbentuk titik (CircleMarker) pada peta dengan warna yang sesuai
    folium.CircleMarker(
        location=[latitude, longitude],
        radius=3,  # Ukuran radius titik
        color=color,  # Warna titik
        fill=True,
        fill_color=color,  # Warna isi titik
        popup=f"Latitude: {latitude}<br> Longitude: {longitude}<br>Depth:
{depth}<br>Mag: {mag}"
    ).add_to(blind_prediction)
```

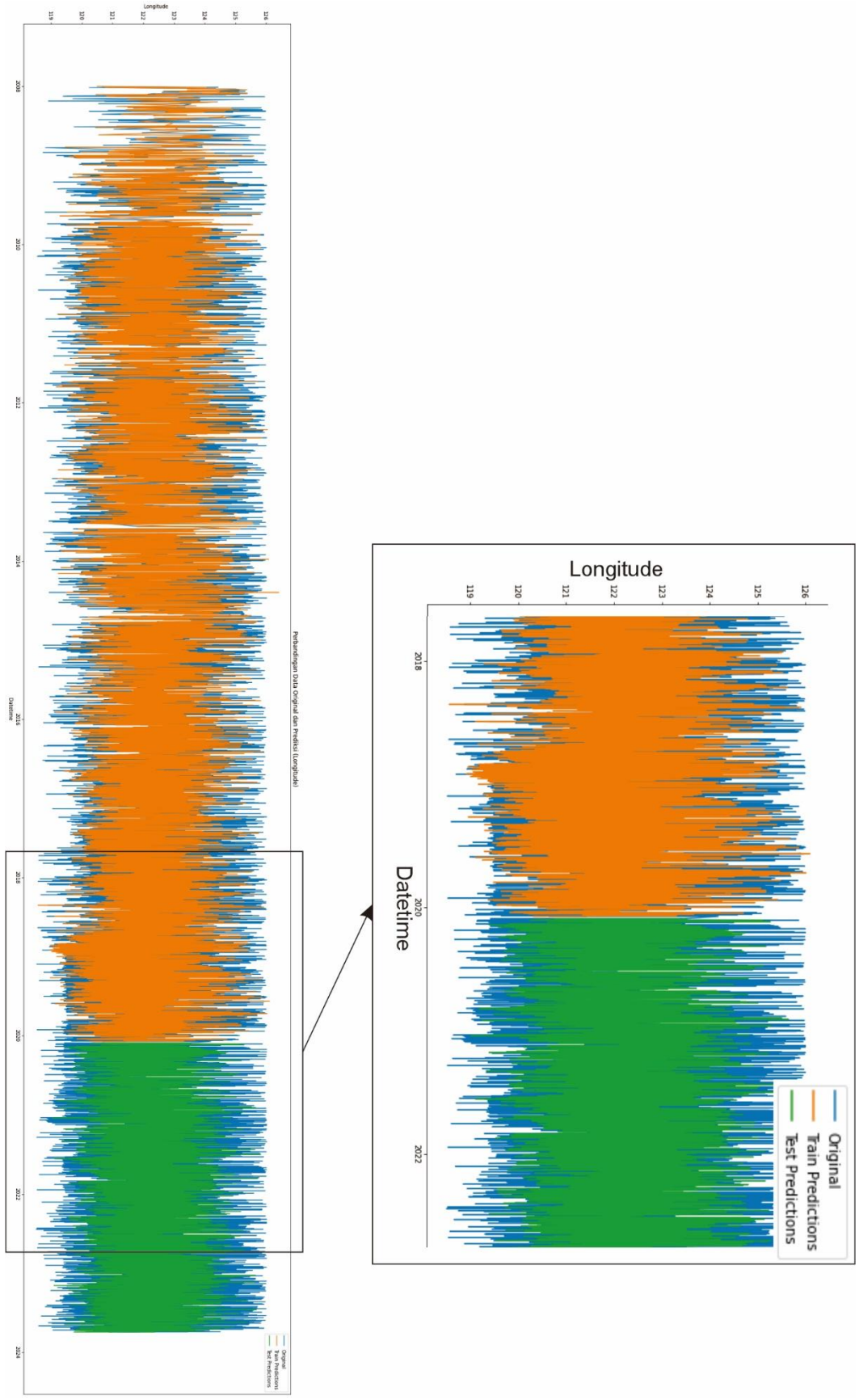
```
# Tampilkan peta
blind_prediction
```

Lampiran 3 Visualisasi Output Pelatihan

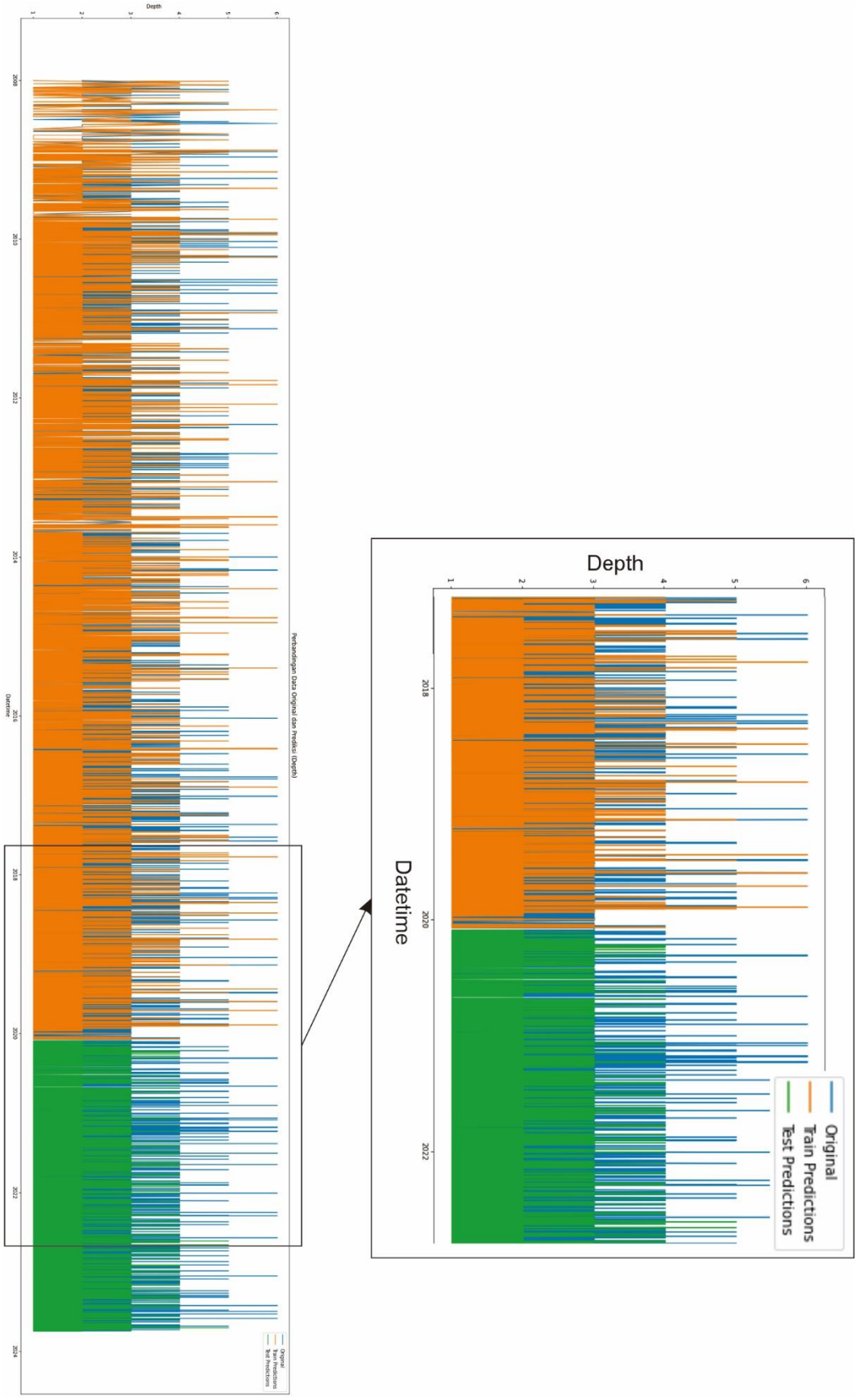
Latitude



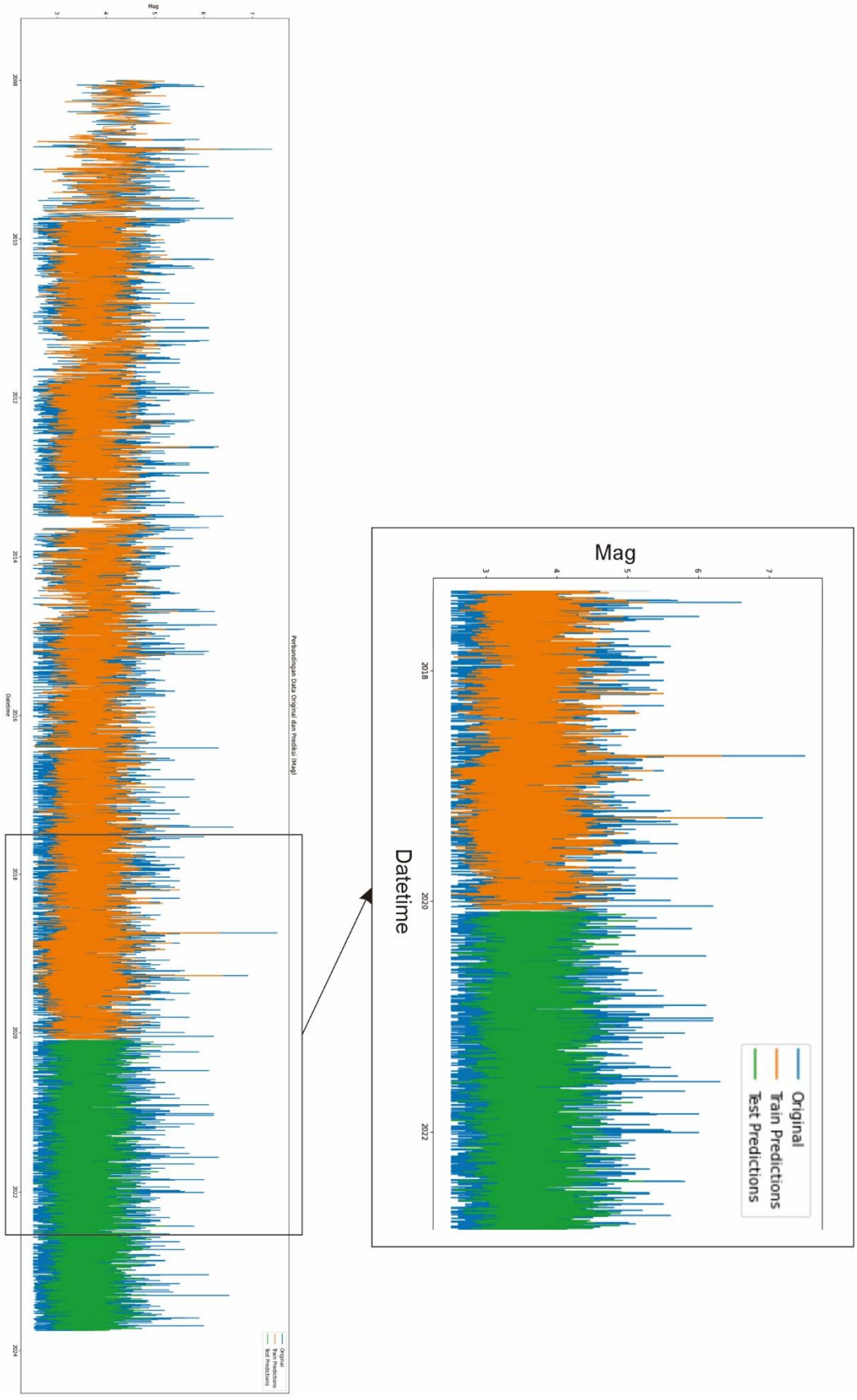
Longitude


Longitude
Datetime
Original
Train Predictions
Test Predictions
Perbandingan Data Original dan Prediksi (Longitude)

Depth

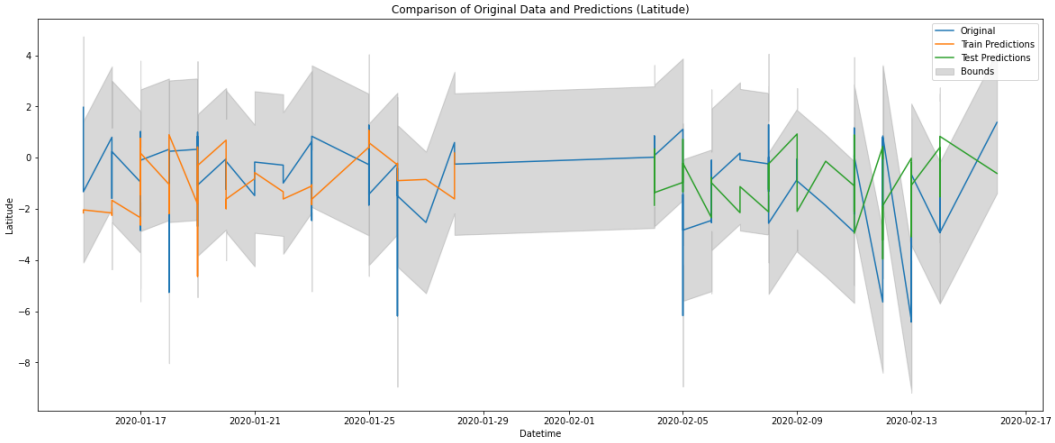

Depth
Datetime
Original
Train Predictions
Test Predictions

Mag

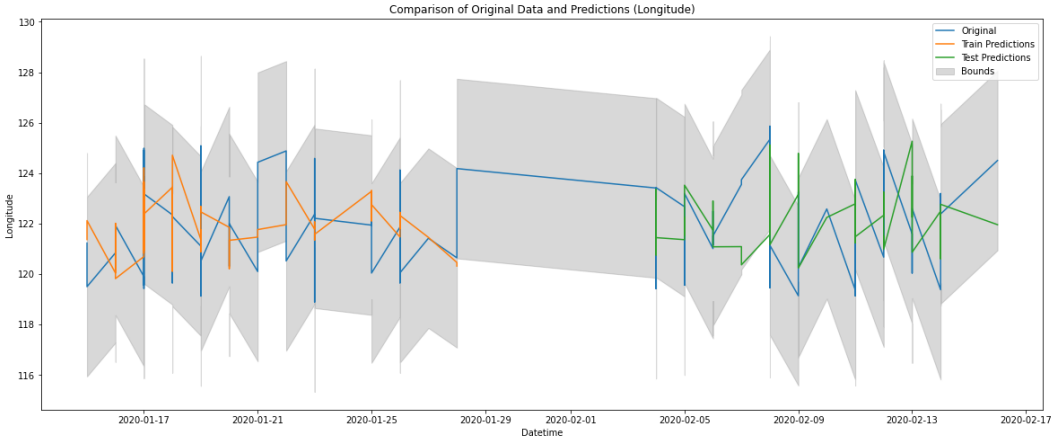

Mag
Datetime
Original
Train Predictions
Test Predictions
Perbandingan Data Original dan Prediksi (Mag)

Potongan Grafik

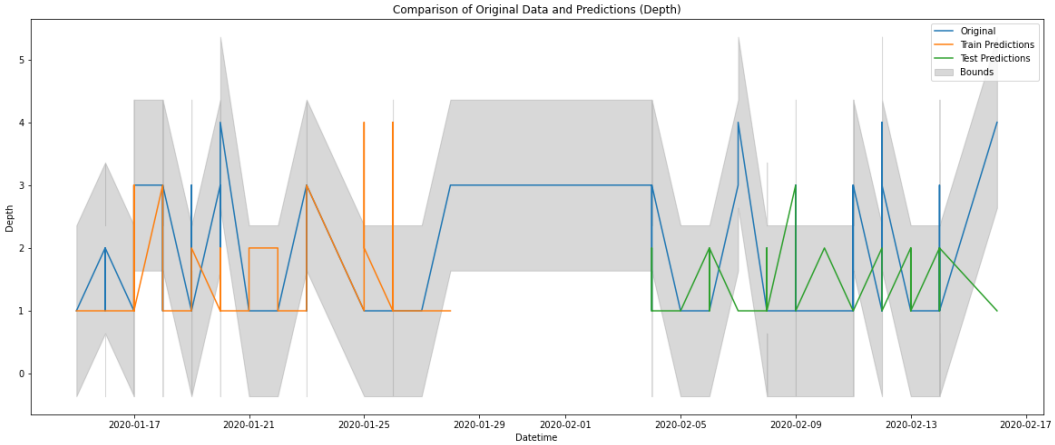
Latitude



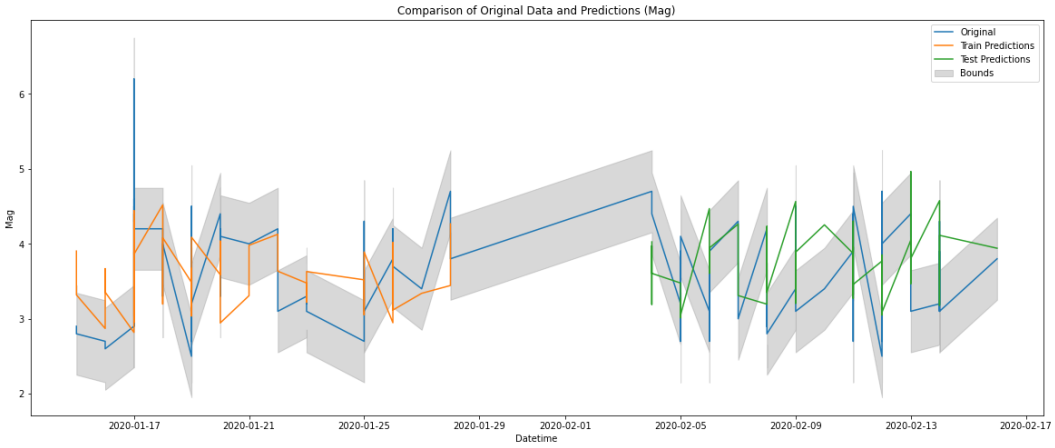
Longitude



Depth

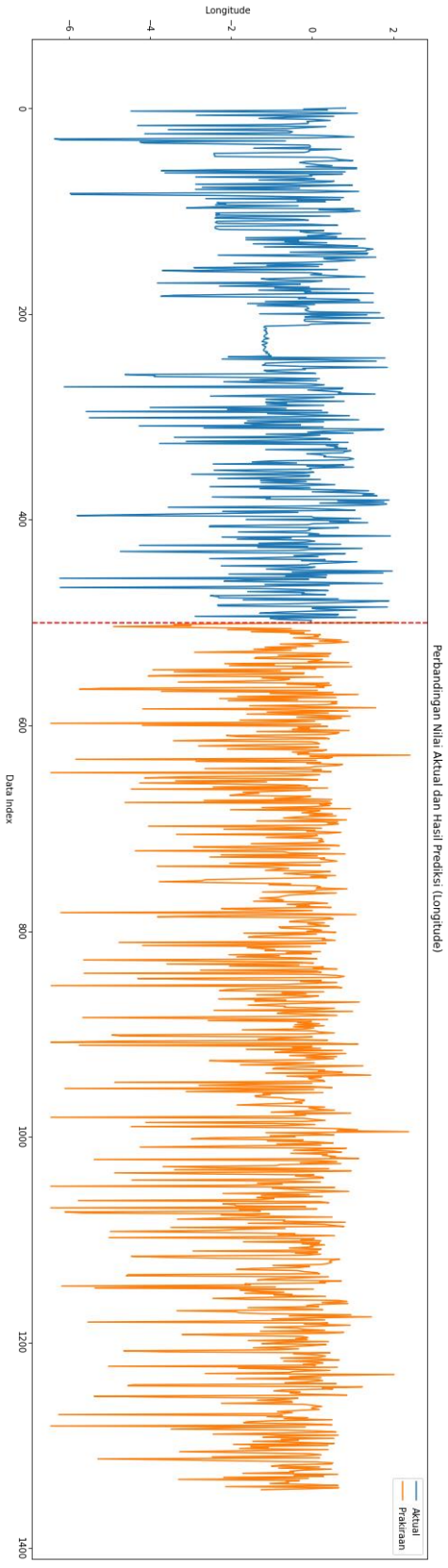


Mag

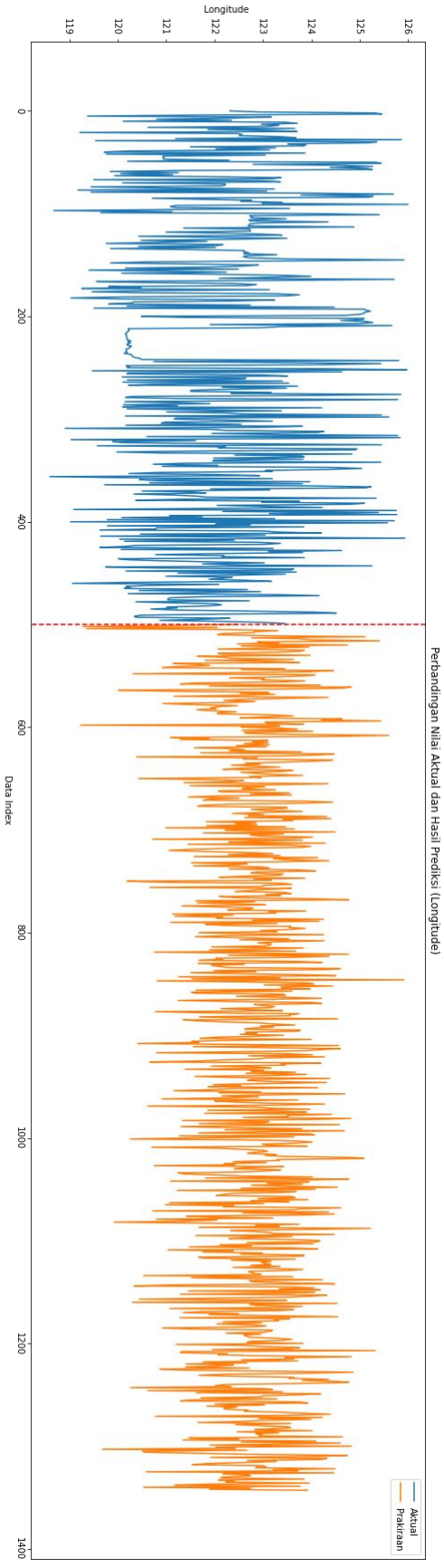

Comparison of Original Data and Predictions (Mag)
Original
Train Predictions
Test Predictions
Bounds
Mag
Datetime
2020-01-17
2020-01-21
2020-01-25
2020-01-29
2020-02-01
2020-02-05
2020-02-09
2020-02-13
2020-02-17

Lampiran 4 Visualisasi *Output* Hasil Prakiraan

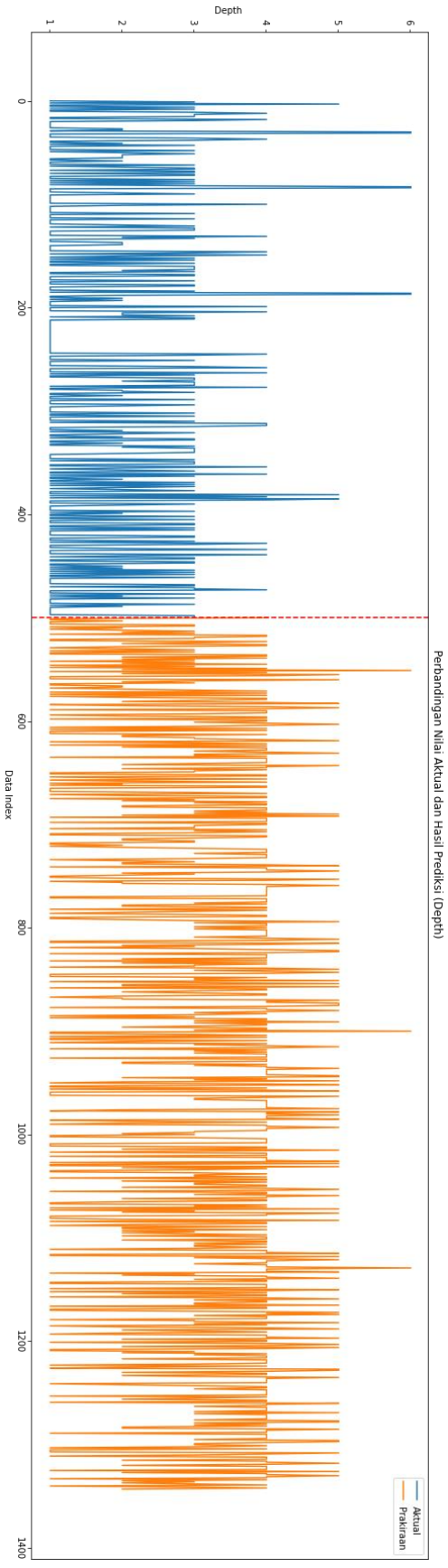
Latitude



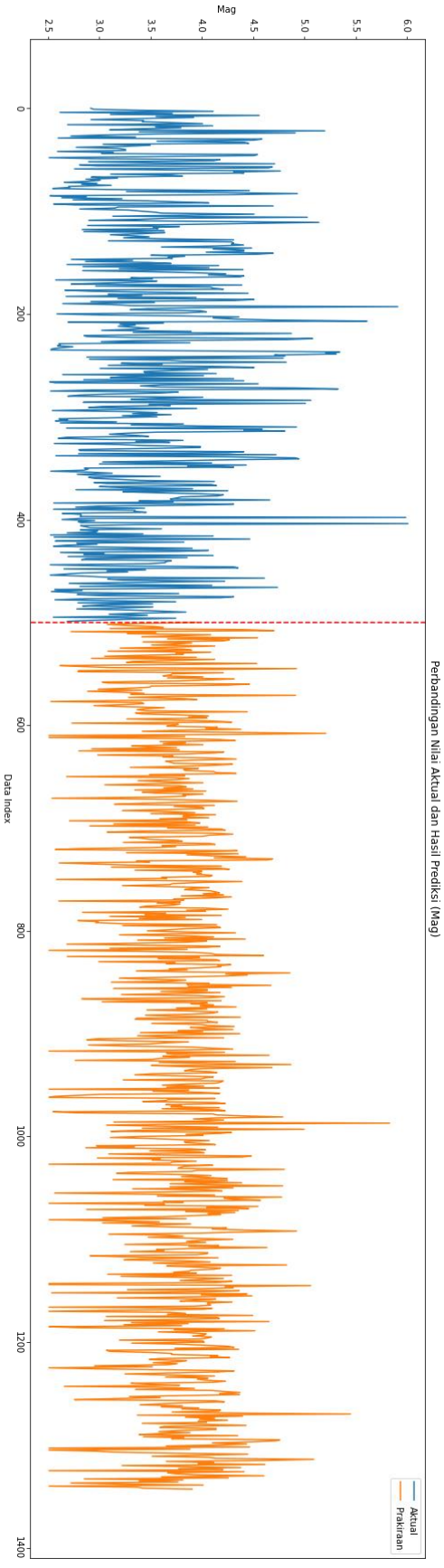
Longitude


Longitude
119
120
121
122
123
124
125
126
0
200
400
600
800
1000
1200
1400
Data Index
Perbandingan Nilai Aktual dan Hasil Prediksi (Longitude)
Aktual
Prakiraan

Depth


Depth
Data Index
Perbandingan Nilai Aktual dan Hasil Prediksi (Depth)
Aktual
Prakiraan

Mag


Mag
2.5
3.0
3.5
4.0
4.5
5.0
5.5
6.0
0
200
400
600
800
1000
1200
1400
Data Index
Perbandingan Nilai Aktual dan Hasil Prediksi (Mag)
Aktual
Prakiraan

Peta Spasial