KERJA PRAKTIK - EF234603

IMPLEMENTASI LSTM DAN ARIMA SEBAGAI MODEL FORECAST DI PT PERTAMINA EP CEPU FIELD JTB

Jalan Raya Gayam-Ngasem, Dusun Pohwayang, Desa Bandungrejo, Kecamatan Ngasem, Kabupaten Bojonegoro, Jawa Timur.
Periode: September – November 2024

Oleh:

Thomas Juan Mahardika Suryono 5025211016

Pembimbing Jurusan
Agus Budi Raharjo, S.Kom, M.Kom., Ph.D.

Pembimbing Jurusan
Ir. Febrita Kusuma Wardana, S.T., M.MT, IPM

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2024

KERJA PRAKTIK – EF234603

IMPLEMENTASI LSTM DAN ARIMA SEBAGAI MODEL FORECAST DI PT PERTAMINA EP CEPU FIELD JTB

Jalan Raya Gayam-Ngasem, Dusun Pohwayang, Desa Bandungrejo, Kecamatan Ngasem, Kabupaten Bojonegoro, Jawa Timur.
Periode: September – November 2024

Oleh:

Thomas Juan Mahardika Suryono 5025211016

Pembimbing Jurusan
Agus Budi Raharjo, S.Kom, M.Kom., Ph.D.

Pembimbing Lapangan
Ir. Febrita Kusuma Wardana, S.T., M.MT, IPM

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2024

DAFTAR ISI

[Halaman ini sengaja dikosongkan]

DAFTAR GAMBAR

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

DAFTAR KODE SUMBER

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

KERJA PRAKTIK

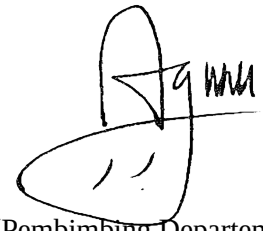
Implementasi LSTM dan ARIMA Sebagai Model Forecast di PT Pertamina EP Cepu Field JTB

Oleh:

Thomas Juan Mahardika Suryono 5025211016

Disetujui oleh Pembimbing Kerja Praktik:

1. Agus Budi Raharjo, S.Kom, M.Kom., Ph.D.
   NIP. 1990202011022



(Pembimbing Departemen)

2. Ir. Febrita Kusuma Wardana, S.T., M.MT, IPM
   Superintendent Facility and Maintenance

(Pembimbing Lapangan)

[Halaman ini sengaja dikosongkan]

Implementasi LSTM dan ARIMA Sebagai Model Forecast di PT Pertamina EP Cepu Field JTB

Nama Mahasiswa	: Thomas Juan Mahardika Suryono
NRP	: 5025211016
Departemen	: Teknik Informatika FTEIC-ITS
Pembimbing Departemen	: Agus Budi Raharjo, S.Kom, M.Kom., Ph.D.
Pembimbing Lapangan	: Ir. Febrita Kusuma Wardana, S.T., M.MT, IPM

ABSTRAK

Kerja praktik ini dilakukan di PT Pertamina EP Cepu Field JTB dengan tujuan untuk mengimplementasikan LSTM dan ARIMA sebagai model forecast. Permasalahan yang dialami perusahaan adalah belum adanya model forecast yang dapat digunakan untuk memforecast berbagai kebutuhan pembersihan mesin. Mesin yang digunakan perusahaan memiliki komponen filter yang seiring berjalannya waktu, filter semakin kotor. Metodologi yang digunakan adalah CRISP DM yang terdiri dari business understanding, data understanding, data preparation, modeling, evaluation, deployment. Setelah evaluasi, didapatkan hasil akurasi pada variabel IN DP, MON DP, MON DP A, dan MON DP C dengan MAPE di bawah 11% dan R^2 di atas 0.95. Dengan forecasting, perusahaan mampu memprediksi kapan filter diganti, strainer dibersihkan, dan offline wash gas turbin.

Kata Kunci : Autoregressive Integrated Moving Average, Forecast, Long Short Term Memory

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Allah SWT atas penyertaan dan karunia-Nya sehingga penulis dapat menyelesaikan salah satu kewajiban penulis sebagai mahasiswa Departemen Teknik Informatika ITS yaitu Kerja Praktik yang berjudul: **Implementasi LSTM dan ARIMA Sebagai Model Forecast di PT Pertamina EP Cepu Field JTB**.

Penulis menyadari bahwa masih banyak kekurangan baik dalam melaksanakan kerja praktik maupun penyusunan buku laporan kerja praktik ini. Namun penulis berharap buku laporan ini dapat menambah wawasan pembaca dan dapat menjadi sumber referensi.

Melalui buku laporan ini penulis juga ingin menyampaikan rasa terima kasih kepada orang-orang yang telah membantu menyusun laporan kerja praktik baik secara langsung maupun tidak langsung antara lain:

1. Kedua orang tua penulis.
2. Bapak Agus Budi Raharjo, S.Kom, M.Kom., Ph.D. selaku dosen pembimbing kerja praktik sekaligus koordinator kerja praktik.
3. Bapak Ir. Febrita Kusuma Wardana, S.T., M.MT, IPM selaku pembimbing lapangan selama kerja praktik berlangsung.
4. Teman-teman penulis yang senantiasa memberikan semangat ketika penulis melaksanakan KP.

Surabaya, 30 November 2024
Thomas Juan Mahardika Suryono

[Halaman ini sengaja dikosongkan]

BAB I
PENDAHULUAN

1.1. Latar Belakang

Sebagai seorang mahasiswa, dalam mempersiapkan kehidupan pasca kampus, diperlukan bekal pengetahuan yang bersifat teoritis dan praktis. Pengetahuan teoritis umumnya didapatkan dari perkuliahan, membaca literatur terkait bidang studi, mengikuti bootcamp atau course, dan mempelajari dari sumber-sumber lainnya. Berbeda dengan pengetahuan teoritis yang didapatkan dan diajarkan di kelas, pengetahuan praktis atau terjun langsung ke dunia kerja sulit untuk didapatkan di lingkungan perkuliahan. Ketika mengikuti perkuliahan di kelas, mahasiswa tidak mendapatkan situasi kerja yang sebenarnya. Oleh karena itu, untuk membantu mahasiswa terjun langsung untuk mengetahui situasi lingkungan kerja, perguruan tinggi mewujudkannya dengan memberi kesempatan kerja prakik di lingkungan kerja bagi mahasiswa.

Salah satu instansi yang memberikan kesempatan bagi mahasiswa untuk mengembangkan keterampilan praktis di dunia kerja adalah PT Pertamina EP Cepu Field JTB. PT Pertamina EP Cepu Field JTB menyediakan lapangan kerja yang luas dan terbuka untuk seluruh orang atau calon pekerja dari berbagai bidang ilmu yang akan berkolaborasi demi menjalankan visi misi PT Pertamina EP Cepu Field JTB sebagai perusahaan energi yang berkelanjutan, berorientasi pada kesejahteraan masyarakat, dan mengutamakan keamanan. Dengan begitu, PT

Pertamina EP Cepu Field JTB memberi kesempatan kepada mahasiswa untuk merasakan lingkungan kerja serta mengembangkan ide, inovasi, dan potensi diri melalui program kerja praktik [1].

Beberapa mesin yang digunakan pada PT Pertamina EP Cepu Field JTB memiliki komponen filter, yang seiring berjalannya waktu, filter semakin kotor. Hal tersebut menyebabkan nilai perbedaan tekanan atau differential pressure meningkat. Oleh karena itu, forecast perlu dilakukan agar dapat dilakukan tindakan penggantian filter sebelum nilai differential pressure menyentuh batas yang telah ditentukan. Karena jika nilai differential pressure menyentuh batas, mesin berpotensi berhenti beroperasi sehingga menyebabkan kerugian besar. *Long short term memory* (LSTM) dan *autoregressive integrated moving average* (ARIMA) dipilih menjadi model *forecast* karena sudah banyak dipercaya menjadi model *forecast* pada kasus lain. Salah satu faktornya adalah kemampuannya untuk melakukan *forecast* data nonlinear untuk LSTM dan kemampuan menangani data nonstationer untuk ARIMA.

Melalui kegiatan kerja praktik ini, diharapkan mahasiswa Teknik Informatika mampu membantu menyelesaikan berbagai permasalahan dan tugas di PT Pertamina EP Cepu Field JTB serta mampu menyerap ilmu lapangan sebanyak-banyaknya sebagai bekal dunia kerja selepas menempuh jenjang perkuliahan.

1.2. Tujuan

Tujuan kerja praktik ini adalah menyelesaikan kewajiban nilai kerja praktik sebesar 4 SKS dan menyelesaikan permasalahan yang diberikan oleh PT Pertamina EP Cepu Field JTB dengan menerapkan ilmu-ilmu informatika yang didapatkan di bangku perkuliahan. Selain itu, mahasiswa memiliki tujuan untuk memperoleh gambaran nyata dari penerapan ilmu dan teori informatika yang diperoleh di bangku perkuliahan serta dapat menerapkan ilmu tersebut dengan kenyataan yang ada di dunia kerja.

Secara khusus, tujuan dari kerja praktik ini adalah:

- Mengimplementasi LSTM dan ARIMA sebagai model *forecast*.
- Mengevaluasi kinerja LSTM dan ARIMA sebagai model *forecast.*
- Mem*forecast differential pressure* dan *pressure compressor discharge* beberapa bulan ke depan.

1.3. Manfaat

Adapun manfaat kerja praktik mahasiswa Departemen Teknik Informatika ITS bagi PT Pertamina EP Cepu Field JTB, mahasiswa, dan Departemen Teknik Informatika ITS adalah sebagai berikut:

- Kerja praktik sebagai sarana untuk memberikan pertimbangan dalam menentukan kriteria kerja yang dibutuhkan oleh instansi yang bersangkutan, dilihat dari

segi sumber daya manusia yang dihasilkan lembaga pendidikan tinggi.

- PT Pertamina EP Cepu Field JTB memperoleh masukan objektif yang dapat dipertanggungjawabkan secara akademis guna membantu peningkatan wawasan dan produktivitas masing-masing.
- Mahasiswa dapat mengaplikasikan ilmu yang diperoleh dari perkuliahan untuk digunakan dalam menyelesaikan permasalahan di instansi pemerintah khususnya PT Pertamina EP Cepu Field JTB dan menjadi lebih siap dalam menghadapi dunia kerja.
- Departemen Teknik Informatika ITS dapat menjalin kerja sama secara tidak langsung dengan PT Pertamina EP Cepu Field JTB sebagai salah satu perusahaan ternama di Indonesia.
- Departemen Teknik Informatika ITS memperoleh informasi mengenai keadaan umum PT Pertamina EP Cepu Field JTB melalui laporan kerja praktik lapangan serta mahasiswa dapat menjalin relasi dengan pihak-pihak terkait.
- Mempersingkat waktu penggantian filter yang sebelumnya perlu menunggu notifikasi nilai *differential pressure* menyentuh batas.
- Menghindari kerugian besar yang diakibatkan berhentinya operasi mesin.

1.4. Rumusan Masalah

Berikut ini rumusan masalah pada kerja praktik evaluasi kinerja plugin Reasonal:

- Bagaimana implementasi LSTM dan ARIMA sebagai model *forecast differential pressure*?
- Bagaimana kinerja LSTM dan ARIMA sebagai model *forecast differential pressure*?
- Bagaimana metode mem*forecast differential pressure* dan *pressure compressor discharge* beberapa bulan ke depan?

1.5. Lokasi dan Waktu Kerja Praktik

Kerja praktik ini dilaksanakan pada waktu dan tempat sebagai berikut:

Lokasi : Hybrid
Waktu : 1 September – 30 November 2024
Hari Kerja : Senin – Jumat
Jam Kerja : Fleksibel (minimal 12 jam per minggu)

1.6. Metodologi Kerja Praktik

1.6.1. Perumusan Masalah

Dalam tahap ini kami perlu mengetahui permasalahan apa saja yang terjadi dan dapat diselesaikan. Kami juga perlu mengetahui semua kebutuhan dalam permasalahan tersebut.

1.6.2. Studi Literatur

Setelah ditentukan rumusan masalah mengenai sistem yang akan dibuat, dilakukan studi literatur mengenai implementasinya. Pada tahap ini dilakukan proses pencarian, pembelajaran, dan pengumpulan informasi tentang tahapan bagaimana melakukan *forecast*, model *forecast* yang berakurasi tinggi dan bagaimana implementasinya.

1.6.3. Analisis dan Perancangan

Pada tahap ini, dijelaskan hasil dari studi literatur yang telah dilakukan. Dari berbagai metode yang ditemukan selama proses literasi, dianalisis metode yang paling sesuai dan efektif untuk menyelesaikan masalah. Selain itu, ditentukan bahasa pemrograman yang akan digunakan serta batasan data yang akan dimanfaatkan, sehingga hasil yang diharapkan dapat dicapai.

1.6.4. Implementasi Sistem

Pada tahap ini dijelaskan implementasi program yang digunakan pada *forecast*.

1.6.5. Pengujian dan Evaluasi

Setelah implemetasi yang telah direncanakan selesai dibuat, perlu dilakukan evaluasi untuk menguji akurasi *forecast*. Pengujian akan menggunakan nilai MAPE, RMSE, dan R^2.

1.6.6. Kesimpulan dan Saran

Pada bab ini, dipaparkan kesimpulan yang dapat diambil dan juga saran dalam pengerjaan kerja praktik.

1.7. Sistematika Laporan

Laporan kerja praktik ini terdiri dari tujuh bab dengan rincian sebagai berikut:

1.7.1. Bab I Pendahuluan

Bab ini berisi latar belakang, tujuan, manfaat, rumusan masalah, lokasi dan waktu kerja praktik, dan sistematika laporan.

1.7.2. Bab II Profil Perusahaan

Bab ini berisi gambaran umum PT Pertamina EP Cepu Field JTB mulai dari profil, lokasi perusahaan.

1.7.3. Bab III Tinjauan Pustaka

Bab ini berisi dasar teori dari teknologi yang digunakan dalam menyelesaikan proyek kerja praktik.

1.7.4. Bab IV Analisis dan Perancangan Infrastruktur Sistem

Bab ini berisi mengenai tahap metodologi dalam menyelesaikan proyek kerja praktik.

1.7.5. Bab V Implementasi Sistem

Bab ini berisi uraian tahap - tahap yang dilakukan untuk proses implementasi.

1.7.6. Bab VI Pengujian dan Evaluasi

Bab ini berisi hasil uji coba dan evaluasi dari sistem yang telah dikembangkan selama pelaksanaan kerja praktik.

1.7.7. Bab VII Kesimpulan dan Saran

Bab ini berisi kesimpulan dan saran yang didapat dari proses pelaksanaan kerja praktik.

[Halaman ini sengaja dikosongkan]

BAB II
PROFIL PERUSAHAAN

2.1. PT Pertamina EP Cepu Field JTB

PT Pertamina EP Cepu (PEPC) adalah anak perusahaan PT Pertamina (Persero) yang bergerak di bidang eksplorasi dan produksi minyak dan gas bumi. Proyek Jambaran-Tiung Biru (JTB) adalah salah satu proyek strategis PEPC yang bertujuan untuk meningkatkan produksi gas di Indonesia. Proyek ini mencakup pengembangan lapangan gas Jambaran dan Tiung Biru di Blok Cepu.

2.2. Lokasi

PT Pertamina EP Cepu Field JTB memiliki kantor, yaitu:
Jalan Raya Gayam-Ngasem, Dusun Pohwayang, Desa Bandungrejo, Kecamatan Ngasem, Kabupaten Bojonegoro, Jawa Timur

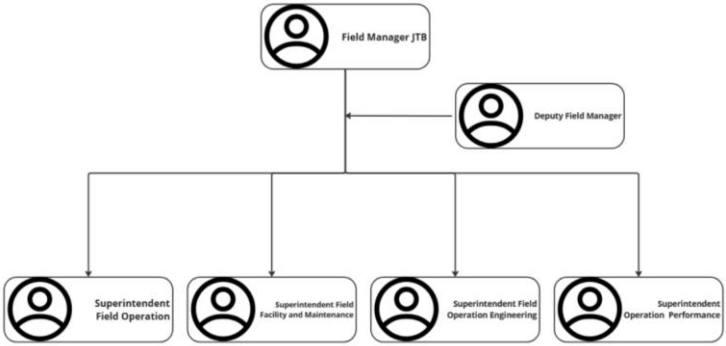
2.3. Visi dan Misi Perusahaan

- Visi: Menjadi perusahaan energi nasional kelas dunia yang bertumbuh kembang dan unggul.
- Misi: Melakukan kegiatan eksplorasi dan produksi minyak dan gas bumi secara profesional dengan mengutamakan keselamatan, kesehatan kerja, dan kelestarian

lingkungan serta memberikan nilai tambah maksimal bagi stakeholder.

2.4. Struktur Organisasi PT Pertamina EP Cepu Field JTB

Struktur organisasi PT Pertamina EP Cepu Field JTB dijelaskan pada Gambar 2.1.



Gambar 2.1 Struktur Organisasi PT Pertamina EP Cepu Field JTB

BAB III
TINJAUAN PUSTAKA

3.1. Python

Python adalah bahasa pemrograman yang memungkinkan penggunanya bekerja dengan lebih cepat dan mengintegrasikan sistem secara lebih efisien[2]. Python merupakan salah satu bahasa pemrograman yang sangat penting dalam data science karena beberapa alasan utama. Python memiliki sintaks yang sederhana, intuitif, dan mudah dipelajari, sehingga cocok untuk pemula. Python bersifat lintas platform dan portabel, memungkinkan pengembang menjalankan kode di berbagai sistem operasi tanpa perubahan signifikan. Python dilengkapi dengan pustaka yang kuat, seperti Pandas, NumPy, dan Matplotlib, yang mempermudah analisis dan visualisasi data[3].

3.2. Time Series

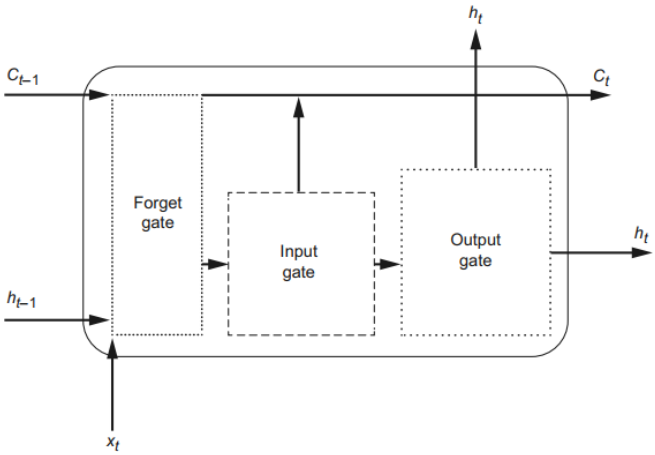
Time series merupakan himpunan data yang terurut dalam waktu. Data tersebut biasanya dalam rentang waktu yang sama, yang dapat diartikan setiap data terpisahkan interval yang sama. Sederhananya, data tersebut bisa dikumpulkan pada setiap jam atau setiap menit, atau dapat dirata-rata setiap bulan atau tahun. Beberapa contoh time series meliputi nilai penutup saham tertentu, konsumsi listrik rumah tangga, atau temperatur luar ruangan[4].

3.3. Forecast

Forecast merupakan prediksi masa depan menggunakan data historis dan pengetahuan terhadap

kejadian masa depan yang dapat mempengaruhi *forecast*. *Forecast* berbeda dengan prediksi karena data *time series* untuk *forecast* terurut berdasarkan waktu, sedangkan prediksi independen terhadap waktu. *Forecast* terbagi menjadi dua, yakni univariat dan multivariat. *Forecast* univariat adalah *forecast* yang hanya menggunakan satu variabel, sedangkan *forecast* multivariat menggunakan lebih dari satu variabel[4].

3.4. *Long Short Term Memory*



Gambar 3.1 Arsitektur Sebuah Neuron LSTM

Long Short Term Memory (LSTM) merupakan salah satu model *deep learning* yang dapat digunakan untuk *forecast* dengan arsitektur sebuah neuronnya terdapat pada Gambar 4.1. Terdapat penambahan *cell state*, yang dilambangkan *C*. *Cell state* ini yang memperbolehkan jaringan untuk menyimpan informasi masa lalu di dalam jaringan untuk watku yang lebih lama sehingga menyelesaikan *vanishing gradient problem*.

Terdapat x_t yang merupakan elemen sebuah rangkaian sedang yang diproses dan h_t yang merupakan *hidden state*. Pada kasus ini, *cell state* C_t dan *hidden* h_t diberikan kepada elemen berikutnya dari rangkaian, memastikan informasi masa lalu digunakan sebagai masukan untuk elemen berikutnya pada rangkaian yang sedang diproses[4].

Terdapat tiga *gate*, yakni *forget gate, input gate, dan output gate*. *Forget gate* merupakan *gate* pertama dalam sel LSTM yang berfungsi untuk menentukan informasi, baik dari masa lalu maupun masa kini, yang sebaiknya dilupakan atau disimpan dalam jaringan. Setelah informasi melewati *forget gate*, dilanjutkan masuk ke *input gate*. Tahap ini digunakan untuk menentukan informasi mana yang relevan dengan elemen terkini dari rangkaian. Setelah informasi melewati *forget gate* dan *input gate*, *output gate* menggunakan informasi tersebut untuk memproses elemen terkini dari rangkaian[4].

3.5. AutoRegressive Integrated Moving Average

Autoregressive integrated moving average (ARIMA) merupakan kombinasi dari proses *autoregressive* AR (p), integration I (d), dan proses *moving average* MA (q). Proses ARIMA menyatakan bahwa nilai masa kini bergantung pada nilai masa lalu yang berasal dari porsi AR (p) dan ralat masa lalu yang berasal dari porsi MA (q). Namun, bukannya menggunakan *series* orisinil, proses ARIMA menggunakan *series* terdiferensiasi. Nilai p menentukan banyaknya nilai *lagged* yang termasuk dalam model, q menentukan banyaknya *lagged error*, dan d mendefinisikan tingkat integrasi. ARIMA mendukung data nonstationer[4].

[Halaman ini sengaja dikosongkan]

BAB IV
ANALISIS DAN PERANCANGAN INFRASTRUKTUR SISTEM

4.1. Analisis Sistem

Pada bab ini akan dijelaskan mengenai identifikasi kebutuhan dalam mengimplementasi LSTM dan ARIMA sebagai model *forecast.*

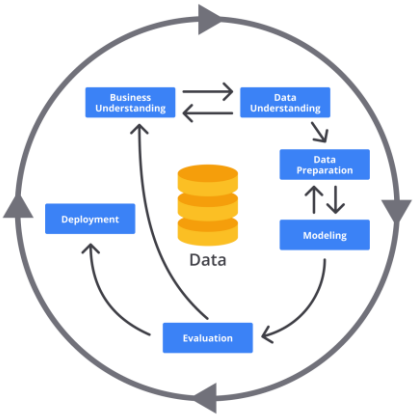
4.1.1. Identifikasi Kebutuhan

Berdasarkan tugas dari pembimbing lapangan, terdapat dua kebutuhan utama dari implementasi model *forecast* yaitu meliputi :

1. Forecast nilai IN DP pada strainer inlet beberapa bulan ke depan
2. Forecast nilai MON DP, MON DP A, MON DP B, dan MON DP C pada particle filter beberapa bulan ke depan
3. Forecast nilai PCD pada gas turbine generator beberapa bulan ke depan

4.2. Perancangan Infrastruktur Sistem

Dalam bab perancangan Infrastruktur ini dijelaskan tentang metodologi yang digunakan untuk mengimplimentasi LSTM dan ARIMA sebagai model *forecast*. Gambar 4.1 merupakan flowchart perancangan infrastruktur sistem.



Gambar 4.1 Flowchart CRISP DM

Metodologi yang digunakan pada implementasi ini adalah CRISP DM. Berikut rinciannya.

1. Business understanding

   Permasalahan yang dialami perusahaan adalah ketiadaan model *forecast* yang dapat digunakan untuk mem*forecast* nilai *differential pressure* dan *pressure compressor discharge*. Diharapkan setelah nilai tersebut di*forecast*, dapat diketahui kapan sebaiknya dilakukan pembersihan strainer, penggantian filter, dan *offline wash* gas turbin dengan memeriksa batasan nilai. Akurasi yang diharapkan adalah MAPE < 10%.
2. Data understanding

   Tabel 4.1 merupakan rincian data yang digunakan pada strainer inlet dan particle filter, sedangkan tabel 4.2 merupakan rincian data yang digunakan pada gas turbin generator.

Tabel 4.1 Rincian Data pada Strainer Inlet dan Particle Filter

Variabel	Satuan	Deskripsi
Differential Pressure	psi	Perbedaan tekanan masuk dan keluar
Flow	MMSCFD	Rate flow gas
Temperature	degF	Suhu gas
Corrected volume	MMSCFD	Volume gas yang disesuaikan kondisi standar temperatur dan tekanan

Tabel 4.2 Rincian Data pada Gas Turbin Generator

Variabel	Satuan	Deskripsi
Engine T5 Average	degF	Rata-rata suhu gas buang dari bagian T5
Engine Air Inlet Temperature (T1)	degF	Suhu gas yang masuk ke kompresor
Turbine Air Inlet Filter Differential Pressure	inH2O	Perbedaan tekanan antara sisi hulu dan hilir filter gas
Pressure Compressor Discharge (PCD)	psig	Tekanan gas keluaran kompresor
Gas Fuel Pressure from HP Gas Fuel Header	psig	Tekanan gas bertekanan tinggi dari HP (High-Pressure) Gas Fuel Header
Gas Fuel Temperature from HP Gas Fuel Header	degF	Suhu gas bertekanan tinggi dari HP (High-Pressure) Gas Fuel Header

Gas Fuel Flow Rate from HP Gas Fuel Header	lb/h	Laju aliran gas bertekanan tinggi dari HP (High-Pressure) Gas Fuel Header

3. Data preparation

   Pada data strainer inlet, terdapat dua variabel corrected volume yang perlu ditambah, juga dua variabel out flow yang perlu ditambah. Pada particle filter, hanya terdapat dua variabel corrected volume yang perlu ditambah. Rentang data yang digunakan pada strainer inlet dan particle filter dari tanggal 1 Januari 2023 pukul 01.00 hingga 10 September 2024 pukul 00.00 dengan jeda per jam. Pada gas turbin generator, dilakukan penyesuaian data time series yang awalnya horizontal, diubah menjadi vertikal. Banyak didapat missing values yang diisi dengan nilai terakhir waktu sebelumnya. Rentang data yang digunakan pada gas turbin generator dari tanggal 12 Mei 2023 pukul 06.00 hingga 22 Agustus 2024 pukul 18.00.

4. Modeling

   Terdapat tiga model yang digunakan di setiap data, yakni LSTM univariat, LSTM multivariat, dan ARIMA. Pada LSTM, digunakan dua layer dengan masing-masing layer memiliki 50 neuron, epoch bernilai 100 dan early stopping hingga 10. Pada LSTM multivariat strainer inlet dan particle filter, variabel independen yang dipilih adalah memprioritaskan dari sisi pengetahuan teknis, kemudian diambil variabel yang berkoefisien korelasi

lebih dari 0.5 pada nilai mutlaknya dengan variabel dependen sehingga dihasilkan variabel sesuai Tabel 4.1. Pada LSTM multivariat gas turbin generator, digunakan variabel pada Tabel 4.2 sesuai dengan pengetahuan teknis. Pada ARIMA, digunakan data *lagged* satu jam sebagai variabel *x* dengan tes ADF (Augmented Dickey-Fuller). Data test yang digunakan adalah data tiga bulan terakhir, sedangkan sisanya menjadi data train.

5. Evaluation
   Evaluasi secara rinci dijelaskan pada Bab VI.
6. Deployment
   Deployment model dilakukan pada platform LEADS milik perusahaan. Hasil forecast menjadi acuan rekomendasi kapan dilakukan tindakan pembersihan strainer, penggantian filter, dan *offline wash* gas turbin.

[Halaman ini sengaja dikosongkan]

BAB V
IMPLEMENTASI SISTEM

Bab ini membahas tentang implementasi LSTM dan ARIMA sebagai model *forecast*. Digunakan kode pada strainer inlet sebagai contoh.

5.1. Implementasi pada Strainer Inlet

Berikut implementasi model pada kasus mesin strainer inlet.

5.1.1. Import Library dan Persiapan Data

Pada tahap pertama, terdapat import library dan data preparation yang juga dilakukan pada data lain. Terdapat penyesuaian data corrected volume dan out flow yang keduanya ditambah antar pasangannya.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import pmdarima as pm
from sklearn.preprocessing import
MinMaxScaler
from sklearn.metrics import
mean_absolute_percentage_error,
mean_squared_error, r2_score
from keras.api.models import Sequential
from keras.api.layers import Dense, LSTM
from keras.api.callbacks import
EarlyStopping

df = pd.read_excel('Strainer Inlet
2ndstage membrane Unit 242.xlsx', [1, 2],
header=None, index_col=0,
usecols=range(1,13), skiprows=range(5))
df = pd.concat([df[2], df[1]])
```

```
df =
df.rename(columns=dict(zip(range(2,13),
['IN DP', 'IN FLOW', 'IN PRESS', 'IN TEMP
2', 'OUT FLOW 1', 'OUT FLOW 2', 'OUT
PRESS', 'CONT PRESS', 'CORRECTED VOLUME
2', 'CORRECTED VOLUME 1', 'IN TEMP 1'])))
df.index.name = 'Datetime'

for i in range(1, len(df)-1):
    if (df.iloc[i-1, 8] == 0) &
(df.iloc[i+1, 8] == 0):
        df.iloc[i, 8] = 0
    if (df.iloc[i-1, 9] == 0) &
(df.iloc[i+1, 9] == 0):
        df.iloc[i, 9] = 0
df['CORRECTED VOLUME 1'] = df['CORRECTED
VOLUME 1'] + df['CORRECTED VOLUME 2']
df = df.drop('CORRECTED VOLUME 2',
axis=1)
df = df.rename(columns={'CORRECTED VOLUME
1': 'CORRECTED VOLUME'})

df['OUT FLOW 1'] = df['OUT FLOW 1'] +
df['OUT FLOW 2']
df = df.drop('OUT FLOW 2', axis=1)
df = df.rename(columns={'OUT FLOW 1':
'OUT FLOW'})
```

Kode Sumber 5.1 Import Library dan Persiapan Data pada Strainer Inlet

5.1.2. Implementasi LSTM Univariat

Berikut implementasi LSTM Univariat pada Strainer Inlet.

```
data = df['IN DP']

for duration in [3, 6, 12]:
```

```
    multidur = duration * 30 * 24
    train_data = data[:-multidur]
    test_data = data[-multidur:]

    scaler = MinMaxScaler()
    train_scaled =
scaler.fit_transform(train_data.to_frame()
)
    test_scaled =
scaler.transform(test_data.to_frame())

    n_steps = 30
    X_train, y_train = [], []

    for i in range(n_steps,
len(train_scaled)):
        X_train.append(train_scaled[i -
n_steps : i])
        y_train.append(train_scaled[i])

    X_train, y_train = np.array(X_train),
np.array(y_train)


    model = Sequential(
        [
            LSTM(
                50, return_sequences=True,
input_shape=(X_train.shape[1],
X_train.shape[2])
            ),
            LSTM(50),
            Dense(1),
        ]
    )
    model.compile(optimizer="adam",
loss="mean_squared_error")
```

```
    early_stopping = EarlyStopping(
        monitor="val_loss", patience=10,
restore_best_weights=True
    )

    history = model.fit(
        X_train,
        y_train,
        epochs=100,
        batch_size=32,
        validation_split=0.2,
        callbacks=[early_stopping],
    )

    X_test, y_test = [], []
    for i in range(n_steps,
len(test_scaled)):
        X_test.append(test_scaled[i -
n_steps : i])
        y_test.append(test_scaled[i])

    X_test, y_test = np.array(X_test),
np.array(y_test)

    y_pred_scaled = model.predict(X_test)
    y_pred =
scaler.inverse_transform(y_pred_scaled)
    y_actual =
scaler.inverse_transform(y_test)

    plt.figure(figsize=(14, 7))
    plt.plot(test_data.index[n_steps:],
y_actual, label="Actual IN DP",
color="blue")
    plt.plot(
        test_data.index[n_steps:],
```

```
        y_pred,
        label="Predicted IN DP",
        color="red",
        linestyle="--",
    )
    plt.xlabel("Date")
    plt.ylabel("IN DP")
    plt.title(f"Strainer Inlet Predicted
vs Actual IN DP for the Last {duration}
Months with LSTM Univariate")
    plt.legend()
    plt.savefig(f'Strainer Univariate
{duration} Months')
    plt.show()

    mape =
mean_absolute_percentage_error(y_actual,
y_pred)
    rmse =
np.sqrt(mean_squared_error(y_actual,
y_pred))
    r2 = r2_score(y_actual, y_pred)

    print("MAPE:", mape)
    print("RMSE:", rmse)
    print("R²:", r2)
```

Kode Sumber 5.2 Implemetasi LSTM Univariat pada Strainer Inlet

Kode Sumber 5.2 merupakan salah satu contoh implementasi model pada salah satu data. Implementasi ini dilakukan pada setiap kombinasi model dan variabel yang ingin di*forecast*, yakni mencoba mengetes *forecast* 3, 6, 12 bulan ke depan, membagi data test dan train sesuai durasi *forecast*, lalu membandingkan hasil

forecast dengan aktual menggunakan metrik akurasi MAPE, RMSE, dan R^2.

BAB VI
PENGUJIAN DAN EVALUASI

Bab ini menjelaskan tahap uji coba sistem yang telah dikembangkan untuk memastikan fungsionalitasnya. Pengujian dilakukan guna memverifikasi kesesuaian hasil implementasi dengan analisis dan desain yang telah dibuat.

6.1. Tujuan Pengujian

Pengujian dilakukan terhadap hasil *forecast* untuk ditinjau kinerjanya terhadap nilai tes.

6.2. Kriteria Pengujian

Penilaian atas pencapaian tujuan pengujian didapatkan dengan memperhatikan beberapa hasil yang diharapkan berikut :

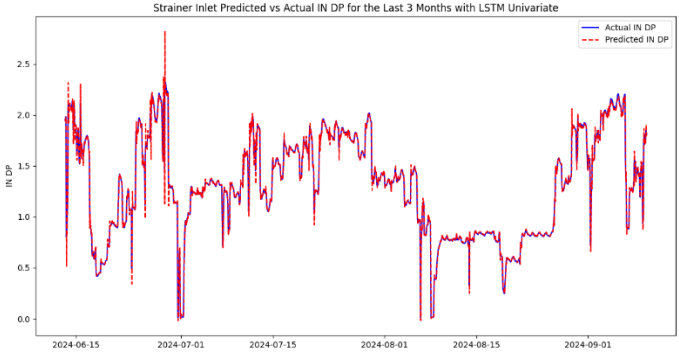
a. MAPE (Mean Absolute Percentage Error)
b. RMSE (Root Mean Squared Error)
c. R^2

6.3. Skenario Pengujian

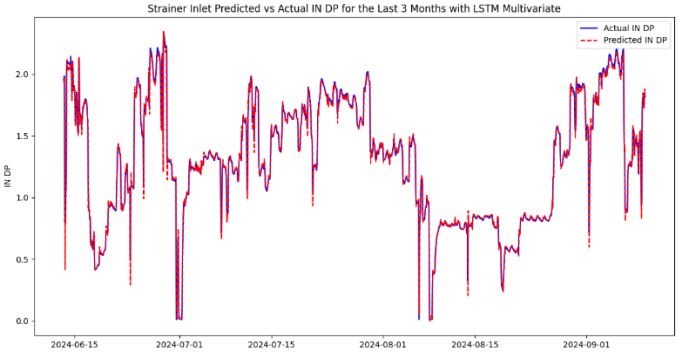
Skenario pengujian dilakukan dengan mengevaluasi hasil *forecast* dengan nilai asli selama tiga bulan ke depan.

6.4. Evaluasi Pengujian

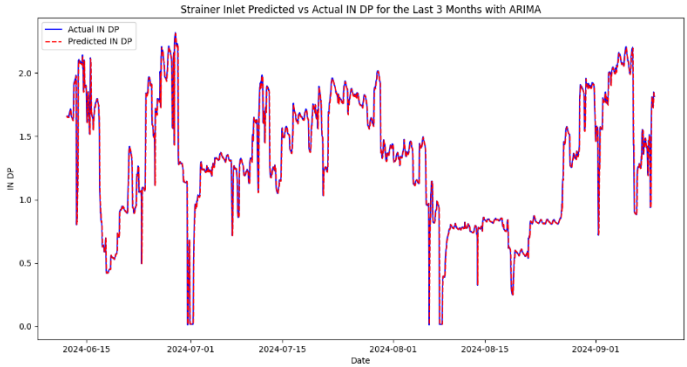
Berikut hasil evaluasi pengujian pada IN DP strainer inlet.



Gambar 6.1 Grafik Perbandingan Prediksi dan Aktual IN DP Strainer Inlet Menggunakan LSTM Univariat



Gambar 6.2 Grafik Perbandingan Prediksi dan Aktual IN DP Strainer Inlet Menggunakan LSTM Multivariat

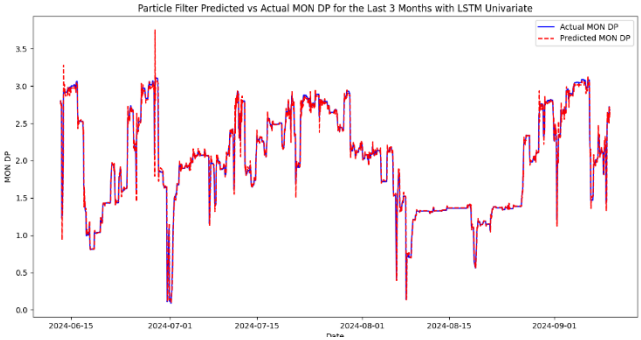


Gambar 6.3 Grafik Perbandingan Prediksi dan Aktual IN DP Strainer Inlet Menggunakan ARIMA

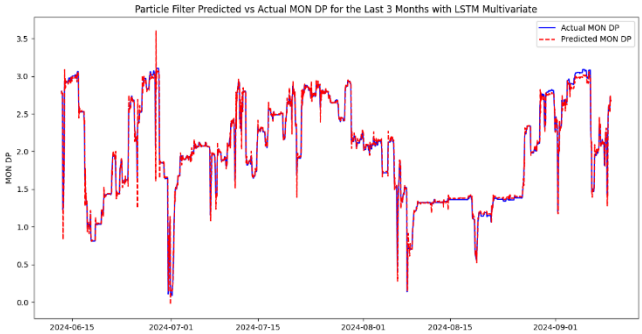
Tabel 6.1 Perbandingan Kinerja Model pada IN DP Strainer Inlet

Model	MAPE	RMSE	R^2
LSTM Uni	0.0942	0.073	0.9778
LSTM Multi	0.0754	0.0708	0.9791
ARIMA	0.1037	0.0759	0.9759

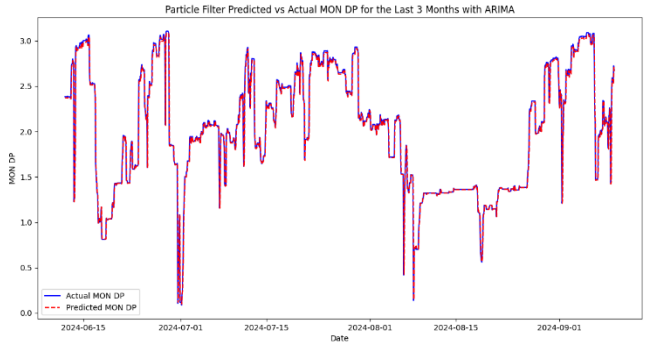
Hasil yang paling unggul didapat oleh model LSTM Multivariat di setiap parameter. Selanjutnya, berikut hasil evaluasi pengujian pada MON DP particle filter.



Gambar 6.4 Grafik Perbandingan Prediksi dan Aktual MON DP Particle Filter Menggunakan LSTM Univariat



Gambar 6.5 Grafik Perbandingan Prediksi dan Aktual MON DP Particle Filter Menggunakan LSTM Multivariat



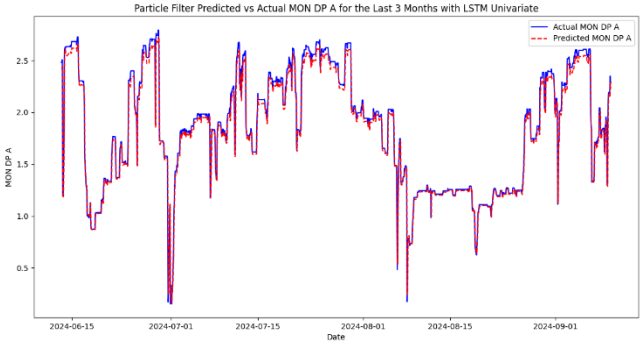
Gambar 6.6 Grafik Perbandingan Prediksi dan Aktual MON DP Particle Filter Menggunakan ARIMA

Tabel 6.2 Perbandingan Kinerja Model pada MON DP Particle Filter

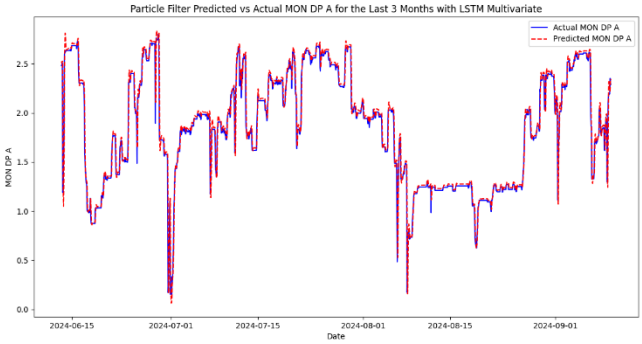
Model	MAPE	RMSE	R^2
LSTM Uni	0.0286	0.0885	0.9816
LSTM Multi	0.027	0.0892	0.9813
ARIMA	0.0299	0.0938	0.9792

Hasil yang paling unggul didapat oleh LSTM multivariat jika ditinjau dari nilai MAPE dan LSTM univariat jika ditinjau dari nilai RMSE dan R^2. Selanjutnya,

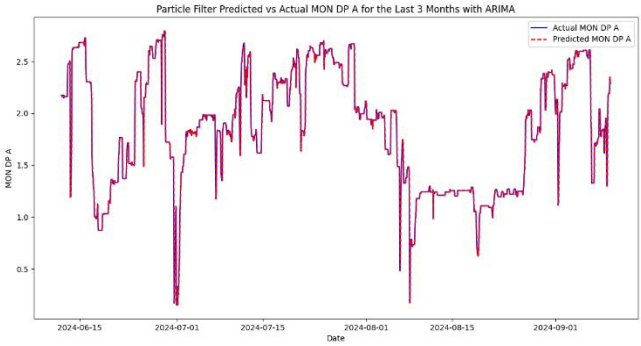
berikut hasil evaluasi pengujian pada MON DP A particle filter.



Gambar 6.7 Grafik Perbandingan Prediksi dan Aktual MON DP A Particle Filter Menggunakan LSTM Univariat



Gambar 6.8 Grafik Perbandingan Prediksi dan Aktual MON DP A Particle Filter Menggunakan LSTM Multivariat

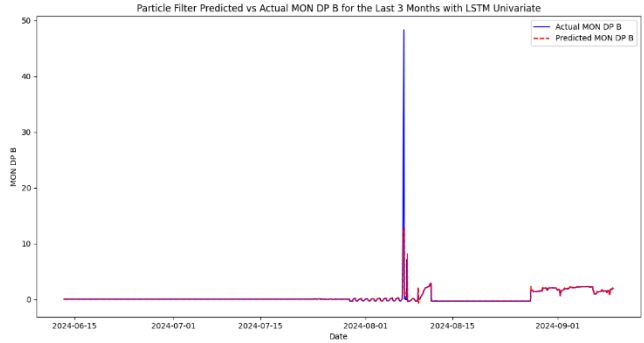


Gambar 6.9 Grafik Perbandingan Prediksi dan Aktual MON DP A Particle Filter Menggunakan ARIMA

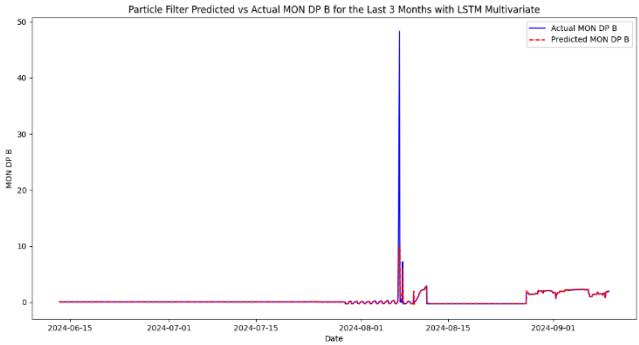
Tabel 6.3 Perbandingan Kinerja Model pada MON DP A Particle Filter

Model	MAPE	RMSE	R^2
LSTM Uni	0.0348	0.0879	0.9754
LSTM Multi	0.0414	0.1039	0.9656
ARIMA	0.0233	0.0802	0.9794

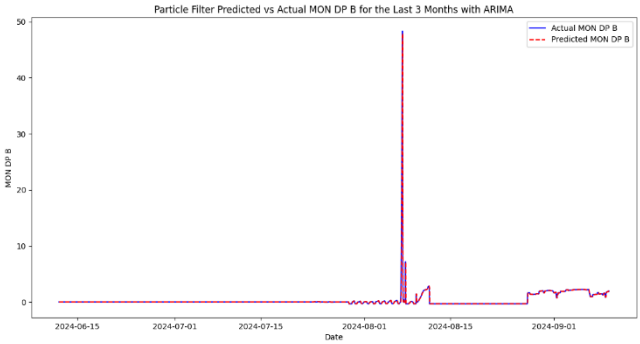
Hasil yang paling unggul didapat oleh ARIMA pada setiap metrik. Selanjutnya, berikut hasil evaluasi pengujian pada MON DP B particle filter.



Gambar 6.10 Grafik Perbandingan Prediksi dan Aktual MON DP B Particle Filter Menggunakan LSTM Univariat



Gambar 6.11 Grafik Perbandingan Prediksi dan Aktual MON DP B Particle Filter Menggunakan LSTM Multivariat



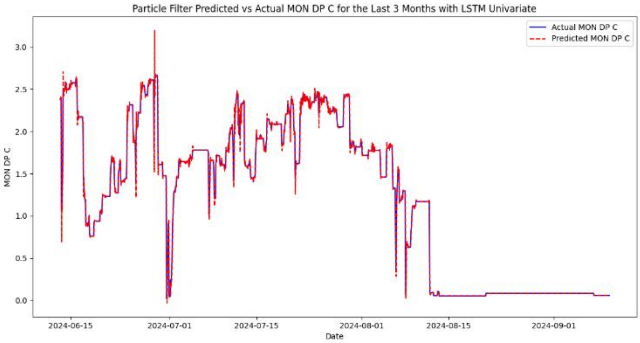
Gambar 6.12 Grafik Perbandingan Prediksi dan Aktual MON DP B Particle Filter Menggunakan ARIMA

Tabel 6.4 Perbandingan Kinerja Model pada MON DP B Particle Filter

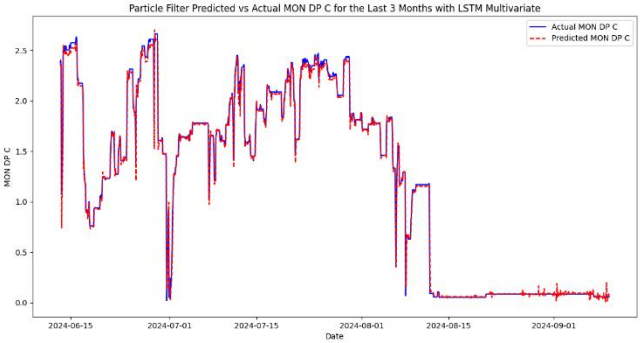
Model	MAPE	RMSE	R^2
LSTM Uni	0.4476	1.1411	0.5914
LSTM Multi	0.5192	1.2393	0.518
ARIMA	0.5375	1.0505	0.6489

Hasil yang paling unggul didapat LSTM univariat jika ditinjau dari MAPE dan ARIMA jika ditinjau dari RMSE dan R^2. Hasil yang kurang akurat tersebut

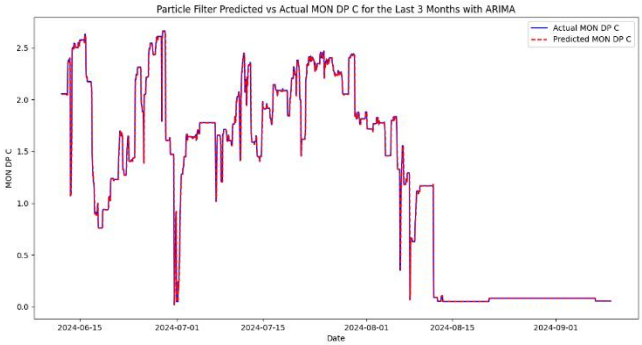
dikarenakan terdapat data anomali yang mencapai hampir 50. Selanjutnya, berikut hasil evaluasi pengujian pada MON DP B particle filter.



Gambar 6.13 Grafik Perbandingan Prediksi dan Aktual MON DP C Particle Filter Menggunakan LSTM Univariat



Gambar 6.14 Grafik Perbandingan Prediksi dan Aktual MON DP C Particle Filter Menggunakan LSTM Multivariat

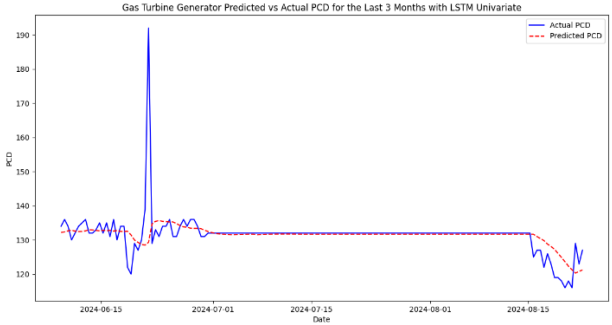


Gambar 6.15 Grafik Perbandingan Prediksi dan Aktual MON DP C Particle Filter Menggunakan ARIMA

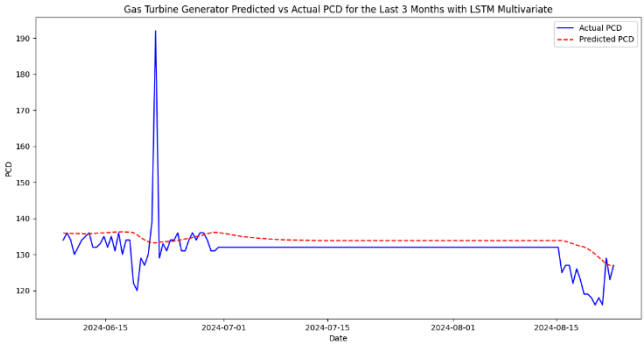
Tabel 6.5 Perbandingan Kinerja Model pada MON DP C Particle Filter

Model	MAPE	RMSE	R^2
LSTM Uni	0.0406	0.0688	0.9942
LSTM Multi	0.0926	0.0743	0.9933
ARIMA	0.054	0.0721	0.9937

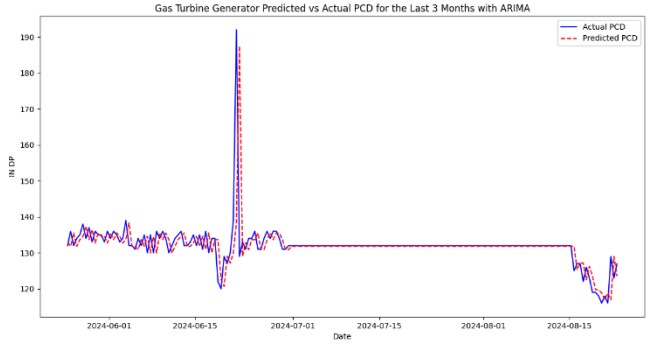
Hasil yang paling unggul didapat LSTM univariat pada semua metrik. Selanjutnya, berikut hasil evaluasi pengujian pada PCD gas turbin generator.



Gambar 6.16 Grafik Perbandingan Prediksi dan Aktual PCD Gas Turbin Generator Menggunakan LSTM Univariat



Gambar 6.17 Grafik Perbandingan Prediksi dan Aktual PCD Gas Turbin Generator Menggunakan LSTM Multivariat

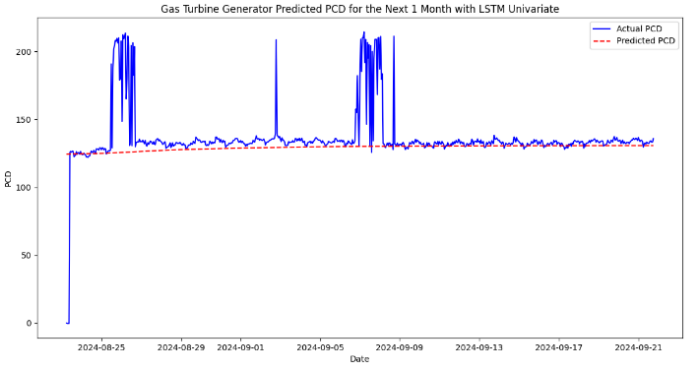


Gambar 6.18 Grafik Perbandingan Prediksi dan Aktual PCD Gas Turbin Generator Menggunakan ARIMA

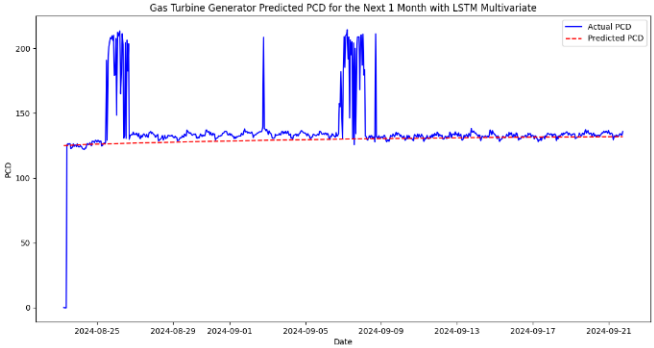
Tabel 6.6 Perbandingan Kinerja Model pada PCD Gas Turbin Generator

Model	MAPE	RMSE	R^2
LSTM Uni	0.014	5.8096	0.1242
LSTM Multi	0.025	6.3971	-0.0619
ARIMA	0.016	6.4603	-0.2348

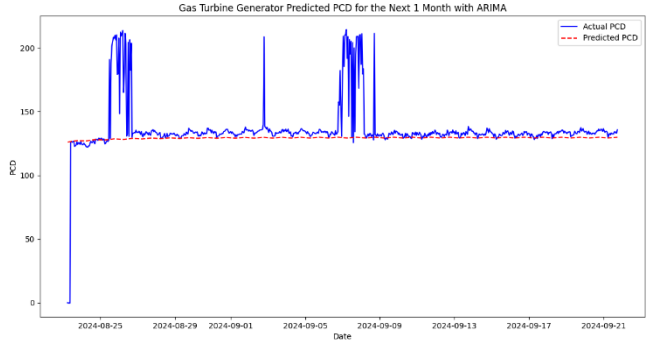
Hasil yang paling unggul didapat oleh LSTM univariat di setiap metrik. Hasil yang kurang akurat tersebut dikarenakan banyaknya missing values pada data aslinya serta sifat data yang jika terjadi fluktuasi, nilainya jauh di atas rata-rata nilai lainnya, seperti contoh pada Gambar 6.18 yang fluktuasinya mencapai 190, sedangkan yang lain di kisaran 120-140. Berikut merupakan evaluasi pengujian pada PCD Gas Turbin Generator dengan data satu bulan setelah data asli, yakni dari tanggal 23 Agustus 2024 pukul 06.00 hingga 21 September 2024 pukul 18.00 dibandingkan dengan data aktual yang kembali disediakan perusahaan.



Gambar 6.19 Grafik Perbandingan Prediksi dan Aktual PCD Gas Turbin Generator Satu Bulan ke Depan Menggunakan LSTM Univariat



Gambar 6.20 Grafik Perbandingan Prediksi dan Aktual PCD Gas Turbin Generator Satu Bulan ke Depan Menggunakan LSTM Multivariat



Gambar 6.21 Grafik Perbandingan Prediksi dan Aktual PCD Gas Turbin Generator Satu Bulan ke Depan Menggunakan ARIMA

Hasil yang paling mendekati didapat LSTM Multivariat. Hasil *forecast* masih belum memproduksi fluktuasi yang mana pada data aktualnya terjadi.

6.5. Evaluasi Hasil, Manfaat, dan Pengembangan Kedepan

Hasil pada subbab 6.4 menunjukkan bahwa tidak ada model yang paling unggul dari setiap kasus.

Semua model patut dicoba. Manfaat yang dapat diambil dari *forecast* ini adalah mengembangkan wawasan ilmu pengetahuan dan teknologi terutama mengenai *forecasting*. Perusahaan juga mendapat manfaat agar dapat mengantisipasi nilai DP dan PCD menyentuh batas, mempersingkat waktu pembersihan stainer, penggantian filter dan *offline wash* gas turbin. Pengembangan yang dapat dilakukan adalah mengaplikasikan model pada mesin lainnya dan mengembangkan model yang lebih akurat.

BAB VII
KESIMPULAN DAN SARAN

7.1. Kesimpulan

Kesimpulan yang didapat setelah melakukan implementasi pengembangan dashboard PowerBI adalah sebagai berikut :

a. Implementasi model LSTM menggunakan dua layer, masing-masing memiliki 50 neuron. Pada LSTM multivariat, dipilih variabel sesuai pengetahuan teknis, kemudian variabel yang berkoefisien korelasi lebih dari 0.5 terhadap nilai mutlaknya dengan variabel dependen. Implementasi ARIMA menggunakan tes ADF.
b. Berdasarkan kinerja model LSTM univariat, LSTM multivariat, dan ARIMA, tidak ada model yang terunggul. Didapatkan hasil akurasi pada variabel IN DP, MON DP, MON DP A, dan MON DP C dengan MAPE di bawah 11% dan R^2 di atas 0.95.
c. *Forecast* satu bulan ke depan PCD gas turbin generator menggunakan ketiga model belum memproduksi fluktuasi yang terjadi pada data aktual.
d. Manfaat yang dapat diambil dari *forecast* ini adalah mengembangkan wawasan ilmu pengetahuan dan teknologi terutama mengenai *forecasting*. Perusahaan juga mendapat manfaat agar dapat mengantisipasi nilai DP dan PCD menyentuh batas, mempersingkat waktu pembersihan stainer, penggantian filter dan *offline wash* gas turbin.

[Halaman ini sengaja dikosongkan]

DAFTAR PUSTAKA

[1] PT Pertamina EP Cepu Field JTB.. Tentang PT Pertamina EP Cepu Field JTB. [ONLINE] Available at: https://www.pertamina.com/. [Diakses 1 Agustus 2024].

[2] Python Software Foundation. About. 2024. [ONLINE] Available at: https://www.python.org/about/. [Diakses 27 November 2024].

[3] GeeksforGeeks. Python for Data Science – Learn the Uses of Python in Data Science. Last Updated: 13 August 2024. [ONLINE] Available at: https://www.geeksforgeeks.org/python-for-data-science/. [Diakses 27 November 2024].

[4] Peixeiro, M. 2022. Time Series Forecasting in Python. Manning, Shelter Island.

[Halaman ini sengaja dikosongkan]

BIODATA PENULIS

Nama	: Thomas Juan Mahardika Suryono
Tempat, Tanggal Lahir	: Surabaya, 7 Maret 2003
Jenis Kelamin	: Laki-laki
Telepon	: 082231146051
Email	: thomasjuanmahardika@gmaiil.com

AKADEMIS

Kuliah	: Departemen Teknik Informatika – FTEIC , ITS
Angkatan	: 2021
Semester	: 7 (Tujuh)

[Halaman ini sengaja dikosongkan]

LAMPIRAN

Data Strainer Inlet

Datetime	IN DP	IN FLOW	IN PRESS	IN TEMP 1	OUT FLOW 1	OUT FLOW 2	OUT PRESS	CONT PRESS	CORRECTED VOLUME 2	CORRECTED VOLUME 1	IN TEMP 2
1/1/23 1:00	0.146	31.379	533.159	118.795	12.003	17.988	532.14	10.011	0	66.414	73.183
1/1/23 2:00	0.146	31.406	533.064	119.351	11.98	18.055	532.121	10.006	0	66.17	73.778
1/1/23 3:00	0.147	31.457	533.865	119.494	11.998	18.086	532.625	9.9972	0	66.189	74.038
…											
9/9/24 22:00	1.826	80.494	550.288	124.611	49.498	34.786	530.611	8.0032	180.146	0	94.935
9/9/24 23:00	1.819	80.402	550.005	124.365	49.507	34.706	530.345	8.0149	180.302	0	94.928
9/10/24 0:00	1.815	80.387	550.211	124.242	49.345	34.786	530.529	8.0092	180.509	0	94.71

Data Particle Filter

Datetime	MON DP	OUT PRESS	OUT TEMP	MON DP A	MON DP B	MON DP C	CONT TEMP	IN PRESS	CORRECTED VOLUME 2	CORRECTED VOLUME 1	IN FLOW A	IN FLOW B	IN FLOW C	IN FLOW D	IN FLOW E
1/1/23 1:00	0.61798	531.742	97.8403	2.17276	0.0267	-0.0238	98.0153	532.416	0	66.4139	-0.0178	69.9624	71.9966	0.02966	0.01608
1/1/23 2:00	0.61798	531.742	97.8403	2.1526	0.0267	-0.0238	97.9961	532.416	0	66.1695	-0.0178	69.9629	71.9816	0.02966	0.01608
1/1/23 3:00	0.61798	531.742	97.8403	2.12755	0.0267	-0.0238	98.0204	532.416	0	66.1891	-0.0178	69.9693	71.9453	0.02966	0.01608
...															
9/9/24 22:00	2.68183	520.959	96.724	2.3017	1.9402	0.05617	96.9217	538.28	180.146	0	67.7582	61.2849	62.4318	67.6493	66.7174
9/9/24 23:00	2.67892	520.672	96.768	2.28968	1.93105	0.05617	96.6573	537.134	180.302	0	67.726	61.2953	62.3938	67.7577	66.5261
9/10/24 0:00	2.67892	520.767	96.2729	2.28968	1.93991	0.05617	96.6313	537.206	180.509	0	67.6758	61.2126	62.3449	67.6827	66.6895

Data Gas Turbin Generator

Parameter	12-May-23		13-May-23		14-May-23			20-Aug-24		21-Aug-24		22-Aug-24	
	06:00	18:00	06:00	18:00	06:00	18:00		06:00	18:00	06:00	18:00	06:00	18:00
Real Power	3884	4264	3964	4240	4101	4362		3056	3325	3095	4620	4338	4764
Engine T5 Average	1241	1263	1237	1253	1253	1264		1160	1185	1162	1246	1266	1279
Engine Air Inlet Temperature (T1)	91	82	82	91	89	82		85	79	85	80	83	80
Turbine Air Inlet Filter Differential Pressure	0.23	0.24	0.23	0.23	0.23	0.23	...	0.29	0.28	0.29	0.31	0.29	0.29
Pressure Compressor Discharge (PCD)	119	122	120	122	120	123		116	118	116	129	123	127
Gas Fuel Pressure from HP Gas Fuel Header	432	432	432	431	432	432		430	430	431	430	429	430
Gas Fuel Temperature from HP Gas Fuel Header	134	134	134	132	135	130		127	124	127	128	127	126
Gas Fuel Flow Rate from HP Gas Fuel Header	3795	3923	3729	3868	3793	3890		3439	3600	3521	4062	3878	4073

Kode Sumber Strainer Inlet

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import pmdarima as pm
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import
mean_absolute_percentage_error,
mean_squared_error, r2_score
from keras.api.models import Sequential
from keras.api.layers import Dense, LSTM
from keras.api.callbacks import EarlyStopping

df = pd.read_excel('Strainer Inlet.xlsx', [1,
2], header=None, index_col=0,
usecols=range(1,13), skiprows=range(5))
df = pd.concat([df[2], df[1]])
df = df.rename(columns=dict(zip(range(2,13),
['IN DP', 'IN FLOW', 'IN PRESS', 'IN TEMP 2',
'OUT FLOW 1', 'OUT FLOW 2', 'OUT PRESS', 'CONT
PRESS', 'CORRECTED VOLUME 2', 'CORRECTED VOLUME
1', 'IN TEMP 1'])))
df.index.name = 'Datetime'

for i in range(1, len(df)-1):
    if (df.iloc[i-1, 8] == 0) & (df.iloc[i+1, 8]
== 0):
        df.iloc[i, 8] = 0
    if (df.iloc[i-1, 9] == 0) & (df.iloc[i+1, 9]
== 0):
        df.iloc[i, 9] = 0
df['CORRECTED VOLUME 1'] = df['CORRECTED VOLUME
1'] + df['CORRECTED VOLUME 2']
df = df.drop('CORRECTED VOLUME 2', axis=1)
df = df.rename(columns={'CORRECTED VOLUME 1':
'CORRECTED VOLUME'})
```

```
df['OUT FLOW 1'] = df['OUT FLOW 1'] + df['OUT
FLOW 2']
df = df.drop('OUT FLOW 2', axis=1)
df = df.rename(columns={'OUT FLOW 1': 'OUT
FLOW'})

data = df['IN DP']

for duration in [3, 6, 12]:
    multidur = duration * 30 * 24
    train_data = data[:-multidur]
    test_data = data[-multidur:]

    scaler = MinMaxScaler()
    train_scaled =
scaler.fit_transform(train_data.to_frame())
    test_scaled =
scaler.transform(test_data.to_frame())

    n_steps = 30
    X_train, y_train = [], []

    for i in range(n_steps, len(train_scaled)):
        X_train.append(train_scaled[i - n_steps
: i])
        y_train.append(train_scaled[i])

    X_train, y_train = np.array(X_train),
np.array(y_train)


    model = Sequential(
        [
            LSTM(
                50, return_sequences=True,
input_shape=(X_train.shape[1], X_train.shape[2])
```

```
            ),
            LSTM(50),
            Dense(1),
        ]
    )
    model.compile(optimizer="adam",
loss="mean_squared_error")

    early_stopping = EarlyStopping(
        monitor="val_loss", patience=10,
restore_best_weights=True
    )

    history = model.fit(
        X_train,
        y_train,
        epochs=100,
        batch_size=32,
        validation_split=0.2,
        callbacks=[early_stopping],
    )

    X_test, y_test = [], []
    for i in range(n_steps, len(test_scaled)):
        X_test.append(test_scaled[i - n_steps :
i])
        y_test.append(test_scaled[i])

    X_test, y_test = np.array(X_test),
np.array(y_test)

    y_pred_scaled = model.predict(X_test)
    y_pred =
scaler.inverse_transform(y_pred_scaled)
    y_actual = scaler.inverse_transform(y_test)

    plt.figure(figsize=(14, 7))
```

```
    plt.plot(test_data.index[n_steps:],
y_actual, label="Actual IN DP", color="blue")
    plt.plot(
        test_data.index[n_steps:],
        y_pred,
        label="Predicted IN DP",
        color="red",
        linestyle="--",
    )
    plt.xlabel("Date")
    plt.ylabel("IN DP")
    plt.title(f"Strainer Inlet Predicted vs
Actual IN DP for the Last {duration} Months with
LSTM Univariate")
    plt.legend()
    plt.savefig(f'Strainer Univariate {duration}
Months')
    plt.show()

    mape =
mean_absolute_percentage_error(y_actual, y_pred)
    rmse = np.sqrt(mean_squared_error(y_actual,
y_pred))
    r2 = r2_score(y_actual, y_pred)

    print("MAPE:", mape)
    print("RMSE:", rmse)
    print("R²:", r2)

data = df.loc['2024-01-19 07:00:00':, 'IN DP']

scaler = MinMaxScaler()
train_scaled =
scaler.fit_transform(data.to_frame())

n_steps = 30
X_train, y_train = [], []
```

```
for i in range(n_steps, len(train_scaled)):
    X_train.append(train_scaled[i - n_steps :
i])
    y_train.append(train_scaled[i])

X_train, y_train = np.array(X_train),
np.array(y_train)

model = Sequential(
    [
        LSTM(
            50, return_sequences=True,
input_shape=(X_train.shape[1], X_train.shape[2])
        ),
        LSTM(50),
        Dense(1),
    ]
)
model.compile(optimizer="adam",
loss="mean_squared_error")

early_stopping = EarlyStopping(
    monitor="val_loss", patience=10,
restore_best_weights=True
)

history = model.fit(
    X_train,
    y_train,
    epochs=100,
    batch_size=32,
    validation_split=0.2,
    callbacks=[early_stopping],
)
```

```
X_test = train_scaled[-n_steps:].reshape(1,
n_steps, -1)
for _ in range(30 * 24):
    y_pred_scaled = model.predict(X_test)
    X_test = np.append(X_test,
np.append(X_test[-1, 1:], y_pred_scaled[-1:],
axis=0).reshape(1, n_steps, -1), axis=0)
y_pred = scaler.inverse_transform(y_pred_scaled)

date_range = pd.date_range(start='2024-09-10
01:00:00', periods=30 * 24, freq='h')
plt.figure(figsize=(14,7))
plt.plot(date_range, y_pred)
plt.xlabel("Date")
plt.ylabel("IN DP")
plt.title("Strainer Inlet Predicted IN DP for
the Next 1 Month with LSTM Univariate")
plt.savefig('Strainer Univariate Next 1 Month')
plt.show()

correlation_matrix = df.corr()
correlation_threshold = 0.5
selected_features = correlation_matrix['IN
DP'].abs().sort_values(ascending=False)
selected_features = selected_features[
    selected_features > correlation_threshold
].index.tolist()
data = df[selected_features]
print(selected_features)

for duration in [3, 6, 12]:
    multidur = duration * 30 * 24
    train_data = data[:-multidur]
    test_data = data[-multidur:]

    scaler = MinMaxScaler()
```

```
    train_scaled =
scaler.fit_transform(train_data)
    test_scaled = scaler.transform(test_data)

    n_steps = 30
    X_train, y_train = [], []

    for i in range(n_steps, len(train_scaled)):
        X_train.append(train_scaled[i - n_steps
: i])
        y_train.append(train_scaled[i])

    X_train, y_train = np.array(X_train),
np.array(y_train)


    model = Sequential(
        [
            LSTM(
                50, return_sequences=True,
input_shape=(X_train.shape[1], X_train.shape[2])
            ),
            LSTM(50),
            Dense(len(selected_features)),
        ]
    )
    model.compile(optimizer="adam",
loss="mean_squared_error")

    early_stopping = EarlyStopping(
        monitor="val_loss", patience=10,
restore_best_weights=True
    )

    history = model.fit(
        X_train,
        y_train,
```

```
        epochs=100,
        batch_size=32,
        validation_split=0.2,
        callbacks=[early_stopping],
    )

    X_test, y_test = [], []
    for i in range(n_steps, len(test_scaled)):
        X_test.append(test_scaled[i - n_steps :
i])
        y_test.append(test_scaled[i])

    X_test, y_test = np.array(X_test),
np.array(y_test)

    y_pred_scaled = model.predict(X_test)
    y_pred =
scaler.inverse_transform(y_pred_scaled)[:, 0]
    y_actual =
scaler.inverse_transform(y_test)[:, 0]

    plt.figure(figsize=(14, 7))
    plt.plot(test_data.index[n_steps:],
y_actual, label="Actual IN DP", color="blue")
    plt.plot(
        test_data.index[n_steps:],
        y_pred,
        label="Predicted IN DP",
        color="red",
        linestyle="--",
    )
    plt.xlabel("Date")
    plt.ylabel("IN DP")
    plt.title(f"Strainer Inlet Predicted vs
Actual IN DP for the Last {duration} Months with
LSTM Multivariate")
    plt.legend()
```

```
    plt.savefig(f'Strainer Multivariate
{duration} Months')
    plt.show()

    mape =
mean_absolute_percentage_error(y_actual, y_pred)
    rmse = np.sqrt(mean_squared_error(y_actual,
y_pred))
    r2 = r2_score(y_actual, y_pred)

    print("MAPE:", mape)
    print("RMSE:", rmse)
    print("R²:", r2)

data = df.loc['2024-01-19 07:00:00':,
selected_features]
scaler = MinMaxScaler()
train_scaled = scaler.fit_transform(data)

n_steps = 30
X_train, y_train = [], []

for i in range(n_steps, len(train_scaled)):
    X_train.append(train_scaled[i - n_steps :
i])
    y_train.append(train_scaled[i])

X_train, y_train = np.array(X_train),
np.array(y_train)

model = Sequential(
    [
        LSTM(
            50, return_sequences=True,
input_shape=(X_train.shape[1], X_train.shape[2])
        ),
        LSTM(50),
```

```
        Dense(len(selected_features)),
    ]
)
model.compile(optimizer="adam",
loss="mean_squared_error")

early_stopping = EarlyStopping(
    monitor="val_loss", patience=10,
restore_best_weights=True
)

history = model.fit(
    X_train,
    y_train,
    epochs=100,
    batch_size=32,
    validation_split=0.2,
    callbacks=[early_stopping],
)

X_test = train_scaled[-n_steps:].reshape(1,
n_steps, -1)
for _ in range(30 * 24):
    y_pred_scaled = model.predict(X_test)
    X_test = np.append(X_test,
np.append(X_test[-1, 1:], y_pred_scaled[-1:],
axis=0).reshape(1, n_steps, -1), axis=0)
y_pred =
scaler.inverse_transform(y_pred_scaled)[:, 0]

plt.figure(figsize=(14,7))
plt.plot(date_range, y_pred)
plt.xlabel("Date")
plt.ylabel("IN DP")
plt.title("Strainer Inlet Predicted IN DP for
the Next 1 Month with LSTM Multivariate")
```

```
plt.savefig('Strainer Multivariate Next 1
Month')
plt.show()

lagged = df[['IN DP']]
lagged['lag'] = lagged['IN DP'].shift(1)
lagged = lagged.dropna()

for duration in [3, 6, 12]:
    multidur = duration * 30 * 24
    arima = pm.auto_arima(lagged.iloc[:-
multidur, 0], lagged.iloc[:-multidur, 1:],
test='adf', trace=True)
    y_pred = arima.predict(multidur,
lagged.iloc[-multidur:, 1:])

    plt.figure(figsize=(14, 7))
    plt.plot(lagged.index[-multidur:],
lagged.iloc[-multidur:, 0], label="Actual IN
DP", color="blue")
    plt.plot(
        lagged.index[-multidur:],
        y_pred,
        label="Predicted IN DP",
        color="red",
        linestyle="--",
    )
    plt.xlabel("Date")
    plt.ylabel("IN DP")
    plt.title(f"Strainer Inlet Predicted vs
Actual IN DP for the Last {duration} Months with
ARIMA")
    plt.legend()
    plt.savefig(f'Strainer ARIMA {duration}
Months')
    plt.show()
```

```
    mape =
mean_absolute_percentage_error(lagged.iloc[-
multidur:, 0], y_pred)
    rmse =
np.sqrt(mean_squared_error(lagged.iloc[-
multidur:, 0], y_pred))
    r2 = r2_score(lagged.iloc[-multidur:, 0],
y_pred)

    print("MAPE:", mape)
    print("RMSE:", rmse)
    print("R²:", r2)

lagged = df[['IN DP']]
lagged['lag'] = lagged['IN DP'].shift(1)
lagged = lagged.dropna()

for duration in [3, 6, 12]:
    multidur = duration * 30 * 24
    arima = pm.auto_arima(lagged.iloc[:-
multidur, 0], lagged.iloc[:-multidur, 1:],
test='adf', trace=True)
    y_pred = arima.predict(multidur,
lagged.iloc[-multidur:, 1:])

    plt.figure(figsize=(14, 7))
    plt.plot(lagged.index[-multidur:],
lagged.iloc[-multidur:, 0], label="Actual IN
DP", color="blue")
    plt.plot(
        lagged.index[-multidur:],
        y_pred,
        label="Predicted IN DP",
        color="red",
        linestyle="--",
    )
    plt.xlabel("Date")
```

```
    plt.ylabel("IN DP")
    plt.title(f"Strainer Inlet Predicted vs
Actual IN DP for the Last {duration} Months with
ARIMA")
    plt.legend()
    plt.savefig(f'Strainer ARIMA {duration}
Months')
    plt.show()

    mape =
mean_absolute_percentage_error(lagged.iloc[-
multidur:, 0], y_pred)
    rmse =
np.sqrt(mean_squared_error(lagged.iloc[-
multidur:, 0], y_pred))
    r2 = r2_score(lagged.iloc[-multidur:, 0],
y_pred)

    print("MAPE:", mape)
    print("RMSE:", rmse)
    print("R²:", r2)

arima = pm.auto_arima(df.loc['2024-01-19
07:00:00':, 'IN DP'], test='adf', trace=True)
y_pred = arima.predict(30 * 24)

plt.figure(figsize=(14,7))
plt.plot(date_range, y_pred)
plt.xlabel("Date")
plt.ylabel("IN DP")
plt.title("Strainer Inlet Predicted IN DP for
the Next 1 Month with ARIMA")
plt.savefig('Strainer ARIMA Next 1 Month')
plt.show()

from xgboost import XGBRegressor
from skforecast.direct import ForecasterDirect
```

```
forecaster =
ForecasterDirect(regressor=XGBRegressor(),
lags=30, steps=30*24)
forecaster.fit(df.loc['2024-01-19 07:00:00':,
'IN DP'])
y_pred = forecaster.predict()

plt.figure(figsize=(14,7))
plt.plot(date_range, y_pred)
plt.xlabel("Date")
plt.ylabel("IN DP")
plt.title("Strainer Inlet Predicted IN DP for
the Next 1 Month with XGBoost")
plt.show()

from sklearn.linear_model import
LinearRegression
model = LinearRegression()
X = df.index.map(lambda x: x.toordinal() +
x.hour / 24).to_numpy().reshape(-1, 1)
model.fit(X, df['IN DP'])
y_pred =
model.predict(date_range.to_series().apply(lambd
a x: x.toordinal() + x.hour /
24).to_numpy().reshape(-1, 1))

plt.figure(figsize=(14,7))
plt.plot(date_range, y_pred)
plt.xlabel("Date")
plt.ylabel("IN DP")
plt.title("Strainer Inlet Predicted IN DP for
the Next 1 Month with Linear Regression")
plt.show()
```

Kode Sumber Particle Filter

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import pmdarima as pm
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import
mean_absolute_percentage_error,
mean_squared_error, r2_score
from keras.api.models import Sequential
from keras.api.layers import Dense, LSTM
from keras.api.callbacks import EarlyStopping

df = pd.read_excel('Particle Filter.xlsx', [1,
2], header=None, index_col=0,
usecols=range(1,17), skiprows=range(5))
df[1].index = pd.date_range('2023-01-01
01:00:00', '2023-12-31', freq='h')
df = pd.concat([df[1], df[2]])
df = df.rename(columns=dict(zip(range(2,17),
['MON DP', 'OUT PRESS', 'OUT TEMP', 'MON DP A',
'MON DP B', 'MON DP C', 'CONT TEMP', 'IN PRESS',
'CORRECTED VOLUME 2', 'CORRECTED VOLUME 1', 'IN
FLOW A', 'IN FLOW B', 'IN FLOW C', 'IN FLOW D',
'IN FLOW E'])))
df.index.name = 'Datetime'

for i in range(1, len(df)-1):
    if (df.iloc[i-1, 8] == 0) & (df.iloc[i+1, 8]
== 0):
        df.iloc[i, 8] = 0
    if (df.iloc[i-1, 9] == 0) & (df.iloc[i+1, 9]
== 0):
        df.iloc[i, 9] = 0
df['CORRECTED VOLUME 1'] = df['CORRECTED VOLUME
1'] + df['CORRECTED VOLUME 2']
```

```
df = df.drop('CORRECTED VOLUME 2', axis=1)
df = df.rename(columns={'CORRECTED VOLUME 1':
'CORRECTED VOLUME'})

for target in ['MON DP', 'MON DP A', 'MON DP B',
'MON DP C']:
    data = df[target]

    for duration in [3, 6, 12]:
        multidur = duration * 30 * 24
        train_data = data[:-multidur]
        test_data = data[-multidur:]

        scaler = MinMaxScaler()
        train_scaled =
scaler.fit_transform(train_data.to_frame())
        test_scaled =
scaler.transform(test_data.to_frame())

        n_steps = 30
        X_train, y_train = [], []

        for i in range(n_steps,
len(train_scaled)):
            X_train.append(train_scaled[i -
n_steps : i])
            y_train.append(train_scaled[i])

        X_train, y_train = np.array(X_train),
np.array(y_train)


        model = Sequential(
            [
                LSTM(
                    50, return_sequences=True,
input_shape=(X_train.shape[1], X_train.shape[2])
```

```
                ),
                LSTM(50),
                Dense(1),
            ]
        )
        model.compile(optimizer="adam",
loss="mean_squared_error")

        early_stopping = EarlyStopping(
            monitor="val_loss", patience=10,
restore_best_weights=True
        )

        history = model.fit(
            X_train,
            y_train,
            epochs=100,
            batch_size=32,
            validation_split=0.2,
            callbacks=[early_stopping],
        )

        X_test, y_test = [], []
        for i in range(n_steps,
len(test_scaled)):
            X_test.append(test_scaled[i -
n_steps : i])
            y_test.append(test_scaled[i])

        X_test, y_test = np.array(X_test),
np.array(y_test)

        y_pred_scaled = model.predict(X_test)
        y_pred =
scaler.inverse_transform(y_pred_scaled)
        y_actual =
scaler.inverse_transform(y_test)
```

```
        plt.figure(figsize=(14, 7))
        plt.plot(test_data.index[n_steps:],
y_actual, label=f"Actual {target}",
color="blue")
        plt.plot(
            test_data.index[n_steps:],
            y_pred,
            label=f"Predicted {target}",
            color="red",
            linestyle="--",
        )
        plt.xlabel("Date")
        plt.ylabel(f"{target}")
        plt.title(f"Particle Filter Predicted vs
Actual {target} for the Last {duration} Months
with LSTM Univariate")
        plt.legend()
        plt.savefig(f'Particle {target}
Univariate {duration} Months')
        plt.show()

        mape =
mean_absolute_percentage_error(y_actual, y_pred)
        rmse =
np.sqrt(mean_squared_error(y_actual, y_pred))
        r2 = r2_score(y_actual, y_pred)

        print("MAPE:", mape)
        print("RMSE:", rmse)
        print("R²:", r2)

correlation_matrix = df.corr()
correlation_threshold = 0.5

for target in ['MON DP', 'MON DP A', 'MON DP B',
'MON DP C']:
```

```
    selected_features =
correlation_matrix[target].abs().sort_values(asc
ending=False)
    selected_features = selected_features[
        selected_features >
correlation_threshold
    ].index.tolist()
    data = df[selected_features]
    target_index = data.columns.get_loc(target)

    for duration in [3, 6, 12]:
        multidur = duration * 30 * 24
        train_data = data[:-multidur]
        test_data = data[-multidur:]

        scaler = MinMaxScaler()
        train_scaled =
scaler.fit_transform(train_data)
        test_scaled =
scaler.transform(test_data)

        n_steps = 30
        X_train, y_train = [], []

        for i in range(n_steps,
len(train_scaled)):
            X_train.append(train_scaled[i -
n_steps : i])
            y_train.append(train_scaled[i])

        X_train, y_train = np.array(X_train),
np.array(y_train)


        model = Sequential(
            [
                LSTM(
```

```
                    50, return_sequences=True,
input_shape=(X_train.shape[1], X_train.shape[2])
                ),
                LSTM(50),
                Dense(len(selected_features)),
            ]
        )
        model.compile(optimizer="adam",
loss="mean_squared_error")

        early_stopping = EarlyStopping(
            monitor="val_loss", patience=10,
restore_best_weights=True
        )

        history = model.fit(
            X_train,
            y_train,
            epochs=100,
            batch_size=32,
            validation_split=0.2,
            callbacks=[early_stopping],
        )

        X_test, y_test = [], []
        for i in range(n_steps,
len(test_scaled)):
            X_test.append(test_scaled[i -
n_steps : i])
            y_test.append(test_scaled[i])

        X_test, y_test = np.array(X_test),
np.array(y_test)

        y_pred_scaled = model.predict(X_test)
```

```
        y_pred =
scaler.inverse_transform(y_pred_scaled)[:,
target_index]
        y_actual =
scaler.inverse_transform(y_test)[:,
target_index]

        plt.figure(figsize=(14, 7))
        plt.plot(test_data.index[n_steps:],
y_actual, label=f"Actual {target}",
color="blue")
        plt.plot(
            test_data.index[n_steps:],
            y_pred,
            label=f"Predicted {target}",
            color="red",
            linestyle="--",
        )
        plt.xlabel("Date")
        plt.ylabel(f"{target}")
        plt.title(f"Particle Filter Predicted vs
Actual {target} for the Last {duration} Months
with LSTM Multivariate")
        plt.legend()
        plt.savefig(f'Particle {target}
Multivariate {duration} Months')
        plt.show()

        mape =
mean_absolute_percentage_error(y_actual, y_pred)
        rmse =
np.sqrt(mean_squared_error(y_actual, y_pred))
        r2 = r2_score(y_actual, y_pred)

        print("MAPE:", mape)
        print("RMSE:", rmse)
        print("R²:", r2)
```

```
for target in ['MON DP', 'MON DP A', 'MON DP B',
'MON DP C']:
    lagged = df[[target]]
    lagged['lag'] = lagged[target].shift(1)
    lagged = lagged.dropna()

    for duration in [3, 6, 12]:
        multidur = duration * 30 * 24
        arima = pm.auto_arima(lagged.iloc[:-
multidur, 0], lagged.iloc[:-multidur, 1:],
test='adf', trace=True)
        y_pred = arima.predict(multidur,
lagged.iloc[-multidur:, 1:])

        plt.figure(figsize=(14, 7))
        plt.plot(lagged.index[-multidur:],
lagged.iloc[-multidur:, 0], label=f"Actual
{target}", color="blue")
        plt.plot(
            lagged.index[-multidur:],
            y_pred,
            label=f"Predicted {target}",
            color="red",
            linestyle="--",
        )
        plt.xlabel("Date")
        plt.ylabel(f"{target}")
        plt.title(f"Particle Filter Predicted vs
Actual {target} for the Last {duration} Months
with ARIMA")
        plt.legend()
        plt.savefig(f'Particle {target} ARIMA
{duration} Months')
        plt.show()
```

```
        mape =
mean_absolute_percentage_error(lagged.iloc[-
multidur:, 0], y_pred)
        rmse =
np.sqrt(mean_squared_error(lagged.iloc[-
multidur:, 0], y_pred))
        r2 = r2_score(lagged.iloc[-multidur:,
0], y_pred)

        print("MAPE:", mape)
        print("RMSE:", rmse)
        print("R²:", r2)
```

Kode Sumber Gas Turbin Generator

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import pmdarima as pm
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import
mean_absolute_percentage_error,
mean_squared_error, r2_score
from keras.api.models import Sequential
from keras.api.layers import Dense, LSTM
from keras.api.callbacks import EarlyStopping

skiprows = [i for i in range(8)]
skiprows.extend([i for i in range(24, 34)])
skiprows.append(36)

df = pd.read_excel('Gas Turbin Generator.xlsx',
header=None, skiprows=skiprows)
df = df.drop([0, 2, 3, 4, 5], axis=1)
df = df.T
df = df.rename(df.iloc[0], axis=1)
df = df.drop(1)
df.columns = [col.strip() for col in df.columns]
df = df.reset_index(drop=True)
df = df.drop(range(1, 680, 2))
df.index = pd.date_range('2023-05-12 06:00:00',
'2024-09-20 18:00:00', freq='12h')
df = df.dropna()
df = df[~df.apply(lambda row: all(cell in
{'(NS)', 'SB', 'BD', 'RUN'} for cell in row),
axis=1)]
df.loc['2024-06-25 06:00:00', 'Turbine Air Inlet
Filter Differential Pressure'] = 0.3

for col in df.columns:
```

```
    df[col] = df[col].astype(float)

df.loc[df['Enclosure Vent Filter-A Current
Ampere'] < 20, 'Enclosure Vent Filter-A Current
Ampere'] = df['Enclosure Vent Filter-B Current
Ampere']
df = df.drop('Enclosure Vent Filter-B Current
Ampere', axis=1)
df = df.rename(columns={'Enclosure Vent Filter-A
Current Ampere': 'Enclosure Vent Filter Current
Ampere'})

df = df.resample('12h', offset='6h').ffill()
df.index.name = 'Datetime'

target = 'Pressure Compressor Discharge (PCD)'
data = df[target]

for duration in [3, 6, 12]:
    multidur = duration * 30 * 2
    train_data = data[:-multidur]
    test_data = data[-multidur:]

    scaler = MinMaxScaler()
    train_scaled =
scaler.fit_transform(train_data.to_frame())
    test_scaled =
scaler.transform(test_data.to_frame())

    n_steps = 30
    X_train, y_train = [], []

    for i in range(n_steps, len(train_scaled)):
        X_train.append(train_scaled[i - n_steps
: i])
        y_train.append(train_scaled[i])
```

```
    X_train, y_train = np.array(X_train),
np.array(y_train)


    model = Sequential(
        [
            LSTM(
                50, return_sequences=True,
input_shape=(X_train.shape[1], X_train.shape[2])
            ),
            LSTM(50),
            Dense(1),
        ]
    )
    model.compile(optimizer="adam",
loss="mean_squared_error")

    early_stopping = EarlyStopping(
        monitor="val_loss", patience=10,
restore_best_weights=True
    )

    history = model.fit(
        X_train,
        y_train,
        epochs=100,
        batch_size=32,
        validation_split=0.2,
        callbacks=[early_stopping],
    )

    X_test, y_test = [], []
    for i in range(n_steps, len(test_scaled)):
        X_test.append(test_scaled[i - n_steps :
i])
        y_test.append(test_scaled[i])
```

```
    X_test, y_test = np.array(X_test),
np.array(y_test)

    y_pred_scaled = model.predict(X_test)
    y_pred =
scaler.inverse_transform(y_pred_scaled)
    y_actual = scaler.inverse_transform(y_test)

    plt.figure(figsize=(14, 7))
    plt.plot(test_data.index[n_steps:],
y_actual, label="Actual PCD", color="blue")
    plt.plot(
        test_data.index[n_steps:],
        y_pred,
        label="Predicted PCD",
        color="red",
        linestyle="--",
    )
    plt.xlabel("Date")
    plt.ylabel("PCD")
    plt.title(f"Gas Turbine Generator Predicted
vs Actual PCD for the Last {duration} Months
with LSTM Univariate")
    plt.legend()
    plt.savefig(f'GTG Univariate {duration}
Months')
    plt.show()

    mape =
mean_absolute_percentage_error(y_actual, y_pred)
    rmse = np.sqrt(mean_squared_error(y_actual,
y_pred))
    r2 = r2_score(y_actual, y_pred)

    print("MAPE:", mape)
    print("RMSE:", rmse)
    print("R²:", r2)
```

```
scaler = MinMaxScaler()
train_scaled =
scaler.fit_transform(data.to_frame())

n_steps = 30
X_train, y_train = [], []

for i in range(n_steps, len(train_scaled)):
    X_train.append(train_scaled[i - n_steps :
i])
    y_train.append(train_scaled[i])

X_train, y_train = np.array(X_train),
np.array(y_train)

model = Sequential(
    [
        LSTM(
            50, return_sequences=True,
input_shape=(X_train.shape[1], X_train.shape[2])
        ),
        LSTM(50),
        Dense(1),
    ]
)
model.compile(optimizer="adam",
loss="mean_squared_error")

early_stopping = EarlyStopping(
    monitor="val_loss", patience=10,
restore_best_weights=True
)

history = model.fit(
    X_train,
    y_train,
```

```
    epochs=100,
    batch_size=32,
    validation_split=0.2,
    callbacks=[early_stopping],
)

X_test = train_scaled[-n_steps:].reshape(1,
n_steps, -1)
for _ in range(30 * 2):
    y_pred_scaled = model.predict(X_test)
    X_test = np.append(X_test,
np.append(X_test[-1, 1:], y_pred_scaled[-1:],
axis=0).reshape(1, n_steps, -1), axis=0)
y_pred = scaler.inverse_transform(y_pred_scaled)

actual_data = pd.read_excel('GTG New.xlsx', 1,
usecols=[1, 2, 5])
actual_data['Datetime'] =
pd.to_datetime(actual_data['Date'].dt.strftime('
%Y-%m-%d') + ' ' + actual_data['Time'])
actual_data.set_index('Datetime', inplace=True)
actual_data = actual_data.loc['2024-08-23
06:00:00':'2024-09-21 18:00:00',
'analog.an_engine_pcd']

date_range = pd.date_range(start='2024-08-23
06:00:00', periods=30 * 2, freq='12h')
plt.figure(figsize=(14,7))
plt.plot(actual_data, label='Actual PCD',
color='blue')
plt.plot(date_range, y_pred, label='Predicted
PCD', color='red', linestyle='--')
plt.xlabel("Date")
plt.ylabel("PCD")
plt.legend()
plt.title("Gas Turbine Generator Predicted PCD
for the Next 1 Month with LSTM Univariate")
```

```
plt.savefig('GTG Univariate Next 1 Month')
plt.show()

selected_features = ['Real Power', 'Engine T5
Average', 'Engine Air Inlet Temperature (T1)',
'Turbine Air Inlet Filter Differential
Pressure', 'Pressure Compressor Discharge
(PCD)', 'Gas Fuel Pressure from HP Gas Fuel
Header', 'Gas Fuel Temperature from HP Gas Fuel
Header', 'Gas Fuel Flow Rate from HP Gas Fuel
Header']
data = df[selected_features]
target_index = data.columns.get_loc(target)

for duration in [3, 6, 12]:
    multidur = duration * 30 * 2
    train_data = data[:-multidur]
    test_data = data[-multidur:]

    scaler = MinMaxScaler()
    train_scaled =
scaler.fit_transform(train_data)
    test_scaled = scaler.transform(test_data)

    n_steps = 30
    X_train, y_train = [], []

    for i in range(n_steps, len(train_scaled)):
        X_train.append(train_scaled[i - n_steps
: i])
        y_train.append(train_scaled[i])

    X_train, y_train = np.array(X_train),
np.array(y_train)


    model = Sequential(
```

```
        [
            LSTM(
                50, return_sequences=True,
input_shape=(X_train.shape[1], X_train.shape[2])
            ),
            LSTM(50),
            Dense(len(selected_features)),
        ]
    )
    model.compile(optimizer="adam",
loss="mean_squared_error")

    early_stopping = EarlyStopping(
        monitor="val_loss", patience=10,
restore_best_weights=True
    )

    history = model.fit(
        X_train,
        y_train,
        epochs=100,
        batch_size=32,
        validation_split=0.2,
        callbacks=[early_stopping],
    )

    X_test, y_test = [], []
    for i in range(n_steps, len(test_scaled)):
        X_test.append(test_scaled[i - n_steps :
i])
        y_test.append(test_scaled[i])

    X_test, y_test = np.array(X_test),
np.array(y_test)

    y_pred_scaled = model.predict(X_test)
```

```
    y_pred =
scaler.inverse_transform(y_pred_scaled)[:,
target_index]
    y_actual =
scaler.inverse_transform(y_test)[:,
target_index]

    plt.figure(figsize=(14, 7))
    plt.plot(test_data.index[n_steps:],
y_actual, label="Actual PCD", color="blue")
    plt.plot(
        test_data.index[n_steps:],
        y_pred,
        label="Predicted PCD",
        color="red",
        linestyle="--",
    )
    plt.xlabel("Date")
    plt.ylabel("PCD")
    plt.title(f"Gas Turbine Generator Predicted
vs Actual PCD for the Last {duration} Months
with LSTM Multivariate")
    plt.legend()
    plt.savefig(f'GTG Multivariate {duration}
Months')
    plt.show()

    mape =
mean_absolute_percentage_error(y_actual, y_pred)
    rmse = np.sqrt(mean_squared_error(y_actual,
y_pred))
    r2 = r2_score(y_actual, y_pred)

    print("MAPE:", mape)
    print("RMSE:", rmse)
    print("R²:", r2)
```

```
scaler = MinMaxScaler()
train_scaled = scaler.fit_transform(data)

n_steps = 30
X_train, y_train = [], []

for i in range(n_steps, len(train_scaled)):
    X_train.append(train_scaled[i - n_steps :
i])
    y_train.append(train_scaled[i])

X_train, y_train = np.array(X_train),
np.array(y_train)

model = Sequential(
    [
        LSTM(
            50, return_sequences=True,
input_shape=(X_train.shape[1], X_train.shape[2])
        ),
        LSTM(50),
        Dense(len(selected_features)),
    ]
)
model.compile(optimizer="adam",
loss="mean_squared_error")

early_stopping = EarlyStopping(
    monitor="val_loss", patience=10,
restore_best_weights=True
)

history = model.fit(
    X_train,
    y_train,
    epochs=100,
    batch_size=32,
```

```
    validation_split=0.2,
    callbacks=[early_stopping],
)

X_test = train_scaled[-n_steps:].reshape(1,
n_steps, -1)
for _ in range(30 * 2):
    y_pred_scaled = model.predict(X_test)
    X_test = np.append(X_test,
np.append(X_test[-1, 1:], y_pred_scaled[-1:],
axis=0).reshape(1, n_steps, -1), axis=0)
y_pred =
scaler.inverse_transform(y_pred_scaled)[:,
target_index]

plt.figure(figsize=(14,7))
plt.plot(actual_data, label='Actual PCD',
color='blue')
plt.plot(date_range, y_pred, label='Predicted
PCD', color='red', linestyle='--')
plt.xlabel("Date")
plt.ylabel("PCD")
plt.legend()
plt.title("Gas Turbine Generator Predicted PCD
for the Next 1 Month with LSTM Multivariate")
plt.savefig('GTG Multivariate Next 1 Month')
plt.show()

lagged = df[[target]]
lagged['lag'] = lagged[target].shift(1)
lagged = lagged.dropna()

for duration in [3, 6, 12]:
    multidur = duration * 30 * 2
    arima = pm.auto_arima(lagged.iloc[:-
multidur, 0], lagged.iloc[:-multidur, 1:],
test='adf', trace=True)
```

```
    y_pred = arima.predict(multidur,
lagged.iloc[-multidur:, 1:])

    plt.figure(figsize=(14, 7))
    plt.plot(lagged.index[-multidur:],
lagged.iloc[-multidur:, 0], label="Actual PCD",
color="blue")
    plt.plot(
        lagged.index[-multidur:],
        y_pred,
        label="Predicted PCD",
        color="red",
        linestyle="--",
    )
    plt.xlabel("Date")
    plt.ylabel("IN DP")
    plt.title(f"Gas Turbine Generator Predicted
vs Actual PCD for the Last {duration} Months
with ARIMA")
    plt.legend()
    plt.savefig(f'GTG ARIMA {duration} Months')
    plt.show()

    mape =
mean_absolute_percentage_error(lagged.iloc[-
multidur:, 0], y_pred)
    rmse =
np.sqrt(mean_squared_error(lagged.iloc[-
multidur:, 0], y_pred))
    r2 = r2_score(lagged.iloc[-multidur:, 0],
y_pred)

    print("MAPE:", mape)
    print("RMSE:", rmse)
    print("R²:", r2)
```

```
arima = pm.auto_arima(df[target], test='adf',
trace=True)
y_pred = arima.predict(30 * 2)

plt.figure(figsize=(14,7))
plt.plot(actual_data, label='Actual PCD',
color='blue')
plt.plot(date_range, y_pred, label='Predicted
PCD', color='red', linestyle='--')
plt.xlabel("Date")
plt.ylabel("PCD")
plt.legend()
plt.title("Gas Turbine Generator Predicted PCD
for the Next 1 Month with ARIMA")
plt.savefig('GTG ARIMA Next 1 Month')
plt.show()

from xgboost import XGBRegressor
from skforecast.direct import ForecasterDirect

forecaster =
ForecasterDirect(regressor=XGBRegressor(),
lags=30, steps=30*2)
forecaster.fit(df[target])
y_pred = forecaster.predict()

plt.figure(figsize=(14,7))
plt.plot(actual_data, label='Actual PCD',
color='blue')
plt.plot(date_range, y_pred, label='Predicted
PCD', color='red', linestyle='--')
plt.xlabel("Date")
plt.ylabel("PCD")
plt.legend()
plt.title("Gas Turbine Generator Predicted PCD
for the Next 1 Month with XGBoost")
plt.show()
```

```
from sklearn.linear_model import
LinearRegression
model = LinearRegression()
X = df.index.map(lambda x: x.toordinal() +
x.hour / 24).to_numpy().reshape(-1, 1)
model.fit(X, df[target])
y_pred =
model.predict(date_range.to_series().apply(lambd
a x: x.toordinal() + x.hour /
24).to_numpy().reshape(-1, 1))

plt.figure(figsize=(14,7))
plt.plot(actual_data, label='Actual PCD',
color='blue')
plt.plot(date_range, y_pred, label='Predicted
PCD', color='red', linestyle='--')
plt.xlabel("Date")
plt.ylabel("PCD")
plt.legend()
plt.title("Gas Turbine Generator Predicted PCD
for the Next 1 Month with Linear Regression")
plt.show()
```