

# **Kinerja Model Regresi Linear dan Polinomial dalam Memprediksi Konsumsi Daya Rumah Tangga**

Disusun untuk memenuhi tugas 2 mata kuliah Pembelajaran Mesin

Oleh:

<b>Willy Jonathan Arsyad</b>	<b>(2208107010037)</b>
<b>Agil Mughni</b>	<b>(2208107010025)</b>
<b>Alfi Zamriza</b>	<b>(2208107010080)</b>
<b>T.M Fadlul Ihsan</b>	<b>(2208107010088)</b>
<b>M. Arkan Haris</b>	<b>(2208107010022)</b>



**JURUSAN INFORMATIKA  
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM  
UNIVERSITAS SYIAH KUALA  
DARUSSALAM, BANDA ACEH  
2025**

## 1. Pemahaman Dataset

### 1.1 Sumber Data

Dataset yang digunakan dalam proyek ini adalah "*Individual household electric power consumption Data Set*", yang tersedia secara publik melalui UCI Machine Learning Repository pada tautan berikut:

<https://archive.ics.uci.edu/dataset/235/individual+household+electric+power+consumption>

Dataset ini mencatat konsumsi listrik dari satu rumah tangga di Prancis selama periode Desember 2006 hingga November 2010. Data dikumpulkan dengan interval waktu setiap satu menit.

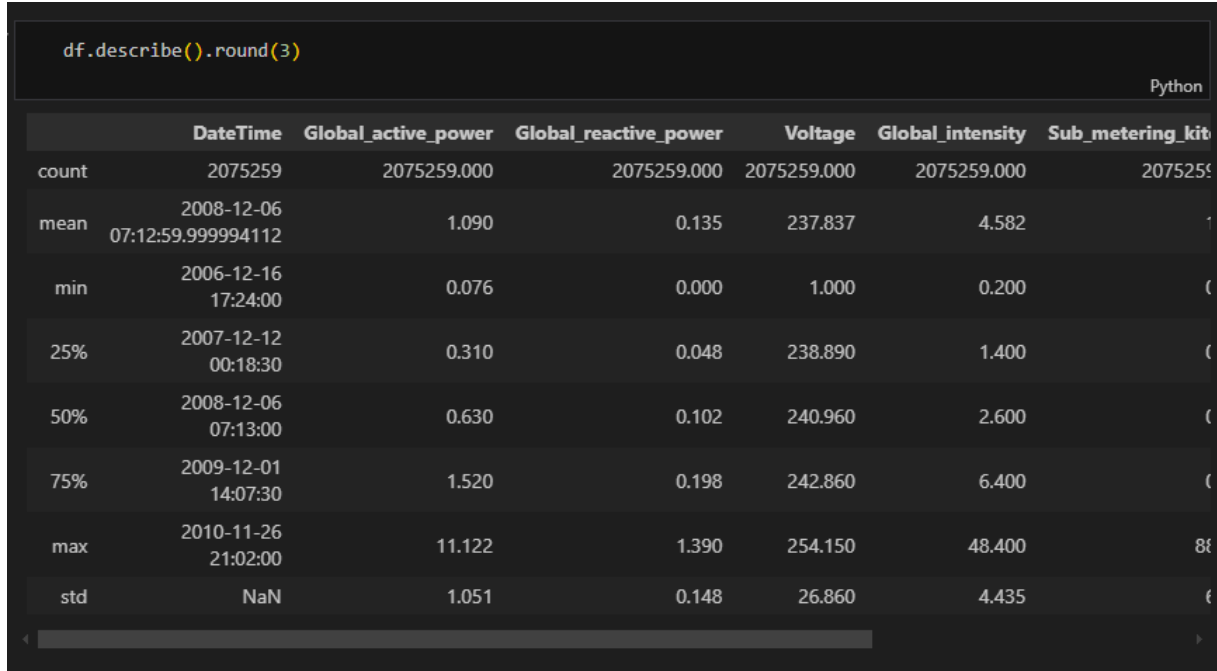
### 1.2 Deskripsi Variabel

Beberapa variabel penting yang digunakan dalam analisis ini antara lain:

Nama Variabel	Deskripsi
Date	Tanggal Pencatatan (format: dd/mm/yyyy)
Time	Waktu pencatatan (format: hh:mm:ss)
Global_active_power	Konsumsi daya aktif global (kilowatt) - digunakan sebagai target regresi
Global_reactive_power	Konsumsi daya reaktif global (kilowatt)
Voltage	Tegangan (volt)
Global_intensity	Intensitas arus global (ampere)
Sub_metering_1	Energi dalam wattt-hour untuk area dapur
Sub_metering_2	Energi dalam wattt-hour untuk laundry room
Sub_metering_3	Energi dalam wattt-hour untuk pemanasan air dan AC

### 1.3 Statistika Deskriptif

Statistik deskriptif dari data numerik dapat ditampilkan menggunakan fungsi `.describe()`:



```
df.describe().round(3)
```

	DateTime	Global_active_power	Global_reactive_power	Voltage	Global_intensity	Sub_metering_kit
count	2075259	2075259.000	2075259.000	2075259.000	2075259.000	2075259.000
mean	2008-12-06 07:12:59.999994112	1.090	0.135	237.837	4.582	1.090
min	2006-12-16 17:24:00	0.076	0.000	1.000	0.200	0.076
25%	2007-12-12 00:18:30	0.310	0.048	238.890	1.400	0.310
50%	2008-12-06 07:13:00	0.630	0.102	240.960	2.600	0.630
75%	2009-12-01 14:07:30	1.520	0.198	242.860	6.400	1.520
max	2010-11-26 21:02:00	11.122	1.390	254.150	48.400	8.122
std	NaN	1.051	0.148	26.860	4.435	1.051

Insight Awal:

- Rata-rata konsumsi daya aktif sekitar **1.09 kW**.
- Tegangan listrik bervariasi di sekitar **240 Volt**.
- Terdapat outlier yang bisa mempengaruhi model prediksi.

## 2. Eksplorasi Data dan Pra-pemrosesan

### 2.1 Menggabungkan Kolom Date dan Time menjadi 1 Kolom (DateTime)

```
[ ] # combine both Date and Time into 1 col
df['Date'] = df['Date'].astype(str) + ' ' + df['Time']
df = df.rename({'Date': 'DateTime'}, axis=1) # rename column into DateTime
df['DateTime'] = pd.to_datetime(df['DateTime'], format='%d/%m/%Y %H:%M:%S', dayfirst=True)
```

Menggabungkan dua kolom yang terpisah yaitu Date (tanggal) dan Time (waktu) menjadi satu kolom DateTime. Hal ini dilakukan agar dapat lebih mudah dalam manipulasi dan analisis data berbasis waktu. Kolom DateTime kemudian diubah menjadi tipe data datetime dengan format yang sesuai (%d/%m/%Y %H:%M:%S). Manfaat penggabungan ini memungkinkan analisis berbasis waktu (misalnya, analisis tren harian, mingguan, atau bulanan) lebih mudah dilakukan.

## 2.2 Menghapus Kolom Time yang Tidak Digunakan

```
# Remove unused Time column
df = df.drop(['Time'], axis=1)
```

Kolom Time sudah tidak diperlukan lagi karena informasi waktu sudah tercakup dalam kolom DateTime. Menghapus kolom Time mengurangi redundansi dan mempermudah analisis.

## 2.3 Mengubah Tipe Data Kolom Lainnya menjadi Numeric

```
# Change other column dtype
for col in df.columns:
    if col != 'DateTime':
        df[col] = pd.to_numeric(df[col], errors='coerce')
```

Mengubah tipe data setiap kolom lainnya menjadi numerik (kecuali DateTime). `errors='coerce'` berarti bahwa jika ada nilai yang tidak dapat diubah menjadi numerik, nilai tersebut akan diubah menjadi NaN. Ini berguna untuk menangani nilai-nilai yang tidak valid atau format yang tidak sesuai. Manfaat memastikan bahwa semua kolom yang berisi data numerik diproses dengan benar, sehingga analisis statistik dan pemodelan dapat dilakukan dengan akurat.

## 2.4 Mengganti Nama Kolom Sub\_metering\_1 dan Sub\_metering\_2

```
[ ] df = df.rename({
    'Sub_metering_1': 'Sub_metering_kitchen',
    'Sub_metering_2': 'Sub_metering_laundry'}, axis=1)
```

Mengubah nama kolom `Sub_metering_1` menjadi `Sub_metering_kitchen` dan `Sub_metering_2` menjadi `Sub_metering_laundry` untuk memberikan makna yang lebih jelas mengenai penggunaan energi dalam dataset. **Manfaat** nama yang lebih deskriptif memudahkan pemahaman data dan interpretasi hasil analisis.

## 2.5 Memeriksa Ukuran Dataset dan Mengecek Nilai Null

- a. **df.shape**: Menampilkan jumlah baris dan kolom dalam dataset. Ini memberikan gambaran tentang ukuran dataset.

```
[ ] df.shape
(2075259, 8)
```

- b. **df.isna().sum()**: Memeriksa jumlah nilai NaN atau kosong dalam setiap kolom. Ini membantu mengetahui apakah masih ada missing values yang perlu diatasi.

```
[ ] df.isna().sum()

0
DateTime      0
Global_active_power  25979
Global_reactive_power  25979
Voltage        25979
Global_intensity  25979
Sub_metering_kitchen  25979
Sub_metering_laundry  25979
Sub_metering_3      25979

dtype: int64
```

- c. **df.duplicated().sum()**: Memeriksa apakah ada duplikat dalam dataset yang perlu dihapus untuk menjaga kualitas data.

```
[ ] df.duplicated().sum()

np.int64(0)
```

## 2.6 Time Series Plot

```
# Get numeric columns (excluding 'DateTime')
numeric_cols = df.select_dtypes(include=[np.number]).columns.tolist()

# Create subplots
fig, axes = plt.subplots(len(numeric_cols), 1, figsize=(15, 3*len(numeric_cols)))
fig.suptitle('Household Power Consumption Time Series Analysis', fontsize=16)

# Format the date on x-axis
date_format = mdates.DateFormatter('%d-%m-%Y')

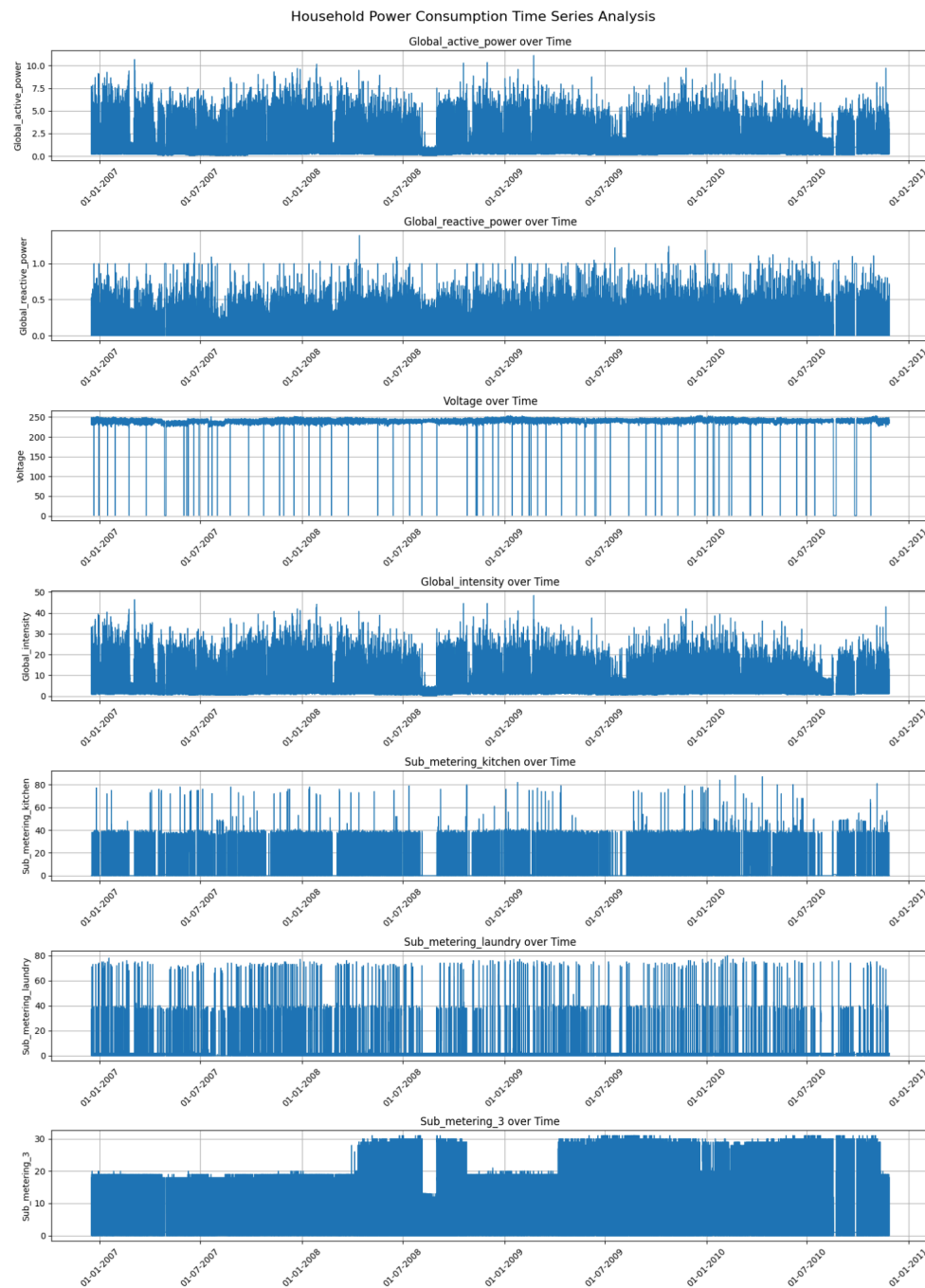
# Ensure axes is iterable (even if there's only one plot)
if len(numeric_cols) == 1:
    axes = [axes]

# Plot each numeric column against the DateTime column
for i, col in enumerate(numeric_cols):
    axes[i].plot(df['DateTime'], df[col], linewidth=1)
    axes[i].set_title(f'{col} over Time')
    axes[i].set_ylabel(col)
    axes[i].grid(True)
    axes[i].xaxis.set_major_formatter(date_format)
    plt.setp(axes[i].xaxis.get_majorticklabels(), rotation=45)

plt.tight_layout()
plt.subplots_adjust(top=0.95)

# plt.savefig('time_series_plots.png', dpi=300)

plt.show()
```



Menganalisis **konsumsi daya rumah tangga** (dan variabel terkait) seiring waktu dalam bentuk **time series**. Membantu dalam memeriksa pola atau tren konsumsi daya yang terjadi sepanjang waktu.

## 2.7 Daily Patterns (Pola Harian)

```
# Add hour column
df_hourly = df.copy()
df_hourly['hour'] = df_hourly['DateTime'].dt.hour

# Group by hour and calculate mean
hourly_avg = df_hourly.groupby('hour').mean()

# Get numeric columns excluding hour
numeric_cols = df_hourly.select_dtypes(include=[np.number]).columns.tolist()
if 'hour' in numeric_cols:
    numeric_cols.remove('hour')

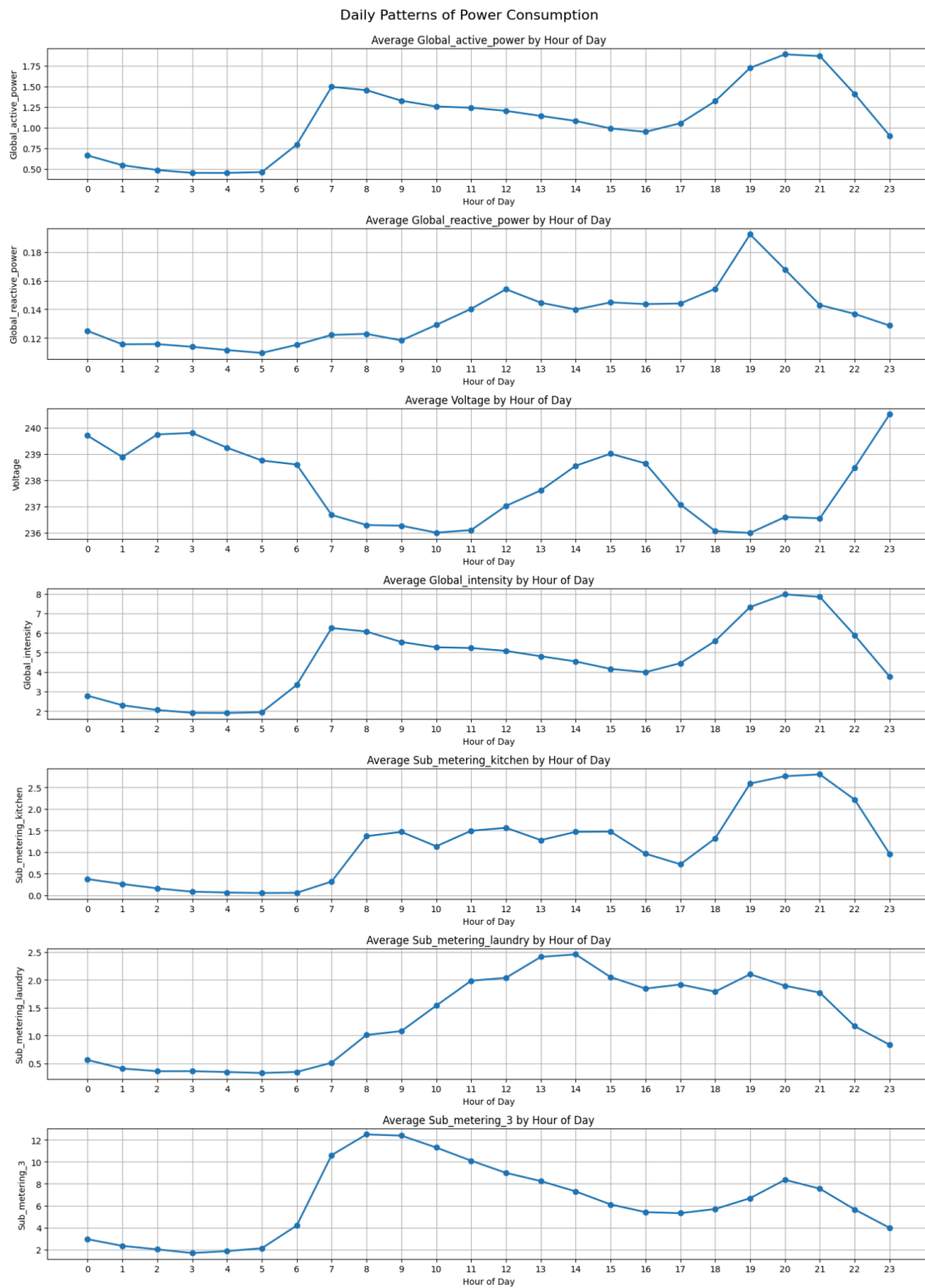
# Plot daily patterns
fig, axes = plt.subplots(len(numeric_cols), 1, figsize=(15, 3*len(numeric_cols)))
fig.suptitle('Daily Patterns of Power Consumption', fontsize=16)

for i, col in enumerate(numeric_cols):
    axes[i].plot(hourly_avg.index, hourly_avg[col], marker='o', linewidth=2)
    axes[i].set_title(f'Average {col} by Hour of Day')
    axes[i].set_xlabel('Hour of Day')
    axes[i].set_ylabel(col)
    axes[i].grid(True)
    axes[i].set_xticks(range(0, 24))

plt.tight_layout()
plt.subplots_adjust(top=0.95)

# plt.savefig('daily_patterns.png', dpi=300)

plt.show()
```



Menganalisis pola konsumsi daya rumah tangga berdasarkan waktu dalam sehari (per jam). Menemukan pola atau fluktuasi konsumsi daya yang terjadi pada setiap jam dalam sehari.



## 2.8 Sub Metering Comparison (Perbandingan Sub-metering)

```
# Create a figure
plt.figure(figsize=(15, 8))

# Plot sub-metering data
plt.plot(df['DateTime'], df['Sub_metering_kitchen'], label='Kitchen', alpha=0.7)
plt.plot(df['DateTime'], df['Sub_metering_laundry'], label='Laundry Room', alpha=0.7)
plt.plot(df['DateTime'], df['Sub_metering_3'], label='Water Heater & AC', alpha=0.7)

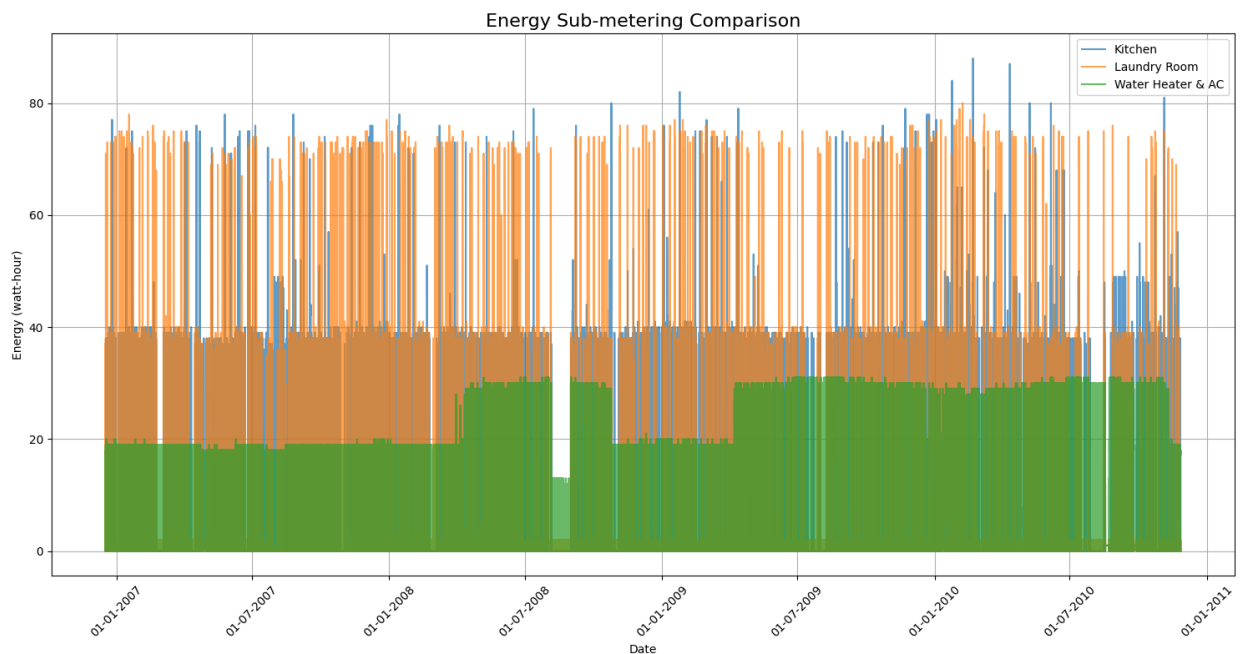
# Add labels and title
plt.title('Energy Sub-metering Comparison', fontsize=16)
plt.xlabel('Date')
plt.ylabel('Energy (watt-hour)')
plt.legend()
plt.grid(True)

# Format the date on x-axis
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%d-%m-%Y'))
plt.xticks(rotation=45)

plt.tight_layout()

# plt.savefig('sub_metering_comparison.png', dpi=300)

plt.show()
```



Membandingkan konsumsi energi untuk beberapa area rumah tangga, yang tercatat dalam sub-metering: dapur, ruang cuci, dan pemanas air/AC. Membantu memahami dimana sebagian besar konsumsi energi terjadi di rumah tangga.

## 2.9 Correlation Heatmap (Peta Panas Korelasi)

```
[ ] # Select numeric columns
numeric_df = df.select_dtypes(include=[np.number])

# Compute correlation matrix
corr_matrix = numeric_df.corr()

# Create a figure
plt.figure(figsize=(12, 10))

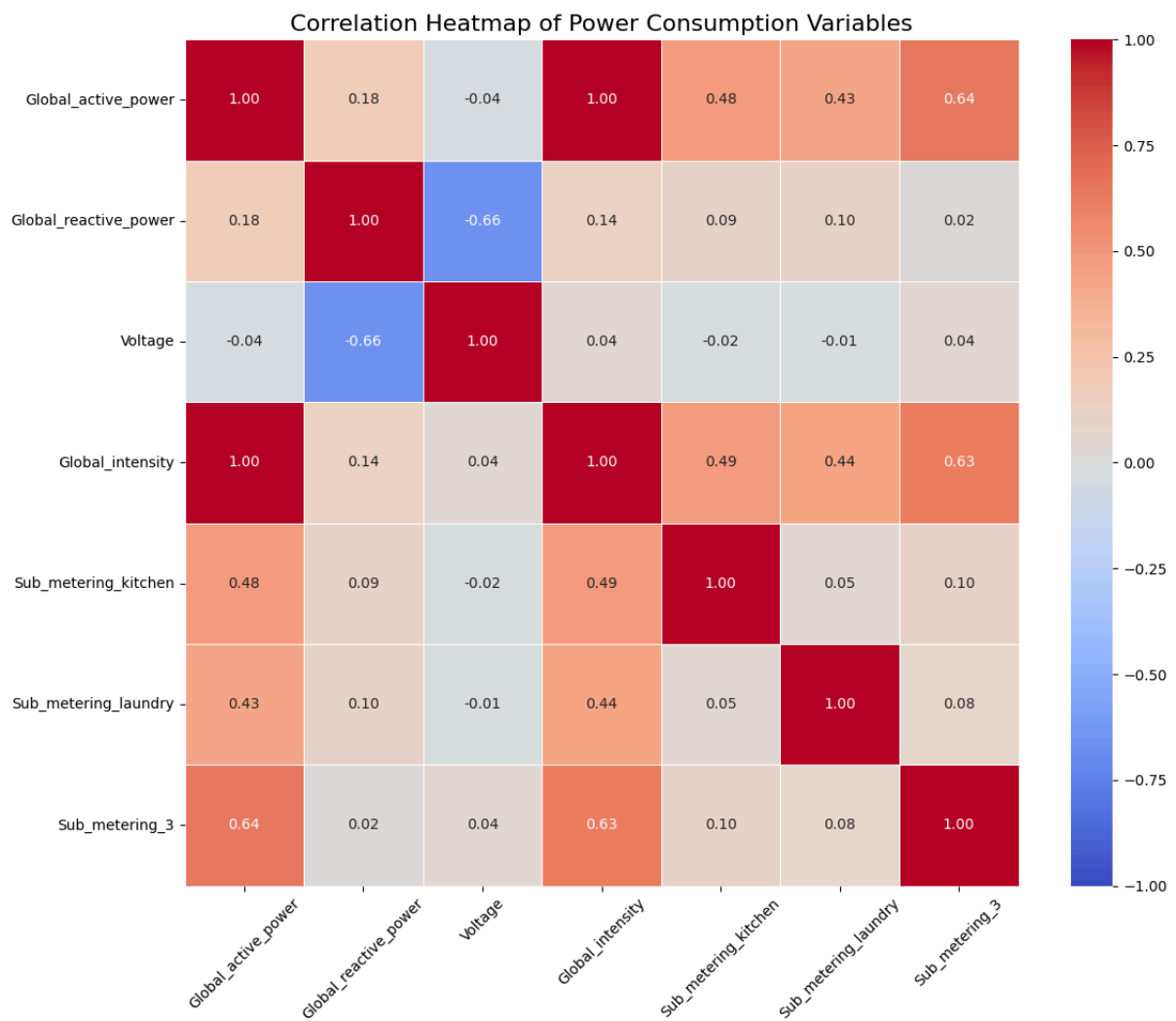
# Plot the heatmap
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt='.2f',
            linewidths=0.5, vmin=-1, vmax=1)

plt.xticks(rotation=45)

plt.title('Correlation Heatmap of Power Consumption Variables', fontsize=16)
plt.tight_layout()

# plt.savefig('correlation_heatmap.png', dpi=300)

plt.show()
```



Menganalisis hubungan **korelasi antar variabel numerik** dalam dataset. Membantu menemukan variabel-variabel yang saling berhubungan dan mempengaruhi satu sama lain, seperti hubungan antara `Global_active_power` dan `Global_intensity`.

## 2.10 Weekday vs Weekend Comparison (Perbandingan Hari Kerja vs Akhir Pekan)

```
[ ] # Add day of week and weekend columns
df_with_day = df.copy()
df_with_day['day_of_week'] = df_with_day['DateTime'].dt.dayofweek
df_with_day['hour'] = df_with_day['DateTime'].dt.hour
df_with_day['is_weekend'] = df_with_day['day_of_week'].apply(lambda x: 1 if x >= 5 else 0)

# Group by hour and weekend flag
hourly_group = df_with_day.groupby(['hour', 'is_weekend']).mean()

# Reset index to have hour and is_weekend as columns
hourly_pivot = hourly_group.reset_index()

# Split data into weekday and weekend
weekday_data = hourly_pivot[hourly_pivot['is_weekend'] == 0]
weekend_data = hourly_pivot[hourly_pivot['is_weekend'] == 1]

# Get numeric columns excluding day_of_week, hour, and is_weekend
numeric_cols = df_with_day.select_dtypes(include=[np.number]).columns.tolist()
cols_to_exclude = ['day_of_week', 'hour', 'is_weekend']
for col in cols_to_exclude:
    if col in numeric_cols:
        numeric_cols.remove(col)

# Plot the comparison
fig, axes = plt.subplots(len(numeric_cols), 1, figsize=(15, 3*len(numeric_cols)))
fig.suptitle('Weekday vs Weekend Power Consumption Patterns', fontsize=16)

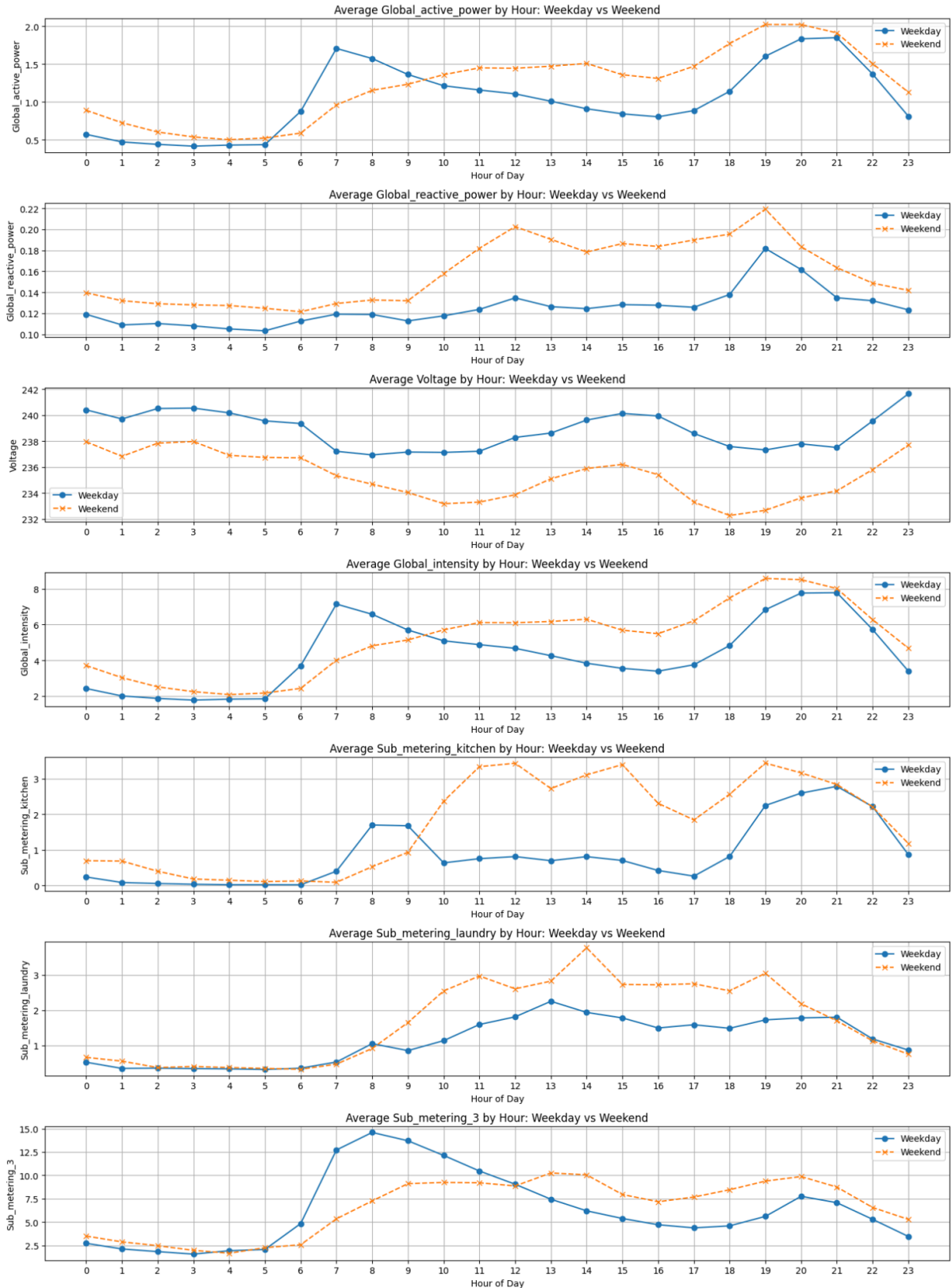
for i, col in enumerate(numeric_cols):
    try:
        axes[i].plot(weekday_data['hour'], weekday_data[col],
                     label='Weekday', marker='o', linestyle='-')
        axes[i].plot(weekend_data['hour'], weekend_data[col],
                     label='Weekend', marker='x', linestyle='--')
        axes[i].set_title(f'Average {col} by Hour: Weekday vs Weekend')
        axes[i].set_xlabel('Hour of Day')
        axes[i].set_ylabel(col)
        axes[i].grid(True)
        axes[i].legend()
        axes[i].set_xticks(range(0, 24))
    except Exception as e:
        print(f"Error plotting {col}: {e}")

plt.tight_layout()
plt.subplots_adjust(top=0.95)

# plt.savefig('weekday_weekend_comparison.png', dpi=300)

plt.show()
```

## Weekday vs Weekend Power Consumption Patterns



Menganalisis perbedaan pola konsumsi daya antara hari kerja dan akhir pekan. Memahami bagaimana konsumsi daya bervariasi antara hari-hari biasa (Senin hingga Jumat) dan akhir pekan (Sabtu dan Minggu).

### 3. Implementasi Model

#### 3.1 Linear Regression

##### a. Menyiapkan Data

```
[ ] # Membuat salinan dataset
df_train = df.copy()

# Membuat fitur waktu dari DateTime
df_train['hour'] = df_train['DateTime'].dt.hour
df_train['day_of_week'] = df_train['DateTime'].dt.dayofweek
df_train['month'] = df_train['DateTime'].dt.month
```

Membuat fitur waktu tambahan dari kolom DateTime, yaitu jam, hari dalam minggu, dan bulan. Fitur-fitur ini dapat membantu model memahami pola yang bergantung pada waktu, misalnya konsumsi daya di jam tertentu atau pada hari-hari tertentu dalam seminggu.

##### b. Memisahkan Fitur dan Target

```
# Memisahkan fitur dan target
X = df_train[['Global_reactive_power', 'Voltage', 'Global_intensity',
              'Sub_metering_kitchen', 'Sub_metering_laundry', 'Sub_metering_3',
              'hour', 'day_of_week', 'month']]

y = df_train['Global_active_power']
```

Memisahkan fitur (X) dan target (y). Target yang ingin diprediksi adalah Global\_active\_power, sedangkan fitur-fitur yang digunakan adalah variabel-variabel yang berhubungan dengan konsumsi daya.

##### c. Membagi Data menjadi Training dan Testing Set

```
# Membagi data menjadi training dan testing set (80% training, 20% testing)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Memisahkan data menjadi dua bagian: 80% untuk training dan 20% untuk testing. Pembagian ini digunakan untuk melatih model pada data training dan menguji kinerjanya pada data testing yang belum pernah dilihat sebelumnya.

##### d. Normalisasi Fitur

```
# Normalisasi fitur
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Standarisasi data fitur yang memiliki skala yang berbeda (seperti tegangan dan konsumsi daya) perlu distandarisasi agar model regresi linear dapat bekerja dengan baik. Menggunakan StandardScaler untuk mengubah data agar memiliki rata-rata 0 dan standar deviasi 1.

##### e. Membangun dan Melatih Model Regresi Linear

```
[ ] # Membuat dan melatih model regresi linear
linear_model = LinearRegression()
linear_model.fit(X_train_scaled, y_train)
```

Membangun model regresi linear dengan menggunakan `LinearRegression()` dari `scikit-learn` dan melatihnya pada data training yang telah distandarisasi.

f. Prediksi dengan Model Linear

```
[ ] # Membuat dan melatih model regresi linear
linear_model = LinearRegression()
linear_model.fit(X_train_scaled, y_train)
```

Menggunakan model yang telah dilatih untuk melakukan prediksi konsumsi daya aktif (`Global_active_power`) pada data testing yang belum dilihat oleh model.

### 3.2 Polynomial Regression

a. Menambahkan Fitur Polinomial

```
[ ] poly = PolynomialFeatures(degree=2)

X_poly_train = poly.fit_transform(X_train_scaled)
X_poly_test = poly.transform(X_test_scaled)
```

Menggunakan `PolynomialFeatures(degree=2)` untuk menghasilkan fitur polinomial dari fitur yang ada. Ini memungkinkan model menangkap hubungan non-linear antara fitur dan target. `Degree=2` berarti bahwa model akan menggunakan kuadrat (squared) dan interaksi antar fitur (misalnya  $X_1^2$ ,  $X_2^2$ , dan  $X_1 \cdot X_2$ ).

b. Membangun dan Melatih Model Regresi Polinomial

```
[ ] poly = PolynomialFeatures(degree=2)

X_poly_train = poly.fit_transform(X_train_scaled)
X_poly_test = poly.transform(X_test_scaled)
```

Setelah mengubah fitur menjadi bentuk polinomial, model regresi linear digunakan untuk melatih model polinomial pada data yang sudah diproses (fitur polinomial).

c. Prediksi dengan Model Polinomial

```
[ ] poly = PolynomialFeatures(degree=2)

X_poly_train = poly.fit_transform(X_train_scaled)
X_poly_test = poly.transform(X_test_scaled)
```

Melakukan prediksi konsumsi daya aktif menggunakan model polinomial yang telah dilatih.

## 4. Evaluasi Model

### 4.1 Evaluasi Model Regresi Linear dan Regresi Polinomial

```
[ ] print("=== Evaluasi Regresi Linear ===")
mse_linear = mean_squared_error(y_test, y_pred_linear)
mae_linear = mean_absolute_error(y_test, y_pred_linear)
r2_linear = r2_score(y_test, y_pred_linear)

print(f"MSE : {mse_linear:.4f}")
print(f"MAE : {mae_linear:.4f}")
print(f"R2 : {r2_linear:.4f}")

print("\n=== Evaluasi Regresi Polinomial (Degree 2) ===")
mse_poly = mean_squared_error(y_test, y_pred_poly)
mae_poly = mean_absolute_error(y_test, y_pred_poly)
r2_poly = r2_score(y_test, y_pred_poly)

print(f"MSE : {mse_poly:.4f}")
print(f"MAE : {mae_poly:.4f}")
print(f"R2 : {r2_poly:.4f}")

# Perbandingan singkat
print("\n=== Perbandingan Singkat ===")
improvement_r2 = r2_poly - r2_linear
print(f"Perbedaan R2 Polynomial vs Linear: {improvement_r2:.4f}")
```

```
=== Evaluasi Regresi Linear ===
MSE : 0.0022
MAE : 0.0317
R2 : 0.9980

=== Evaluasi Regresi Polinomial (Degree 2) ===
MSE : 0.0013
MAE : 0.0217
R2 : 0.9988

=== Perbandingan Singkat ===
Perbedaan R2 Polynomial vs Linear: 0.0008
```

Berdasarkan hasil evaluasi model, Regresi Linear menghasilkan MSE sebesar 0.0022, MAE sebesar 0.0317, dan  $R^2$  sebesar 0.9980, yang menunjukkan bahwa model ini mampu memprediksi dengan cukup akurat, meskipun ada sedikit kesalahan. Di sisi lain, Regresi Polinomial (Derajat 2) menunjukkan hasil yang sedikit lebih baik dengan MSE 0.0013, MAE 0.0217, dan  $R^2$  0.9988, yang menunjukkan peningkatan akurasi dalam memprediksi dibandingkan regresi linear. Perbedaan  $R^2$  antara model polinomial dan linear adalah 0.0008, menunjukkan sedikit peningkatan kinerja dengan model polinomial, meskipun perbedaannya cukup kecil. Secara keseluruhan, kedua model menunjukkan kinerja yang sangat baik, dengan regresi polinomial sedikit lebih baik dalam menangkap variasi data.

## 4.2 Koefisien Regresi Linear dan Polinomial

```
[ ] # Koefisien Regresi Linear
print("\nKoefisien Regresi Linear:")
coefficients_linear = pd.DataFrame({
    'Feature': X.columns,
    'Coefficient': linear_model.coef_
}).sort_values(by='Coefficient', ascending=False)
print(f"Intercept: {linear_model.intercept_}")
print(coefficients_linear)

# Koefisien Regresi Polinomial
print("\nKoefisien Regresi Polinomial:")
feature_names_poly = poly.get_feature_names_out(input_features=X.columns)
coefficients_poly = pd.DataFrame({
    'Feature': feature_names_poly,
    'Coefficient': poly_model.coef_
}).sort_values(by='Coefficient', ascending=False)
print(f"Intercept: {poly_model.intercept_}")
print(coefficients_poly)
```

```
Koefisien Regresi Linear:
Intercept: 1.090274850638764

   Feature Coefficient
2  Global_intensity  1.042555
5   Sub_metering_3   0.018652
6         hour       0.001383
7   day_of_week     0.000284
8         month    -0.000858
3  Sub_metering_kitchen -0.001326
4  Sub_metering_laundry -0.001792
0  Global_reactive_power -0.022841
1         Voltage    -0.098912

Koefisien Regresi Polinomial:
Intercept: 1.0626307620571225

   Feature Coefficient
3  Global_intensity  1.045335e+00
20 Voltage Global_intensity  1.195923e-01
2         Voltage       9.981926e-02
6   Sub_metering_3    1.973381e-02
```



Pada Regresi Linear, koefisien menunjukkan seberapa besar pengaruh masing-masing fitur terhadap prediksi `Global_active_power`. `Global_intensity` memiliki koefisien tertinggi (1.0426), yang menunjukkan bahwa fitur ini memiliki pengaruh paling besar terhadap konsumsi daya aktif. Sebaliknya, `Voltage` memiliki koefisien negatif yang signifikan (-0.0989), menunjukkan bahwa meningkatnya tegangan cenderung mengurangi konsumsi daya aktif. Fitur lain seperti `Global_reactive_power`, `Sub_metering_kitchen`, dan `Sub_metering_laundry` juga berpengaruh, meskipun dengan koefisien yang lebih kecil.

Untuk Regresi Polinomial (Derajat 2), koefisien yang lebih kompleks mencakup interaksi antar fitur, serta kuadrat dari beberapa fitur. `Global_intensity` tetap menjadi fitur dengan koefisien tertinggi (1.0453), tetapi fitur interaksi seperti `Voltage Global_intensity` (0.1196) dan `Voltage` (0.0998) juga muncul sebagai fitur penting. Fitur polinomial dan interaksi seperti `Voltage^2` dan `Global_reactive_power Global_intensity` memiliki koefisien yang lebih kecil, namun tetap menunjukkan kontribusi yang signifikan dalam model polinomial. Hal ini menunjukkan bahwa model polinomial dapat menangkap hubungan yang lebih kompleks dan non-linear antara fitur-fitur dalam dataset.

Secara keseluruhan, regresi polinomial dengan derajat 2 memberikan gambaran yang lebih rinci tentang hubungan antar fitur dan konsumsi daya aktif, dibandingkan dengan model linear yang hanya menangkap hubungan linier.

### 4.3 Feature Importance (Regresi Linear)

```
[ ] # Koefisien dari model linear
coef_linear = linear_model.coef_

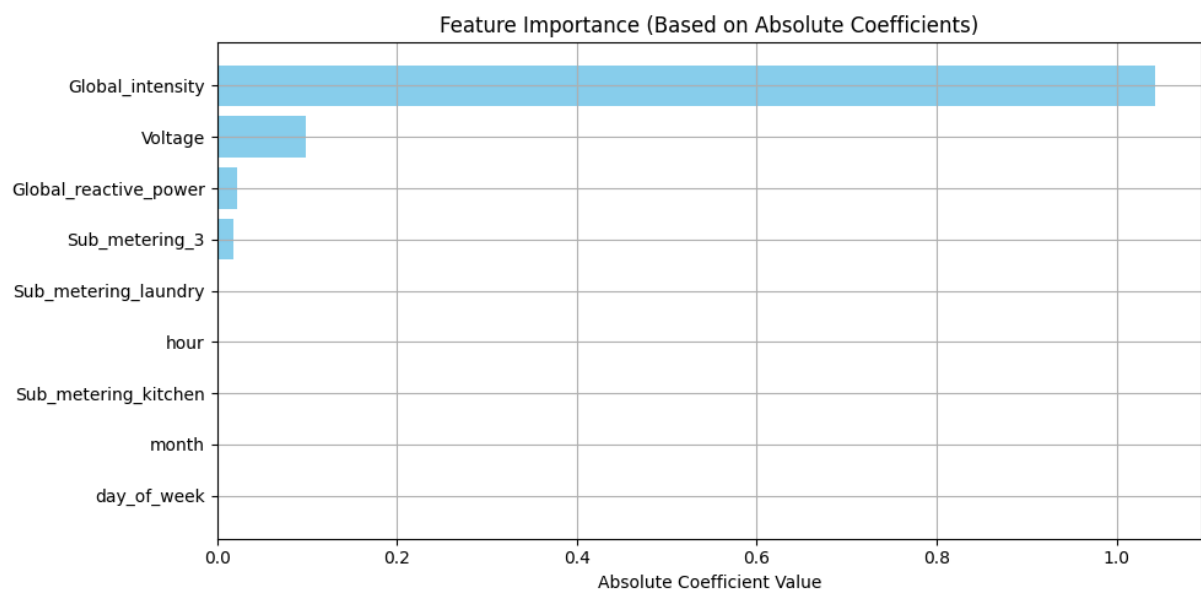
# Buat dataframe untuk fitur dan importance
importance_df = pd.DataFrame({
    'Feature': X.columns,
    'Linear Coefficient': coef_linear,
    'Absolute Importance': np.abs(coef_linear)
}).sort_values(by='Absolute Importance', ascending=False)

print("=== Feature Importance (Linear Regression) ===")
print(importance_df)

# Visualisasi
plt.figure(figsize=(10, 5))
plt.barh(importance_df['Feature'], importance_df['Absolute Importance'], color='skyblue')
plt.title("Feature Importance (Based on Absolute Coefficients)")
plt.xlabel("Absolute Coefficient Value")
plt.gca().invert_yaxis()
plt.grid(True)
plt.tight_layout()
plt.show()
```

```
=== Feature Importance (Linear Regression) ===
```

	Feature	Linear Coefficient	Absolute Importance
2	Global_intensity	1.042555	1.042555
1	Voltage	-0.098912	0.098912
0	Global_reactive_power	-0.022841	0.022841
5	Sub_metering_3	0.018652	0.018652
4	Sub_metering_laundry	-0.001792	0.001792
6	hour	0.001383	0.001383
3	Sub_metering_kitchen	-0.001326	0.001326
8	month	-0.000858	0.000858
7	day_of_week	0.000284	0.000284



Berdasarkan Feature Importance dari model Regresi Linear yang ditampilkan dalam plot dan tabel koefisien, kita dapat melihat fitur-fitur yang paling berpengaruh terhadap `Global_active_power`. `Global_intensity` memiliki koefisien terbesar (1.0426), yang berarti fitur ini memiliki pengaruh paling besar terhadap konsumsi daya aktif. Voltage juga memiliki pengaruh yang cukup signifikan dengan koefisien -0.0989, meskipun koefisiennya negatif, menunjukkan bahwa peningkatan tegangan cenderung menurunkan konsumsi daya aktif. Selain itu, `Global_reactive_power` dan `Sub_metering_3` juga berperan, meskipun dengan koefisien yang lebih kecil. Fitur lainnya, seperti `Sub_metering_laundry`, `hour`, dan `month`, memiliki pengaruh yang lebih kecil dalam model ini.

Plot Feature Importance menunjukkan bahwa fitur seperti `Global_intensity` dan Voltage sangat dominan dalam prediksi konsumsi daya aktif, sedangkan fitur-fitur lainnya memberikan kontribusi yang lebih kecil. Ini memberikan pemahaman yang jelas mengenai fitur mana yang paling relevan dalam model prediksi.

## 4.4 Feature Importance (Regresi Polinomial Derajat 2)

```
[ ] feature_names = poly.get_feature_names_out(input_features=X.columns)
coef_poly = poly_model.coef_

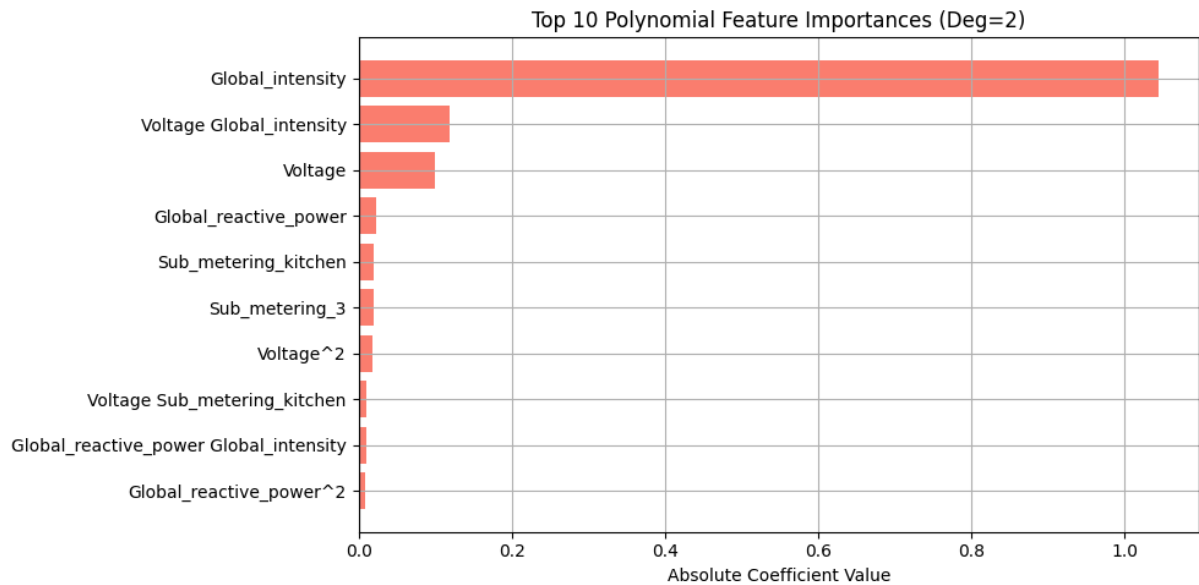
poly_importance_df = pd.DataFrame({
    'Feature': feature_names,
    'Coefficient': coef_poly,
    'Absolute Importance': np.abs(coef_poly)
}).sort_values(by='Absolute Importance', ascending=False)

print("=== Top 10 Feature Interactions (Polynomial) ===")
print(poly_importance_df.head(10))

# Visualisasi untuk Polynomial Regression
plt.figure(figsize=(10, 5))
plt.barh(poly_importance_df.head(10)['Feature'],
          poly_importance_df.head(10)['Absolute Importance'],
          color='salmon')
plt.title("Top 10 Polynomial Feature Importances (Deg=2)")
plt.xlabel("Absolute Coefficient Value")
plt.gca().invert_yaxis()
plt.grid(True)
plt.tight_layout()
plt.show()
```

⇒ === Top 10 Feature Interactions (Polynomial) ===

	Feature	Coefficient	Absolute Importance
3	Global_intensity	1.045335	1.045335
20	Voltage Global_intensity	0.119592	0.119592
2	Voltage	0.099819	0.099819
1	Global_reactive_power	-0.023774	0.023774
4	Sub_metering_kitchen	-0.019763	0.019763
6	Sub_metering_3	0.019734	0.019734
19	Voltage^2	0.018369	0.018369
21	Voltage Sub_metering_kitchen	-0.009951	0.009951
12	Global_reactive_power Global_intensity	0.009732	0.009732
10	Global reactive power^2	-0.007940	0.007940

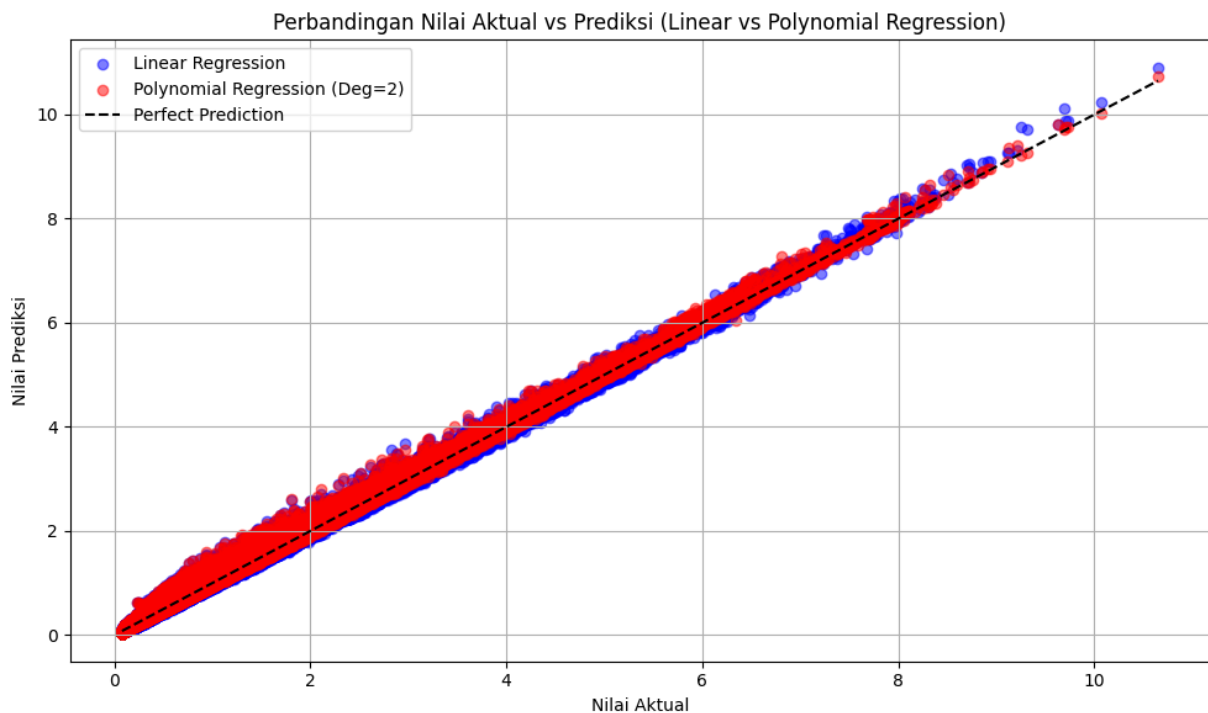


Berdasarkan Feature Importance dari model Regresi Polinomial (Derajat 2), kita dapat melihat bahwa fitur dengan koefisien tertinggi dan pengaruh terbesar terhadap prediksi `Global_active_power` adalah `Global_intensity`. Ini adalah fitur yang paling dominan, dengan nilai Absolute Importance mencapai 1.0453. Selain itu, fitur interaksi seperti `Voltage Global_intensity` dan `Voltage` juga memiliki pengaruh signifikan, dengan nilai koefisien 0.1196 dan 0.0998, yang menunjukkan bahwa interaksi antara tegangan dan intensitas arus, serta tegangan itu sendiri, berperan penting dalam prediksi.

Fitur lain yang menunjukkan pengaruh besar termasuk `Global_reactive_power`, `Sub_metering_kitchen`, dan `Sub_metering_3`, meskipun koefisiennya lebih kecil. Fitur polinomial seperti `Voltage^2` dan interaksi antar fitur seperti `Voltage Sub_metering_kitchen` juga memberikan kontribusi meskipun lebih rendah.

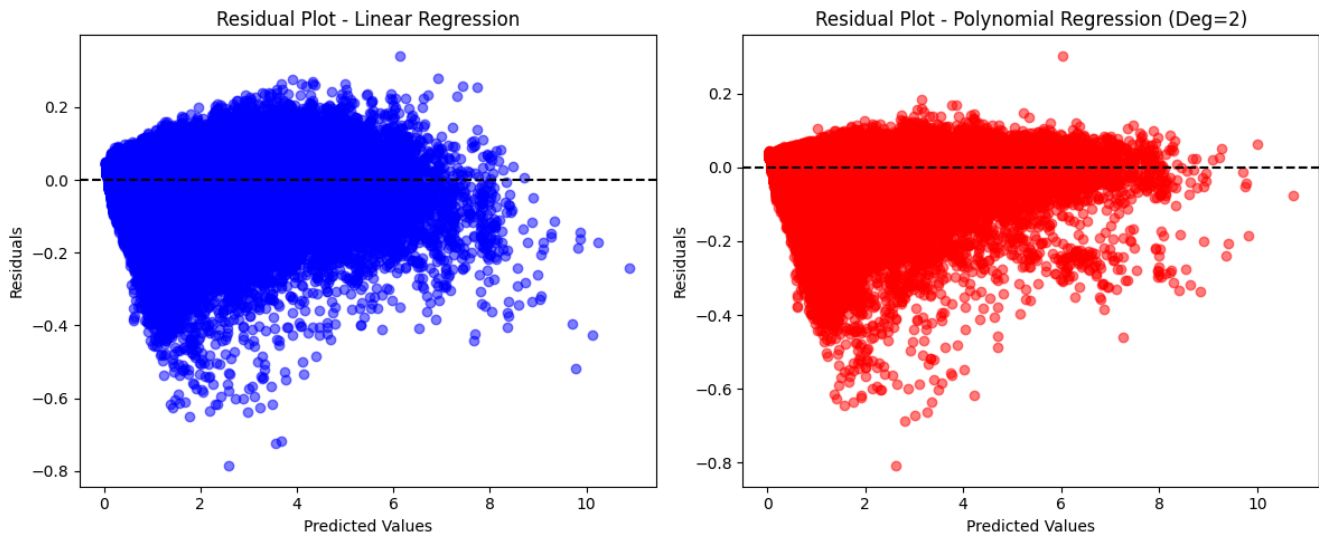
Secara keseluruhan, model polinomial dengan derajat 2 dapat menangkap hubungan non-linear dan interaksi antar fitur yang lebih kompleks, yang mungkin tidak bisa dijelaskan oleh model regresi linear sederhana. Fitur seperti `Global_intensity` dan interaksi antar fitur menjadi sangat penting dalam model ini.

## 4.5 Perbandingan Model Regresi Linear dan Polinomial



Berdasarkan plot perbandingan antara Regresi Linear dan Regresi Polinomial (Derajat 2), terlihat bahwa kedua model memberikan prediksi yang sangat dekat dengan nilai aktual. Titik biru menunjukkan hasil prediksi dari model Linear Regression, sementara titik merah mewakili prediksi dari model Polynomial Regression. Kedua model mengikuti garis prediksi sempurna (dashed black line) dengan sangat baik. Namun, model polinomial (red) cenderung lebih presisi, terutama pada bagian nilai yang lebih tinggi, menunjukkan bahwa model ini lebih mampu menangkap kompleksitas data dan hubungan non-linear antar fitur dengan target. Model Linear juga menunjukkan hasil yang baik tetapi mulai terlihat sedikit perbedaan dari garis prediksi sempurna di bagian nilai yang lebih tinggi. Secara keseluruhan, meskipun kedua model menunjukkan kinerja yang sangat baik, model polinomial sedikit lebih unggul dalam menangani hubungan yang lebih kompleks dalam data.

## 4.6 Residual Plot - Linear vs Polynomial Regression



Berdasarkan Residual Plot untuk model Regresi Linear dan Regresi Polinomial (Derajat 2), kita dapat melihat perbedaan dalam pola residual kedua model. Untuk Regresi Linear, residu tersebar secara tidak teratur di sekitar garis horizontal ( $y=0$ ), dengan pola yang cenderung melebar di bagian nilai yang lebih tinggi. Ini menunjukkan adanya ketidakteraturan dalam model, yang mungkin disebabkan oleh ketidakmampuan model linear dalam menangkap hubungan yang lebih kompleks dan non-linear antar fitur.

Di sisi lain, pada Regresi Polinomial, meskipun masih terdapat beberapa penyimpangan, residu tampak lebih terdistribusi secara merata di sekitar garis horizontal, dengan pola yang lebih terkendali. Hal ini mengindikasikan bahwa model polinomial lebih mampu menangkap hubungan yang lebih kompleks dalam data, dan residu cenderung lebih kecil dan tersebar lebih merata, menunjukkan prediksi yang lebih baik, terutama pada data dengan nilai yang lebih tinggi.

Secara keseluruhan, Residual Plot menunjukkan bahwa model Polinomial lebih unggul dalam menangani data yang lebih kompleks dan memberikan prediksi yang lebih akurat dibandingkan dengan model Linear.

## 5. Kesimpulan

Berdasarkan analisis yang dilakukan, baik Regresi Linear maupun Regresi Polinomial (Derajat 2) menunjukkan kinerja yang sangat baik dalam memprediksi `Global_active_power`, dengan nilai MSE,

MAE, dan  $R^2$  yang sangat rendah. Namun, model Regresi Polinomial sedikit lebih unggul, terutama dalam menangkap hubungan non-linear antar fitur, yang tercermin dari perbaikan kecil pada  $R^2$  dan distribusi residual yang lebih merata. Global\_intensity menjadi fitur paling dominan dalam kedua model, sementara model polinomial juga menunjukkan pentingnya interaksi antar fitur. Secara keseluruhan, Regresi Polinomial lebih efektif dalam menangani kompleksitas data dan memberikan hasil yang lebih akurat dibandingkan Regresi Linear.