Laporan Tugas Mata Kuliah Machine Learning

Prediksi Penjualan Produk E-Commerce Menggunakan Regresi Linear dan Polinomial

disusun untuk memenuhi tugas
mata kuliah Machine Learning A

Oleh :

Fazhira Rizky Harmayani	2308107010012
Cut Dahliana	2308107010027
Naufal Aqil	2208107010043
Hidayat Nur Hakim	2208107010063
Riska Haqika Situmorang	2208107010086

JURUSAN INFORMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS SYIAH KUALA

2025

A. Pemahaman Dataset

Dataset yang digunakan dalam analisis ini adalah eCommerce Product Dataset. Dataset ini berisi informasi tentang produk yang dijual secara online, mencakup harga, rating, jumlah ulasan, diskon, dan lainnya.

Dataset ini akan digunakan untuk mengimplementasikan Regresi Linear dan Regresi Polinomial dalam analisis prediksi. Melalui eksperimen ini, kita akan mengeksplorasi hubungan antara fitur-fitur dalam dataset serta menganalisis pola tren yang dapat dioptimalkan untuk keperluan bisnis e-commerce.Sumber Dataset. Dataset ini dapat diakses melalui tautan berikut: **eCommerce Trends Dataset – Kaggle .**

Variabel yang Digunakan:

- ProductID: ID unik untuk setiap produk
- ProductName: Nama produk
- Category: Kategori produk
- Price: Harga produk
- Rating: Rating pelanggan terhadap produk
- NumReviews: Jumlah ulasan produk
- StockQuantity: Jumlah stok yang tersedia
- Discount: Diskon dalam persentase
- Sales: Jumlah produk terjual
- DateAdded: Tanggal produk ditambahkan
- City: Kota asal produk

Statistik deskriptif dan visualisasi awal dilakukan untuk memahami distribusi data.

	ProductID	Price	Rating	NumReviews	StockQuantity
count	1000.000000	1000.00000	1000.000000	1000.000000	1000.000000
mean	500.500000	253.77551	3.025600	2498.753000	495.395000
std	288.819436	141.40362	1.151004	1463.241871	292.799253
min	1.000000	10.11000	1.000000	3.000000	0.000000
25%	250.750000	133.09250	2.100000	1201.750000	241.750000
50%	500.500000	251.31000	3.100000	2476.000000	505.000000
75%	750.250000	375.82750	4.000000	3797.500000	743.500000
max	1000.000000	499.74000	5.000000	4994.000000	993.000000

	Discount	Sales
count	1000.000000	1000.000000
mean	0.251640	1011.037000
std	0.146455	582.113466
min	0.000000	0.000000
25%	0.130000	502.000000
50%	0.250000	998.000000
75%	0.380000	1540.000000
max	0.500000	1997.000000

Gambar 1. *Statistik deskriptif dan visualisasi awal dilakukan untuk memahami distribusi data.*

Statistik ini menunjukkan data dari 1000 produk, dengan rata-rata harga sekitar 253 dan rating 3 dari 5. Jumlah ulasan cukup tinggi (rata-rata 2498), dan stok rata-rata 495 unit. Diskon rata-rata sebesar 25%, sementara penjualan berkisar rata-rata 1011 unit. Terlihat variasi besar dalam penjualan, kemungkinan dipengaruhi oleh rating, ulasan, harga, dan diskon.

B. Eksplorasi Data dan Pra-pemrosesan

Langkah-langkah yang dilakukan dalam eksplorasi data dan pra-pemrosesan meliputi:

- **Penanganan Missing value**

```
# Mengecek jumlah missing values
print("\nMissing Values dalam Dataset:")
print(df.isnull().sum())
```

Gambar 2. *Kode Mengecek Missing Values*

```
Missing Values dalam Dataset:
ProductID        0
ProductName      0
Category         0
Price            0
Rating           0
NumReviews       0
StockQuantity    0
Discount         0
Sales            0
DateAdded        0
City             0
dtype: int64
```

Gambar 3. *Output Mengecek Missing Values*

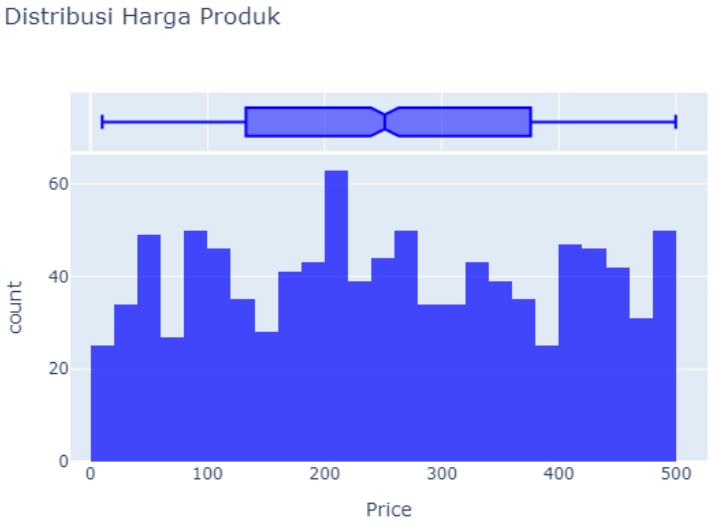
pada gambar diatas, tidak terdapat Missing Values pada dataset.

- **Melihat distribusi pada data**

Kami menggunakan Plotly karena menawarkan visualisasi yang interaktif, informatif, dan modern. Dibanding pustaka lain seperti Matplotlib atau Seaborn, Plotly memungkinkan kombinasi histogram dan boxplot dalam satu grafik, mendukung analisis mendalam dengan fitur zoom dan hover, serta tampil lebih estetis dan responsif untuk data besar.

```python
# Histogram Harga Produk
fig_price = px.histogram(
    df, x="Price", nbins=30, title="Distribusi Harga Produk",
    marginal="box", opacity=0.7, color_discrete_sequence=["blue"]
)

# Menampilkan histogram di Jupyter Notebook
fig_price.show()

# Load dan tampilkan gambar di Jupyter Notebook
img = Image.open("./assets/Distribusi_HargaProduk.png")
display(img)
```

Gambar 4. *Kode distribusi Harga Produk*



Gambar 5. *Visualisasi distribusi Harga Produk dalam bentuk histogram*

Distribusi harga produk divisualisasikan menggunakan histogram dan boxplot untuk memberikan gambaran menyeluruh. Harga tersebar dari sekitar $10 hingga $500, dengan konsentrasi tertinggi antara $100–$250. Histogram menunjukkan beberapa puncak frekuensi yang menandakan rentang harga populer. Sementara itu, boxplot mengonfirmasi bahwa sebagian besar harga berada dalam interquartile range tanpa outlier ekstrem. Visualisasi ini menunjukkan distribusi harga yang relatif merata dan memberikan wawasan penting untuk analisis lebih lanjut terkait pengaruh harga terhadap penjualan.

```
# Histogram Diskon
fig_discount = px.histogram(df, x="Discount", nbins=30, title="Distribusi Diskon", marginal="box", opacity=0.7, color_discrete_sequence=["green"])
fig_discount.show()

# Load dan tampilkan gambar di Jupyter Notebook
img = Image.open("./assets/Distribusi_Diskon.png")
display(img)
```

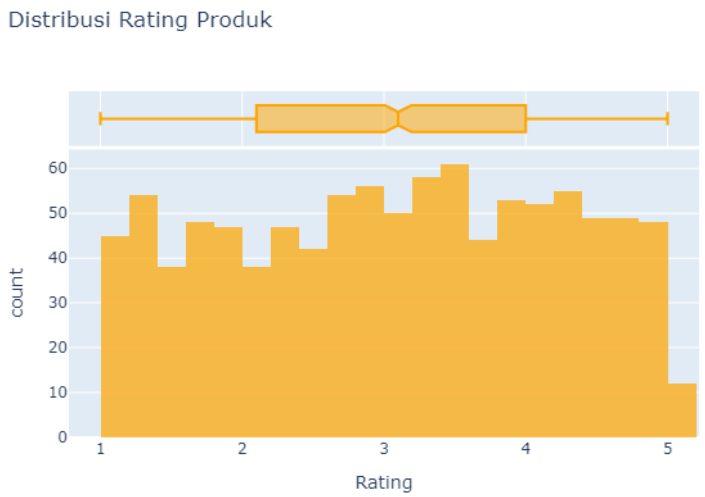
Gambar 6. *Kode distribusi Diskon*

Gambar 7. *Visualisasi distribusi Diskon dalam bentuk histogram*

Distribusi diskon produk divisualisasikan melalui histogram dan boxplot, menunjukkan bahwa diskon berkisar antara 0% hingga 50%, dengan mayoritas produk berada dalam kisaran 10%–40%. Terdapat beberapa puncak frekuensi di sekitar 0.05, 0.2, 0.3, dan 0.4, yang menandakan rentang diskon populer. Median diskon berada di sekitar 25%, dan tidak terdeteksi outlier signifikan. Secara keseluruhan, pola ini mencerminkan strategi diskon yang cukup merata dan dapat menjadi acuan dalam analisis pengaruh diskon terhadap penjualan.

```
# Histogram Rating Produk
fig_rating = px.histogram(df, x="Rating", nbins=30, title="Distribusi Rating Produk", marginal="box", opacity=0.7, color_discrete_sequence=["orange"])

# Menampilkan histogram di Jupyter Notebook
fig_rating.show()

# Load dan tampilkan gambar di Jupyter Notebook
img = Image.open("./assets/Distribusi_RatingProduk.png")
display(img)
```

Gambar 8. *Kode distribusi Rating Produk*

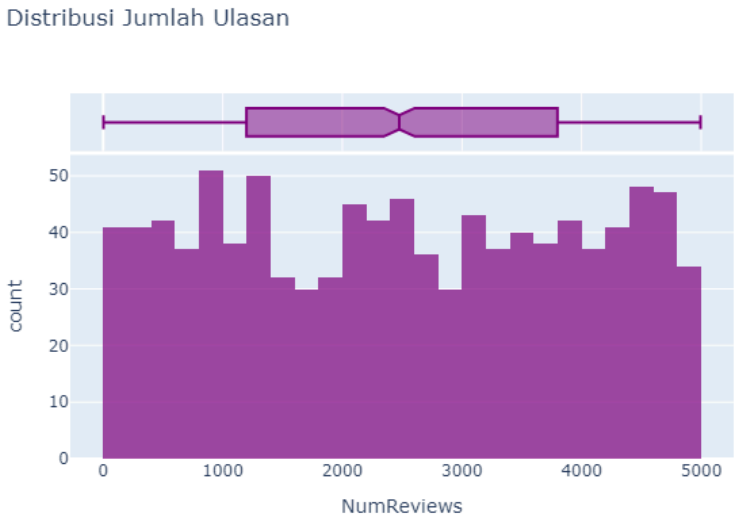


Gambar 9. *Visualisasi distribusi Rating Produk dalam bentuk histogram*

Distribusi rating produk menunjukkan rentang nilai antara 1 hingga 5, dengan mayoritas produk memiliki rating di antara 2 hingga 4.5. Histogram memperlihatkan sebaran yang cukup merata, dengan kecenderungan puncak di rating menengah. Boxplot mengindikasikan IQR berada antara 2.2 hingga 4.5, dan median rating sekitar 3.1, tanpa adanya outlier signifikan. Secara keseluruhan, rating produk dalam dataset cenderung berada di atas angka 2, mencerminkan ulasan pengguna yang cukup bervariasi dan tidak terkonsentrasi di satu nilai ekstrem.

```
# Histogram Jumlah Ulasan
fig_reviews = px.histogram(df, x="NumReviews", nbins=30, title="Distribusi Jumlah Ulasan", marginal="box", opacity=0.7, color_discrete_sequence=["purple"])

# Menampilkan histogram di Jupyter Notebook
fig_reviews.show()

# Load dan tampilkan gambar di Jupyter Notebook
img = Image.open("./assets/Distribusi_JumlahUlasan.png")
display(img)
```

Gambar 10. *Kode distribusi Jumlah Ulasan*

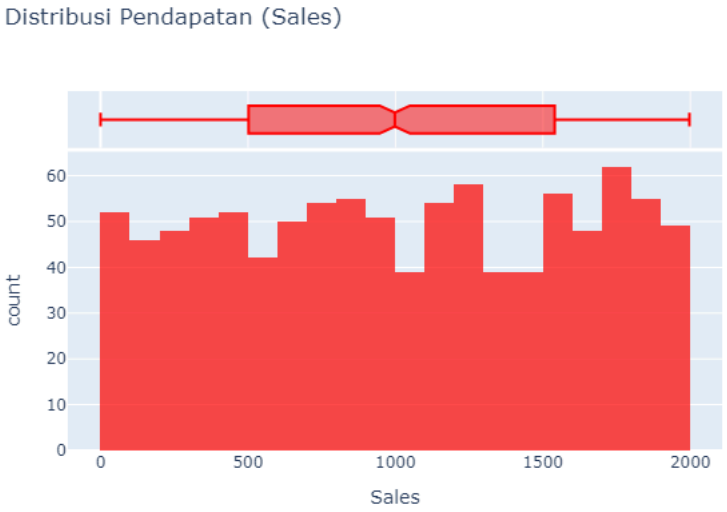


Gambar 11. *Visualisasi distribusi Jumlah Ulasan dalam bentuk histogram*

Distribusi jumlah ulasan produk menunjukkan rentang yang luas dari 0 hingga 5000, dengan sebaran yang cukup merata. Histogram menampilkan beberapa puncak di rentang tertentu, menandakan adanya kelompok produk dengan jumlah ulasan umum. Boxplot mengindikasikan IQR berada di kisaran 500 hingga 4000, dengan median sekitar 2500. Sebagian produk memiliki ulasan sangat rendah maupun sangat tinggi, mencerminkan variasi interaksi pelanggan. Visualisasi ini menunjukkan bahwa sebagian besar produk memiliki ulasan dalam jumlah menengah hingga tinggi yang berpotensi memengaruhi rating atau penjualan.

```
# Histogram Pendapatan (Sales)
fig_sales = px.histogram(df, x="Sales", nbins=30, title="Distribusi Pendapatan (Sales)", marginal="box", opacity=0.7, color_discrete_sequence=["red"])

# Menampilkan histogram di Jupyter Notebook
fig_sales.show()

# Load dan tampilkan gambar di Jupyter Notebook
img = Image.open("./assets/Distribusi_Pendapatan.png")
display(img)
```

Gambar 12. *Kode distribusi Pendapatan (sales)*



Gambar 13. *Visualisasi distribusi Pendapatan (sales) dalam bentuk histogram*

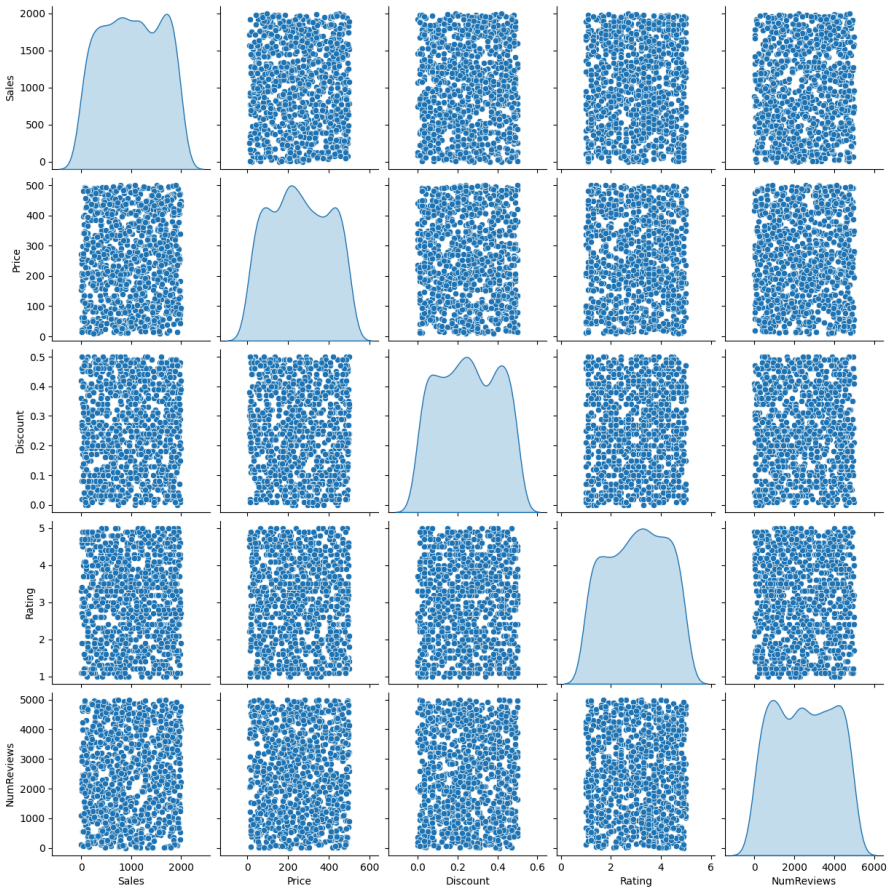
Distribusi pendapatan (Sales) tersebar merata antara 0 hingga 2000, dengan sebagian besar produk menghasilkan pendapatan di kisaran 500–1500. Median pendapatan sekitar 1000, dan beberapa produk memiliki pendapatan sangat rendah atau sangat tinggi. Hal ini mencerminkan variasi performa penjualan yang dipengaruhi oleh harga, diskon, dan rating.

- **Scatter Plot Harga, Diskon, Rating, Jumlah Ulasan vs Sales**

Scatter plot ini bertujuan untuk menganalisis hubungan antar variabel dalam dataset, khususnya bagaimana Harga (Price), Diskon (Discount), Rating, dan Jumlah Ulasan (NumReviews) berhubungan dengan Pendapatan (Sales).

```
# Scatter Plot antara Harga, Diskon, Rating, Jumlah Ulasan vs Sales
sns.pairplot(df[['Sales', 'Price', 'Discount', 'Rating', 'NumReviews']], diag_kind="kde")
plt.show()
```

Gambar 14. *Kode Menganalisis hubungan antar variabel*



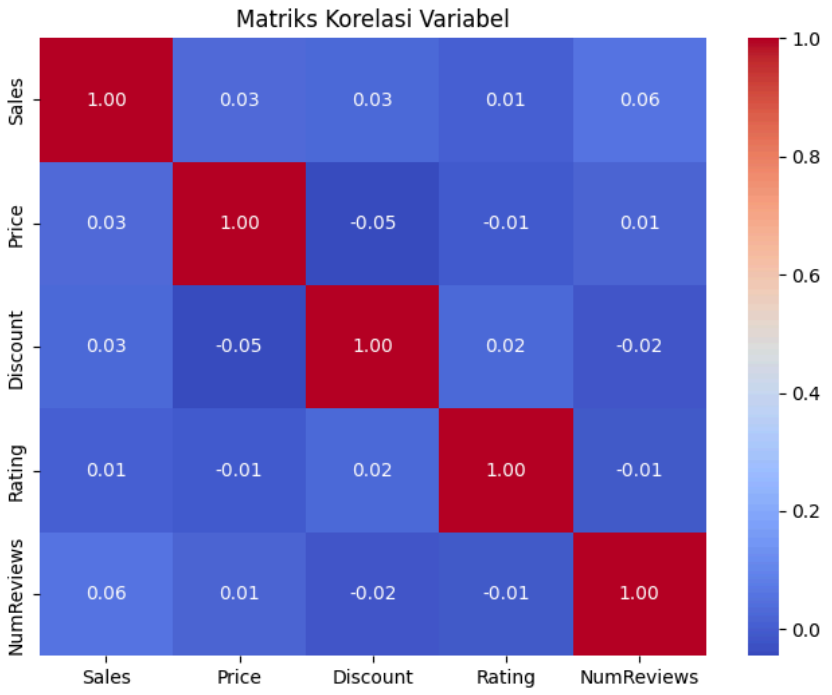
Gambar 15. *Scatter Plot antara Harga, Diskon, Rating, Jumlah Ulasan vs Sales*

Scatter plot ini menunjukkan hubungan antara Harga, Diskon, Rating, Jumlah Ulasan, dan Sales. Hasilnya, tidak tampak korelasi linear yang kuat antar variabel. Distribusi data cukup bervariasi, terutama pada Sales, Price, dan NumReviews. Visualisasi ini membantu memahami bahwa hubungan antar variabel mungkin bersifat kompleks dan perlu dianalisis lebih lanjut dengan metode statistik atau pemodelan.

- **Mengevaluasi hubungan linear antar variabel dengan menggunakan matriks korelasi**

```
# Korelasi antar variabel
plt.figure(figsize=(8, 6))
sns.heatmap(df[['Sales', 'Price', 'Discount', 'Rating', 'NumReviews']].corr(), annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Matriks Korelasi Variabel")
plt.show()
```

Gambar 16. *Kode Menganalisis hubungan antar variabel menggunakan matriks korelasi*



Gambar 17. *Matriks Korelasi*

Heatmap korelasi menunjukkan bahwa tidak ada hubungan linear yang kuat antara Sales dengan variabel lain seperti Price, Discount, Rating, dan NumReviews—semuanya memiliki korelasi sangat lemah (nilai mendekati 0). Begitu pula antar fitur lainnya, hubungan korelatifnya rendah. Hal ini mengindikasikan bahwa pola hubungan dalam data mungkin bersifat non-linear, sehingga dibutuhkan pendekatan analisis lanjutan seperti regresi kompleks atau model machine learning untuk memahami faktor-faktor yang memengaruhi Sales.

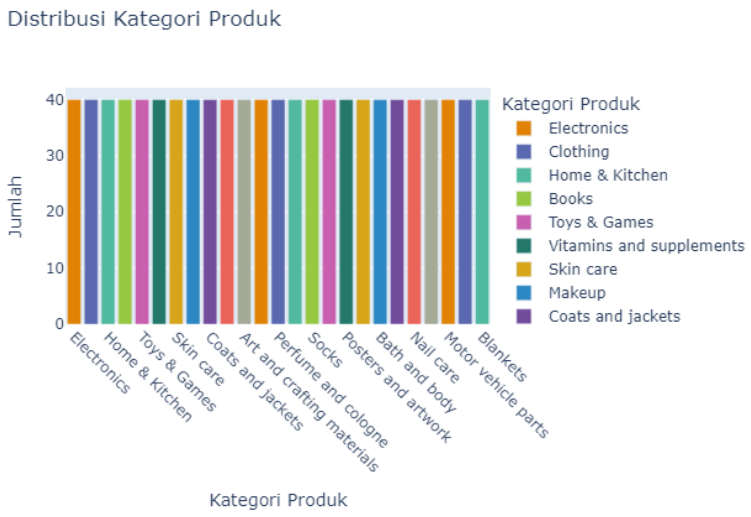
- Analisis Kategori Produk

```python
if "Category" in df.columns:
    # Menampilkan kategori unik dan jumlah kategori
    unique_categories = df["Category"].unique()
    num_categories = df["Category"].nunique()

    print("Kategori Produk Unik:", unique_categories)
    print("Jumlah Kategori Produk:", num_categories)

    # Hitung jumlah tiap kategori
    category_counts = df["Category"].value_counts().reset_index()
    category_counts.columns = ["Category", "count"]  # Pastikan kolom memiliki nama yang sesuai

    # Bar Chart untuk distribusi kategori produk
    fig_bar = px.bar(
        category_counts,
        x="Category",
        y="count",
        title="Distribusi Kategori Produk",
        labels={"Category": "Kategori Produk", "count": "Jumlah"},
        color="Category",
        color_discrete_sequence=px.colors.qualitative.Vivid
    )
    fig_bar.update_xaxes(tickangle=45)
    fig_bar.show()

    # Pie Chart untuk distribusi kategori produk
    fig_pie = px.pie(
        category_counts,
        names="Category",
        values="count",
        title="Proporsi Kategori Produk",
        color_discrete_sequence=px.colors.qualitative.Pastel
    )
    fig_pie.show()
```

Gambar 18. *Kode Analisis Kategori Produk*

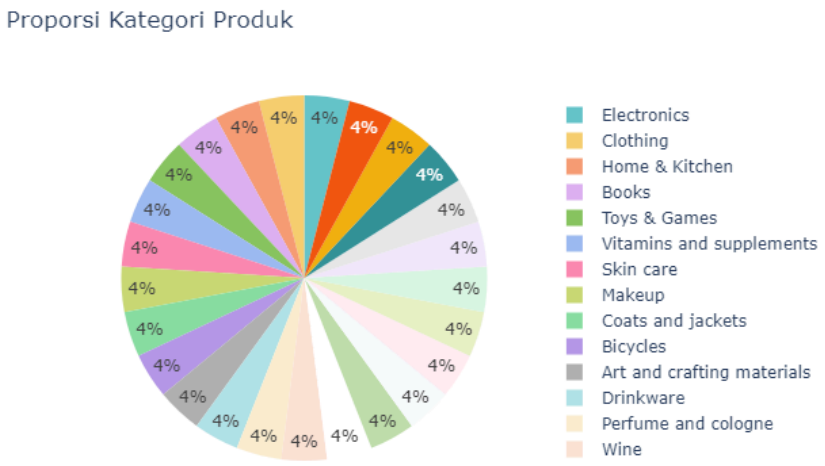
```
Kategori Produk Unik: ['Electronics' 'Clothing' 'Home & Kitchen' 'Books' 'Toys & Games'
 'Vitamins and supplements' 'Skin care' 'Makeup' 'Coats and jackets'
 'Bicycles' 'Art and crafting materials' 'Drinkware' 'Perfume and cologne'
 'Wine' 'Socks' 'Bedsheets' 'Posters and artwork' 'Candles'
 'Bath and body' 'Cookware' 'Nail care' 'Underwear' 'Motor vehicle parts'
 'Mobile phone accessories' 'Blankets']
Jumlah Kategori Produk: 25
```

Gambar 19. *Output kode Analisis Kategori Produk*



Gambar 20. *Distribusi Kategori Produk*

Bar chart distribusi kategori produk menunjukkan bahwa jumlah produk di setiap kategori relatif seimbang, tanpa adanya kategori yang mendominasi secara signifikan. Kategori produk juga cukup beragam, mencakup Electronics, Clothing, Home & Kitchen, Books, Toys & Games, dan lainnya. Visualisasi ini memberikan gambaran umum mengenai sebaran produk dalam dataset dan menunjukkan bahwa tidak ada bias besar terhadap satu kategori tertentu.



Gambar 21. *Proporsi Kategori Produk*

Pie chart proporsi kategori produk menunjukkan bahwa setiap kategori memiliki persentase yang hampir sama, sekitar 4%, tanpa ada yang mendominasi. Distribusi yang seimbang ini menandakan bahwa dataset cukup representatif dan tidak bias terhadap kategori tertentu. Visualisasi ini juga membuka peluang analisis lebih lanjut, seperti melihat hubungan antara kategori dengan variabel lain seperti Sales, Price, atau NumReviews.

- **Eksplorasi Data dan Pra-pemrosesan**

  Kita hanya menggunakan kolom yang mempengaruhi Sales:

  ➢ Price (Harga produk)

  ➢ Discount (Diskon yang diberikan)

  ➢ Rating (Penilaian produk)

  ➢ NumReviews (Jumlah ulasan)

```
# Pilih hanya kolom yang relevan untuk analisis
selected_columns = ["Sales", "Price", "Discount", "Rating", "NumReviews"]
df_clean = df[selected_columns].copy()

# Tampilkan 5 data pertama untuk verifikasi
print(df_clean.head())
```

Gambar 22. *Implementasi Kode*

	Sales	Price	Discount	Rating	NumReviews
0	466	400.31	0.08	1.7	3772
1	1332	235.03	0.33	2.3	2919
2	252	417.90	0.31	1.8	1184
3	1806	152.70	0.49	3.4	2047
4	1508	394.74	0.23	1.8	1267

Fitur-fitur yang digunakan dalam analisis ini meliputi Sales sebagai target atau pendapatan dari produk, Price yang menunjukkan harga produk, Discount berupa potongan harga dalam bentuk desimal (misalnya 0.33 berarti diskon 33%), Rating yang menggambarkan penilaian pelanggan dalam skala 1 sampai 5, serta NumReviews yang menunjukkan jumlah ulasan yang diberikan pelanggan.

- **Menangani Missing Values**

```
# Cek missing values dalam dataset
missing_values = df_clean.isnull().sum()
print("Missing values per kolom:\n", missing_values)
```

Gambar 23.. *kode untuk Mengecek Missing Values*

```
Missing values per kolom:
 Sales         0
Price          0
Discount       0
Rating         0
NumReviews     0
dtype: int64
```

Gambar 24.. *Output Mengecek Missing Values*

Dataset ini tidak memiliki missing values, sehingga semua fitur memiliki data yang lengkap. Oleh karena itu, langkah penanganan seperti imputasi atau penghapusan data tidak diperlukan. Data sudah siap untuk digunakan dalam tahap eksplorasi lebih lanjut seperti analisis korelasi, deteksi outlier, dan pemodelan regresi.

- **Menangani Outlier**

Sebelum menghapus atau menangani outlier, kita perlu mengidentifikasi outlier menggunakan Interquartile Range (IQR):

```python
# Menghitung IQR
Q1 = df_clean.quantile(0.25)
Q3 = df_clean.quantile(0.75)
IQR = Q3 - Q1

# Menentukan batas bawah dan atas
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Menandai outlier
outliers = ((df_clean < lower_bound) | (df_clean > upper_bound))

# Menampilkan jumlah outlier per kolom
print("Jumlah outlier per kolom:\n", outliers.sum())
```

Gambar 25.. *Kode untuk mengidentifikasi outlier*

```
Jumlah outlier per kolom:
 Sales         0
Price          0
Discount       0
Rating         0
NumReviews     0
dtype: int64
```

Gambar 26. *Output mengidentifikasi outlier*

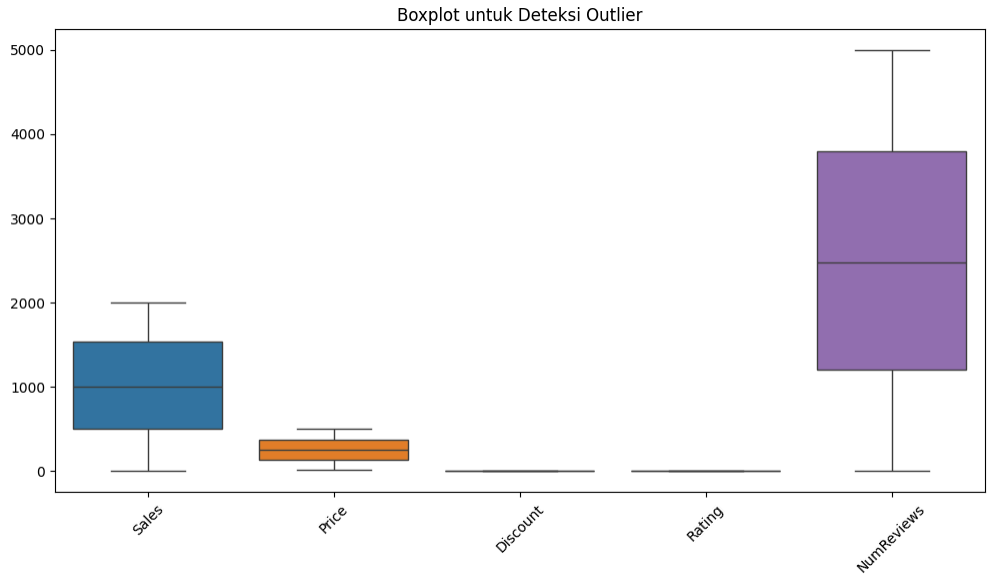
Tidak ditemukan outlier dalam dataset, sehingga tidak diperlukan penanganan khusus seperti penghapusan atau transformasi data. Hal ini menunjukkan bahwa data cukup bersih dan terdistribusi dengan baik, sehingga dapat langsung digunakan untuk pemodelan regresi tanpa risiko distorsi akibat outlier.

- **Visualisasi Outlier dengan Boxplot**

Setelah mendeteksi outlier, kita bisa melihat distribusi datanya dengan boxplot:

```
plt.figure(figsize=(12, 6))
sns.boxplot(data=df_clean)
plt.title("Boxplot untuk Deteksi Outlier")
plt.xticks(rotation=45)
plt.show()
```

Gambar 26. *Kode Visualisasi outlier dengan Boxplot*



Gambar 27. *Boxplot untuk Deteksi Outlier*

Tidak ada outlier yang terdeteksi pada semua fitur, sesuai dengan hasil perhitungan IQR. Fitur Sales dan NumReviews memang memiliki sebaran data yang lebih luas, namun masih berada dalam batas normal. Sementara itu, fitur Discount dan Rating memiliki rentang nilai yang lebih kecil dan menunjukkan distribusi data yang baik.

- **Menghitung Persentase Outlier dengan IQR**

Untuk memahami dampak outlier dalam dataset:

```
# Menghitung persentase outlier per kolom
outlier_percentage = (outliers.sum() / len(df_clean)) * 100
print("Persentase outlier per kolom:\n", outlier_percentage)
```

Gambar 28. *Kode Menghitung Presentase Outlier per kolom*

```
Persentase outlier per kolom:
 Sales         0.0
Price         0.0
Discount      0.0
Rating        0.0
NumReviews    0.0
dtype: float64
```

Gambar 29. *Output Presentase Outlier per kolom*

Karena tidak ada outlier yang terdeteksi, maka tidak diperlukan tindakan tambahan seperti penghapusan atau transformasi nilai ekstrem. Data dapat langsung digunakan untuk proses standarisasi agar setiap fitur yang memiliki skala berbeda dapat diolah secara optimal dalam model regresi.

1. **Normalisasi/Standarisasi Data**

Normalisasi atau standarisasi diperlukan jika skala antar fitur sangat berbeda. Contoh: Sales memiliki nilai ribuan, sementara Discount hanya dalam rentang 0-1.

```python
from sklearn.preprocessing import StandardScaler

# Standarisasi hanya untuk kolom numerik kecuali target (Sales)
scaler = StandardScaler()
df_clean_scaled = df_clean.copy()
df_clean_scaled[["Price", "Discount", "Rating", "NumReviews"]] = scaler.fit_transform(
    df_clean[["Price", "Discount", "Rating", "NumReviews"]]
)

# Tampilkan data setelah standarisasi
print(df_clean_scaled.head())
```

Gambar 30. *Kode (Standarisasi dengan Standard Scaler)*

	Sales	Price	Discount	Rating	NumReviews
0	466	1.036804	-1.172553	-1.152267	0.870590
1	1332	-0.132634	0.535314	-0.630722	0.287346
2	252	1.161262	0.398685	-1.065343	-0.898970
3	1806	-0.715159	1.628349	0.325444	-0.308889
4	1508	0.997393	-0.147833	-1.065343	-0.842218

Gambar 31. *Output Kode (Standarisasi dengan Standard Scaler)*

Standarisasi dilakukan untuk mengubah skala fitur sehingga memiliki rata-rata 0 dan standar deviasi 1, sehingga semua fitur memiliki bobot yang seimbang dalam analisis regresi. Kolom "Sales" tetap dalam skala aslinya karena merupakan target yang akan diprediksi, sementara kolom "Price", "Discount", "Rating", dan "NumReviews" telah diubah skalanya melalui proses standarisasi.

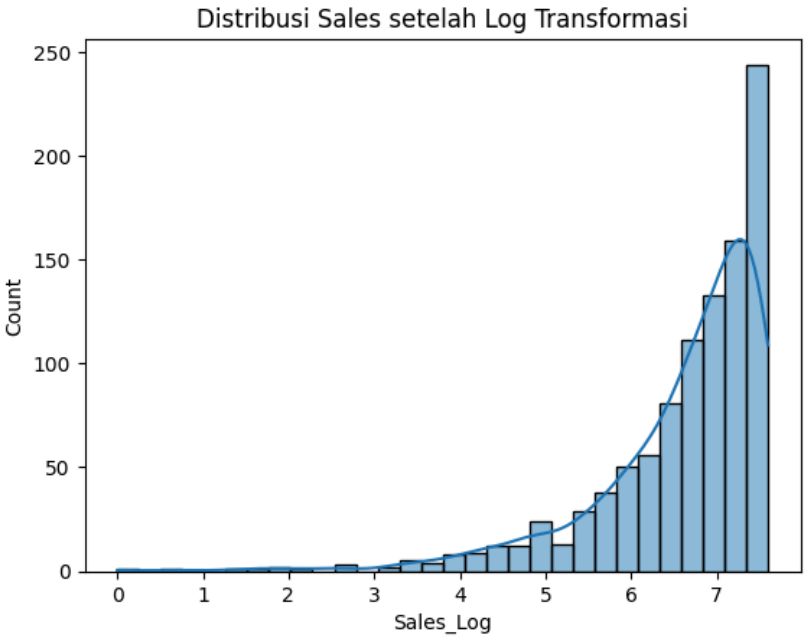
- **Encoding Data Kategorik**

  Tidak perlu dilakukan, karena semua kolom sudah berbentuk numerik.

- **Transformasi Data untuk Meningkatkan Model**

```
# Terapkan log transformasi pada Sales dan fitur numerik yang relevan
df_clean["Sales_Log"] = np.log1p(df_clean["Sales"])
df_clean["Price_Log"] = np.log1p(df_clean["Price"])
df_clean["NumReviews_Log"] = np.log1p(df_clean["NumReviews"])

# Periksa distribusi setelah transformasi
sns.histplot(df_clean["Sales_Log"], bins=30, kde=True)
plt.title("Distribusi Sales setelah Log Transformasi")
plt.show()
```

Gambar 32. *Kode (Transformasi Log pada Fitur Numerik)*



Gambar 33. *Distribusi Sales setelah Log Transformasi*

Transformasi log menggunakan np.log1p() dilakukan pada fitur Sales, Price, dan NumReviews untuk mengatasi distribusi data yang miring (skewed) agar lebih mendekati distribusi normal. Transformasi ini membuat distribusi menjadi lebih simetris, yang bermanfaat untuk meningkatkan performa model regresi linear. Hal ini penting karena regresi linear mengasumsikan bahwa residual terdistribusi normal, dan log transformasi juga membantu mengurangi efek outlier dengan memperkecil perbedaan antara nilai besar dan kecil secara non-linear.

- **Standarisasi Fitur Numerik**

```
scaler = StandardScaler()
numeric_features = ["Price", "Discount", "Rating", "NumReviews"]
df_clean[numeric_features] = scaler.fit_transform(df_clean[numeric_features])
```

Gambar 34. *Kode Standarisasi Fitur Numerik*

Standarisasi dengan StandardScaler() dilakukan pada fitur numerik seperti Price, Discount, Rating, dan NumReviews untuk menyamakan skala antar fitur. Proses ini penting agar tidak ada fitur yang mendominasi model karena perbedaan skala. Selain itu, standarisasi meningkatkan stabilitas dan kinerja model regresi, terutama dalam algoritma seperti Polynomial Regression yang sensitif terhadap skala data.

- **Menambahkan Fitur Polinomial untuk Model Non-Linear**

```
poly = PolynomialFeatures(degree=2, interaction_only=False, include_bias=False)
X_poly = poly.fit_transform(df_clean[["Price", "Discount", "Rating", "NumReviews"]])
```

Gambar 35. *Kode Standarisasi Fitur Numerik*

Transformasi dengan PolynomialFeatures(degree=2) digunakan untuk menghasilkan fitur tambahan hingga derajat 2 dari variabel Price, Discount, Rating, dan NumReviews. Hasilnya mencakup kuadrat dari masing-masing fitur serta kombinasi interaksinya, seperti Price2Price^2Price2, Discount×RatingDiscount \times RatingDiscount×Rating, dan sebagainya. Transformasi ini penting karena model regresi linier hanya mampu menangkap hubungan linear, sementara fitur polinomial memungkinkan model mengenali pola non-linear dan interaksi antar variabel yang lebih kompleks, sehingga meningkatkan akurasi prediksi.

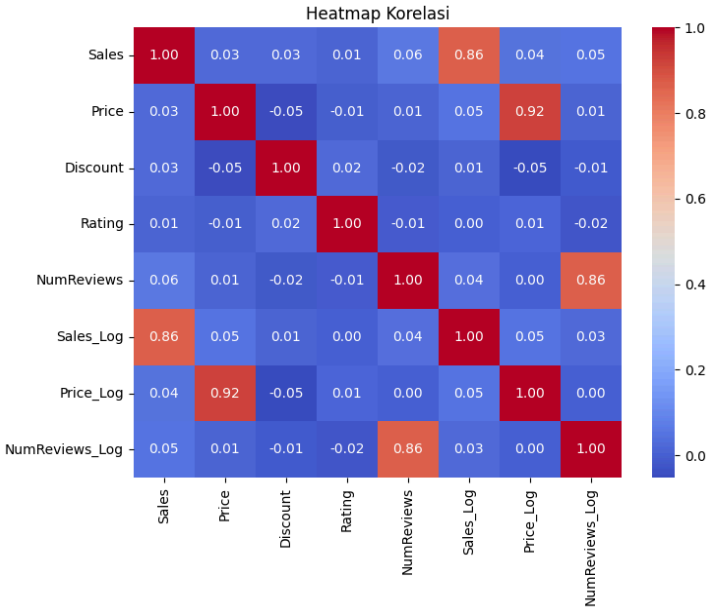
2. Analisis Korelasi

- **Membuat Heatmap Korelasi**

Heatmap korelasi membantu melihat hubungan antar variabel. Korelasi tinggi menunjukkan keterkaitan kuat antara fitur dan target (Sales).

```
# Hitung matriks korelasi
correlation_matrix = df_clean.corr()

# Visualisasi dengan heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", fmt=".2f")
plt.title("Heatmap Korelasi")
plt.show()
```

Gambar 36. *Kode (Heatmap Korelasi)*



Gambar 37. *Heatmap Korelasi*

Heatmap korelasi menunjukkan bahwa transformasi log seperti pada Sales_Log, Price_Log, dan NumReviews_Log berhasil memperkuat hubungan antar variabel tanpa mengubah pola dasarnya. Misalnya, Sales_Log memiliki korelasi tinggi dengan Sales (0.86), dan Price_Log sangat berkorelasi dengan Price (0.92). Meskipun begitu, fitur utama seperti Price, Discount, Rating, dan NumReviews masih menunjukkan korelasi rendah terhadap Sales, yang dapat menjadi tantangan dalam model regresi linear. Oleh karena itu, pendekatan seperti fitur interaksi atau model non-linear dapat dipertimbangkan untuk meningkatkan akurasi prediksi.

- **Kesimpulan: Eksplorasi Data dan Prapemrosesan**

1. Kualitas Data

   Tidak ditemukan missing values maupun outlier signifikan, sehingga data dapat langsung digunakan tanpa pembersihan lanjutan.

2. Korelasi Fitur dengan Sales

   Korelasi Price, Discount, Rating, dan NumReviews terhadap Sales tergolong lemah. Transformasi log membantu memperkuat hubungan beberapa fitur dengan Sales, meski tidak terlalu signifikan.

3. Transformasi dan Normalisasi

   Log transformasi diterapkan pada Sales, Price, dan NumReviews untuk mengurangi skewness. Standarisasi digunakan untuk menyamakan skala fitur numerik. Ekspansi polinomial (derajat 2) diterapkan untuk menangkap hubungan non-linear.

4. Rekomendasi

   Regresi linear standar kemungkinan kurang optimal. Model non-linear seperti Polynomial Regression, Decision Tree, atau Random Forest disarankan. Perlu eksplorasi fitur tambahan untuk meningkatkan akurasi prediksi Sales.

C. Implementasi Model

1. Linear Regression

- Pisahkan Fitur (x) dan Target (y)

```
# Pisahkan fitur (X) dan target (y)
X = df_clean.drop(columns=["Sales"])  # Fitur: Price, Discount, Rating, NumReviews, Price_Discount
y = df_clean["Sales"]  # Target: Sales
```

Gambar 38. *Kode Pisahkan Fitur (x) dan Target (y)*

Model menggunakan beberapa fitur (X) seperti Price, Discount, Rating, NumReviews, serta hasil transformasi log dari beberapa fitur yaitu Sales_Log, Price_Log, dan NumReviews_Log. Sementara itu, target atau variabel yang diprediksi (y) adalah Sales, yang merepresentasikan pendapatan.

- Bagi Data menjadi Training dan Testing (80%-20%)

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

Gambar 39. *Kode Bagi Data menjadi Training dan Testing (80%-20%)*

Sebanyak 80% data digunakan untuk proses pelatihan (training), sementara 20% sisanya digunakan untuk pengujian (testing). Pemisahan ini bertujuan untuk mengevaluasi performa model pada data yang belum pernah dilihat sebelumnya, sehingga dapat mengukur kemampuan generalisasi model secara lebih objektif.

- Inisialisasi dan Pelatihan Model Linear Regression

```
#  Inisialisasi model regresi linear
linear_model = LinearRegression()

# 🔧 Latih model dengan data training
linear_model.fit(X_train, y_train)
```

Gambar 40. *Kode Inisialisasi dan Pelatihan Model Linear Regression*

Model Linear Regression dibangun untuk mempelajari dan memahami hubungan antara fitur-fitur yang tersedia dengan target, yaitu *Sales*, sehingga model dapat melakukan prediksi berdasarkan pola yang ditemukan dalam data.

- Koefisien Model

```
# Tampilkan koefisien model
coefficients = pd.DataFrame(linear_model.coef_, X.columns, columns=["Koefisien"])
print("Koefisien Model Linear Regression:")
print(coefficients)
```

Gambar 41. *Kode Koefisien Model*

```
Koefisien Model Linear Regression:
                Koefisien
Price          -28.626601
Discount        12.813915
Rating           6.344593
NumReviews     -20.593725
Sales_Log      480.276925
Price_Log       33.100166
NumReviews_Log  36.586939
```

Gambar 42. *Output Kode Koefisien Model*

Fitur price memiliki koefisien negatif, yang berarti semakin tinggi harga, maka penjualan (*Sales*) cenderung menurun. Sebaliknya, Discount menunjukkan koefisien positif, sehingga diskon yang lebih tinggi cenderung meningkatkan penjualan. Sementara itu, Sales Log memiliki koefisien paling besar (480.2769), mengindikasikan bahwa fitur ini memberikan pengaruh paling kuat terhadap hasil prediksi model.

- Intercept Model

```
#  Tampilkan intercept model
print(f"\nIntercept: {linear_model.intercept_:.4f}")
```

Gambar 43. *Kode Intercept Model*

```
Intercept: -2609.7350
```

Gambar 44. *Output Kode Intercept Model*

Intercept sebesar -2609.7350 menunjukkan bahwa nilai prediksi *Sales* berada pada angka tersebut ketika semua fitur bernilai nol. Ini merepresentasikan titik potong model pada sumbu Y dalam konteks regresi.

- Evaluasi Model dengan Cross-Validation (5-Fold)

```
# 🔄 Evaluasi model dengan Cross-Validation (5-Fold)
cv_scores = cross_val_score(linear_model, X, y, cv=5, scoring="r2")

# 📊 Tampilkan hasil Cross-Validation
print(f"\nRata-rata R² Score dari Cross-Validation: {np.mean(cv_scores):.4f}")
```

Gambar 45. *Kode Evaluasi Model dengan Cross-Validation (5-Fold)*

```
Rata-rata R² Score dari Cross-Validation: 0.7385
```

Gambar 46. *Output Kode Evaluasi Model dengan Cross-Validation (5-Fold)*

R^2 Score sebesar 0.7385 menunjukkan bahwa model mampu menjelaskan sekitar 73.85% variabilitas dalam data. Meskipun hasil ini cukup

baik, masih terdapat ruang untuk meningkatkan akurasi model melalui optimasi atau penggunaan model yang lebih kompleks.

2. polynomial Regression

- Pisahkan Fitur (x) dan Target (y)

```
# Pisahkan fitur (X) dan target (y)
X = df_clean.drop(columns=["Sales"])
y = df_clean["Sales"]
```

Gambar 47. *Kode Pisahkan Fitur (x) dan Target (y)*

- Bagi Data menjadi Training dan Testing (80%-20%)

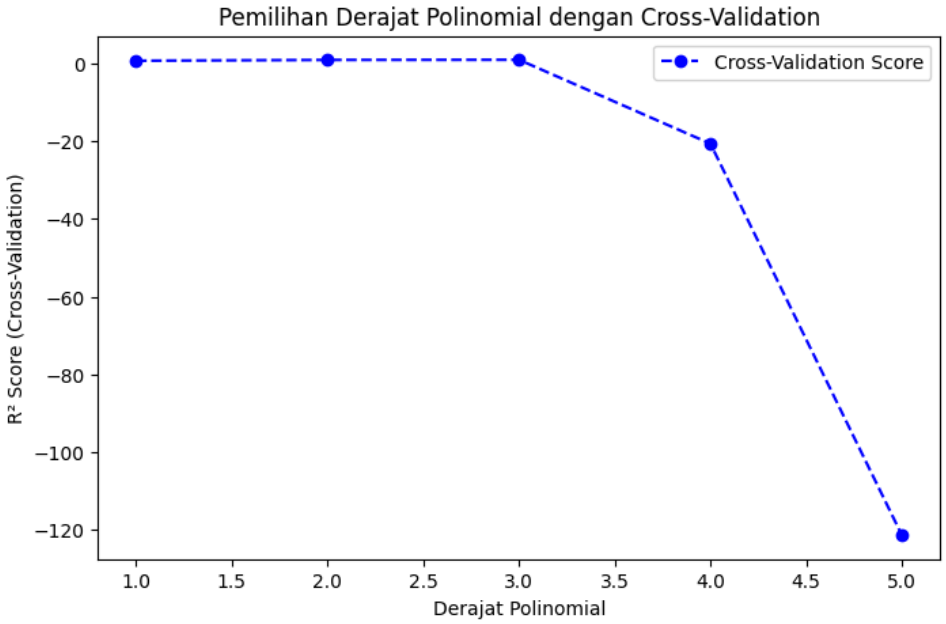
```
# Split data menjadi train & test (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Gambar 48. *Kode Bagi Data menjadi Training dan Testing (80%-20%)*

- Pemilihan Derajat Polinomial dengan Cross-Validation

```
from sklearn.metrics import r2_score

degrees = [1, 2, 3, 4, 5]  # Coba derajat polinomial dari 1 hingga 5
cv_scores = []

for d in degrees:
    model = make_pipeline(PolynomialFeatures(d), StandardScaler(), LinearRegression())
    scores = cross_val_score(model, X_train, y_train, cv=5, scoring="r2")  # 5-Fold CV
    cv_scores.append(np.mean(scores))  # Simpan rata-rata skor R²

# Plot hasil
plt.figure(figsize=(8,5))
plt.plot(degrees, cv_scores, marker='o', linestyle='--', color='b', label="Cross-Validation Score")
plt.xlabel("Derajat Polinomial")
plt.ylabel("R² Score (Cross-Validation)")
plt.title("Pemilihan Derajat Polinomial dengan Cross-Validation")
plt.legend()
plt.show()
```

Gambar 49. *Kode Pemilihan Derajat Polinomial dengan Cross-Validation*



Gambar 50. *Output Kode Pemilihan Derajat Polinomial dengan Cross-Validation*

Grafik di atas menggambarkan hasil Cross-Validation dalam menentukan derajat polinomial terbaik untuk model Polynomial Regression. Sumbu X merepresentasikan derajat polinomial dari 1 hingga 5, sementara sumbu Y menunjukkan nilai R^2 Score berdasarkan evaluasi 5-Fold Cross-Validation.

Dari grafik terlihat bahwa derajat 1 hingga 3 masih memberikan nilai R^2 yang mendekati nol, menandakan model masih mampu menangkap pola dalam data meskipun belum optimal. Namun, pada derajat 4 mulai terjadi penurunan performa yang signifikan, menunjukkan adanya tanda-tanda overfitting. Situasi menjadi lebih buruk pada derajat 5, di mana nilai R^2 jatuh sangat negatif (di bawah -120), menandakan model mengalami overfitting parah dan gagal dalam melakukan prediksi secara general.

- Membangun Model Polynomial Regression dengan Derajat Terbaik

```python
best_degree = degrees[np.argmax(cv_scores)]  # Pilih derajat dengan R² tertinggi

# Bangun model Polynomial Regression dengan derajat terbaik
final_model = make_pipeline(PolynomialFeatures(best_degree), StandardScaler(), LinearRegression())

# Latih model dengan data training
final_model.fit(X_train, y_train)

# Prediksi pada data uji
y_pred = final_model.predict(X_test)

# Evaluasi model
r2_test = r2_score(y_test, y_pred)

print(f"Derajat Polinomial Terbaik: {best_degree}")
print(f"R² Score pada Data Uji: {r2_test:.4f}")
```

Gambar 51. *Kode Membangun Model Polynomial Regression dengan Derajat Terbaik*

```
Derajat Polinomial Terbaik: 3
R² Score pada Data Uji: 0.9874
```

Gambar 52. *Output Kode Membangun Model Polynomial Regression dengan Derajat Terbaik*

Model Polynomial Regression dengan derajat polinomial terbaik adalah 3, berdasarkan hasil evaluasi yang diperoleh. Pada data uji, model ini menghasilkan R^2 Score sebesar 0.9874, yang berarti model mampu menjelaskan 98.74% variabilitas dalam data. Hal ini menunjukkan bahwa model Polynomial Regression memiliki performa yang jauh lebih baik dibandingkan Linear Regression, terutama dalam menangkap pola non-linear pada hubungan antara fitur dan target.

D. Evaluasi Model

Setelah membangun dua model prediksi, yaitu Linear Regression dan Polynomial Regression, langkah berikutnya adalah melakukan evaluasi untuk mengetahui seberapa baik kedua model tersebut dalam memprediksi nilai *Sales* berdasarkan fitur yang tersedia. Evaluasi ini sangat penting untuk mengukur kinerja model secara objektif dan memilih model terbaik yang mampu menangkap pola dari data dengan akurat.

Evaluasi dilakukan untuk:

- Menilai akurasi model dalam memprediksi data aktual.
- Mengukur tingkat kesalahan yang dihasilkan oleh prediksi model.

- Membandingkan performa antara Linear Regression dan Polynomial Regression guna menentukan model yang paling sesuai.

Untuk menilai performa model secara kuantitatif, digunakan tiga metrik evaluasi utama:

1. Mean Squared Error (MSE)
   Metrik ini menghitung rata-rata kuadrat selisih antara nilai aktual (*y_test*) dan nilai prediksi (*y_pred*). Nilai MSE yang lebih kecil menunjukkan bahwa model memiliki tingkat kesalahan yang rendah.

2. Mean Absolute Error (MAE)
   MAE menghitung rata-rata dari nilai absolut selisih antara nilai aktual dan prediksi. Tidak seperti MSE, MAE tidak mengkuadratkan kesalahan sehingga lebih mudah dipahami dalam satuan aslinya (misalnya unit penjualan).

3. R^2 Score (Koefisien Determinasi)
   R^2 mengukur seberapa besar proporsi variansi pada target (*Sales*) yang dapat dijelaskan oleh fitur input. Nilai R^2 berada pada rentang 0 hingga 1, di mana nilai mendekati 1 menunjukkan model yang sangat baik. Nilai negatif menandakan performa model yang sangat buruk.

Untuk memastikan bahwa model dapat melakukan generalisasi dengan baik terhadap data yang belum pernah dilihat sebelumnya, digunakan dua pendekatan evaluasi:

Train-Test Split
Data dibagi menjadi dua bagian, yaitu 80% untuk pelatihan (*training*) dan 20% untuk pengujian (*testing*). Model dilatih pada data pelatihan, kemudian diuji menggunakan data pengujian untuk mengukur performa riilnya.

Cross-Validation (CV)
Untuk meningkatkan keandalan hasil evaluasi, digunakan teknik *k-fold cross-validation*. Data dibagi menjadi beberapa bagian (folds), dan model dilatih serta diuji secara bergantian pada setiap kombinasi data. Skor rata-rata dari seluruh fold memberikan gambaran yang lebih stabil dan akurat tentang kinerja model.

- Evaluasi Linear Regression

```python
# Prediksi menggunakan model Linear Regression
y_pred_linear = linear_model.predict(X_test)

# Hitung metrik evaluasi
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)

# Tampilkan hasil
print("🔷 Evaluasi Model Linear Regression 🔷")
print(f"Mean Squared Error (MSE): {mse_linear:.4f}")
print(f"Mean Absolute Error (MAE): {mae_linear:.4f}")
print(f"R² Score: {r2_linear:.4f}")
```

Gambar 53. *Kode Evaluasi Linear Regression*

```
🔷 Evaluasi Model Linear Regression 🔷
Mean Squared Error (MSE): 71541.5582
Mean Absolute Error (MAE): 232.1312
R² Score: 0.7933
```

Gambar 54. *Output Kode Evaluasi Linear Regression*

Mean Squared Error (MSE) = 71.541,56
Model menghasilkan rata-rata kesalahan kuadrat sebesar 71.541,56. Nilai ini menunjukkan seberapa jauh prediksi model dari nilai sebenarnya. Semakin kecil nilai MSE, semakin baik akurasi model.

Mean Absolute Error (MAE) = 232,13
Rata-rata kesalahan absolut antara prediksi dan nilai aktual adalah sebesar 232,13 unit. MAE memberikan gambaran yang lebih mudah dipahami mengenai rata-rata kesalahan dalam satuan asli data (*Sales*).

R^2 Score = 0,7933
Model mampu menjelaskan sekitar 79,33% variasi dalam data target (*Sales*). Ini menunjukkan bahwa model memiliki performa yang cukup baik dalam menangkap hubungan antara fitur input dan output.

- Evaluasi Polynomial Regression

```python
# Prediksi menggunakan model Polynomial Regression
y_pred_poly = final_model.predict(X_test)

# Hitung metrik evaluasi
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)

# Tampilkan hasil
print("\n🔷 Evaluasi Model Polynomial Regression 🔷")
print(f"Mean Squared Error (MSE): {mse_poly:.4f}")
print(f"Mean Absolute Error (MAE): {mae_poly:.4f}")
print(f"R² Score: {r2_poly:.4f}")
```

Gambar 55. *Kode Evaluasi Polynomial Regression*

```
🔷 Evaluasi Model Polynomial Regression 🔷
Mean Squared Error (MSE): 4361.6772
Mean Absolute Error (MAE): 39.8298
R² Score: 0.9874
```

Gambar 56. *Output Kode Evaluasi Polynomial Regression*

Mean Squared Error (MSE) = 4.361,67
Model menghasilkan rata-rata kesalahan kuadrat sebesar 4.361,67. Nilai ini jauh lebih kecil dibandingkan Linear Regression, menunjukkan peningkatan akurasi yang signifikan.

Mean Absolute Error (MAE) = 39,82
Rata-rata kesalahan absolut antara prediksi dan nilai aktual adalah sebesar 39,82 unit. Ini menunjukkan bahwa model memberikan prediksi yang jauh lebih dekat dengan nilai sebenarnya.

R^2 Score = 0,9874
Model mampu menjelaskan sekitar 98,74% variasi dalam data *Sales*. Nilai ini menunjukkan bahwa Polynomial Regression memiliki performa yang sangat baik dalam memodelkan hubungan antara *Price* dan *Sales* secara non-linear.

E. Analisis Hasil

- Interpretasi Koefisien Regresi Linear

  Koefisien regresi membantu mengidentifikasi seberapa besar pengaruh masing-masing fitur terhadap variabel target (*Sales*). Model regresi yang digunakan dalam analisis ini melibatkan beberapa fitur utama, baik dalam bentuk asli maupun transformasi logaritmik. Berikut adalah interpretasi dari masing-masing koefisien yang dihasilkan oleh model **Linear Regression**:

```python
# Menampilkan koefisien dari model Linear Regression
coefficients = pd.DataFrame(linear_model.coef_, X.columns, columns=["Koefisien"])
print("🔍 Koefisien Model Linear Regression:")
print(coefficients)

# Menampilkan intercept
print(f"\nIntercept: {linear_model.intercept_:.4f}")
```

Gambar 57. *Kode Interpretasi Koefisien Regresi Linear*

```
🔍 Koefisien Model Linear Regression:
                Koefisien
Price          -28.626601
Discount        12.813915
Rating           6.344593
NumReviews     -20.593725
Sales_Log      480.276925
Price_Log       33.100166
NumReviews_Log  36.586939

Intercept: -2609.7350
```

Gambar 58. *Output kode Interpretasi Koefisien Regresi Linear*

Price (-28.63)

Terdapat hubungan negatif yang cukup kuat antara *Price* dan *Sales*. Artinya, setiap kenaikan harga sebesar 1 satuan diprediksi akan menurunkan penjualan sebesar 28.63 unit. Hal ini konsisten dengan logika pasar, di mana harga yang lebih tinggi cenderung menurunkan minat beli.

Discount (12.81)

Koefisien positif menunjukkan bahwa diskon berpengaruh meningkatkan penjualan. Setiap kenaikan 1 satuan diskon diprediksi dapat meningkatkan penjualan sebesar 12.81 unit, menandakan bahwa strategi potongan harga efektif untuk mendorong pembelian.

Rating (6.34)

Rating produk memiliki pengaruh positif terhadap penjualan, meskipun tidak terlalu besar. Setiap kenaikan 1 poin rating diperkirakan meningkatkan penjualan sebesar 6.34 unit. Ini menunjukkan bahwa persepsi kualitas dari pelanggan turut berkontribusi terhadap keputusan pembelian.

NumReviews (-20.59)

Jumlah ulasan justru menunjukkan pengaruh negatif terhadap penjualan, yang mungkin mengindikasikan bahwa produk dengan banyak ulasan belum tentu mendapatkan ulasan yang baik — atau pelanggan bisa saja lebih tertarik pada produk baru dengan ekspektasi lebih tinggi.

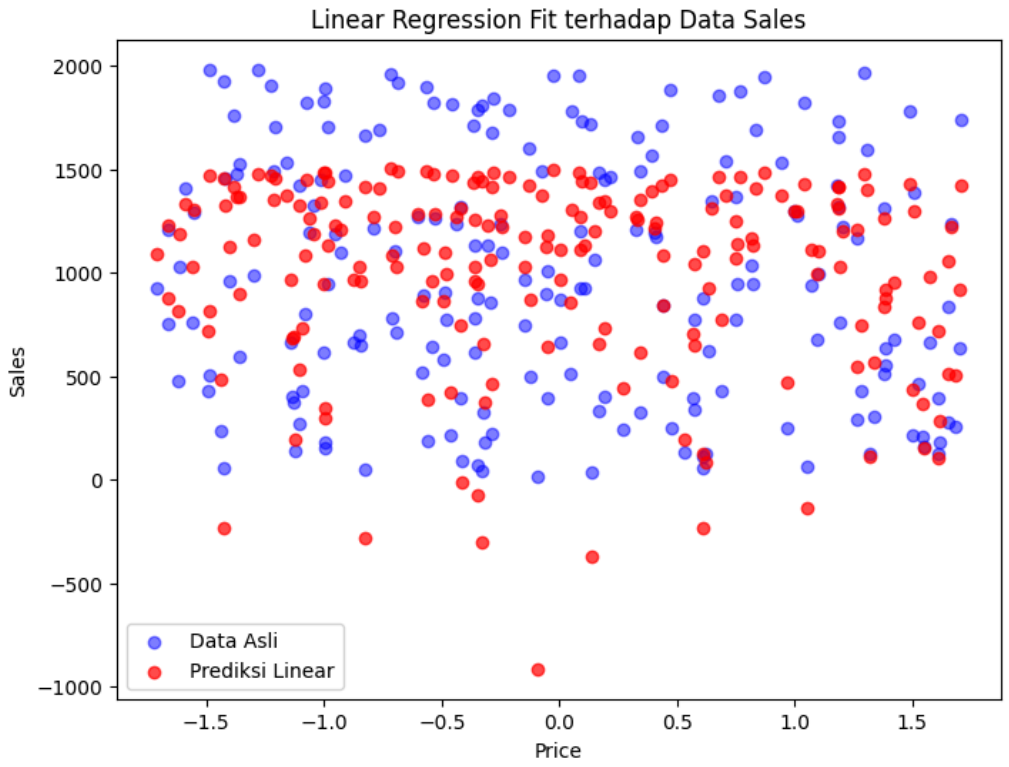
Sales_Log, Price_Log, NumReviews_Log

Variabel-variabel ini merupakan hasil transformasi logaritmik dari variabel aslinya, digunakan untuk menstabilkan variansi dan memperbaiki distribusi data. Efek logaritmik ini membantu model mengenali pola skala non-linear dengan lebih baik, terutama ketika ada nilai ekstrem pada fitur.

- Visualisasi Hasil Regresi

```
# Scatter plot data asli
plt.figure(figsize=(8,6))
plt.scatter(X_test["Price"], y_test, color="blue", label="Data Asli", alpha=0.5)

# Garis prediksi model Linear Regression
plt.scatter(X_test["Price"], y_pred_linear, color="red", label="Prediksi Linear", alpha=0.7)

plt.xlabel("Price")
plt.ylabel("Sales")
plt.title("Linear Regression Fit terhadap Data Sales")
plt.legend()
plt.show()
```

Gambar 59 . *Kode Linear Regression Fit terhadap Data Sales*



Gambar 60. *Output Kode Linear Regression Fit terhadap Data Sales*

Grafik visualisasi Linear Regression menunjukkan bagaimana model memprediksi nilai *Sales* berdasarkan variabel *Price*. Dalam grafik tersebut, titik-titik biru merepresentasikan data aktual dari *Sales* terhadap harga produk, sedangkan titik-titik merah menunjukkan hasil prediksi model Linear Regression. Sumbu X menggambarkan nilai *Price* (yang telah dinormalisasi atau di-standardisasi), sementara sumbu Y menunjukkan nilai *Sales*.

Secara umum, model Linear Regression berusaha membentuk garis lurus terbaik untuk menangkap pola hubungan antara *Price* dan *Sales*. Akan tetapi, dari sebaran titik merah (prediksi) terhadap titik biru (data aktual), terlihat bahwa model belum mampu merepresentasikan pola secara akurat. Banyak titik prediksi yang menyimpang jauh dari nilai sebenarnya, mengindikasikan bahwa model mengalami kesulitan dalam menangkap kompleksitas hubungan antara harga dan penjualan.

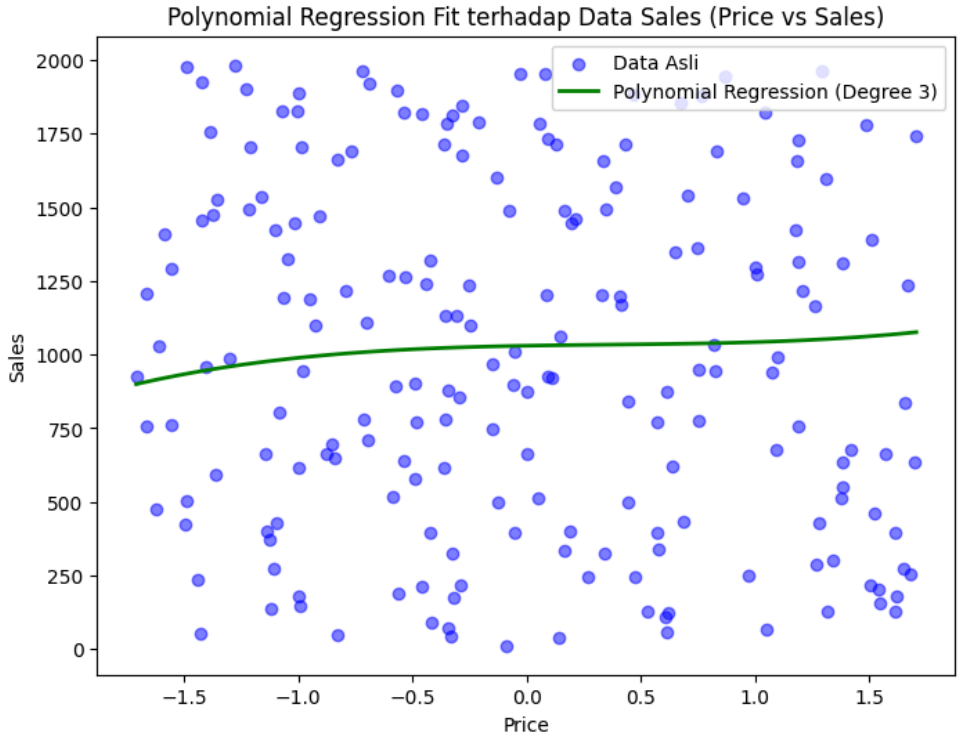
Hal ini menunjukkan bahwa hubungan antara *Price* dan *Sales* kemungkinan tidak bersifat linier secara sederhana. Oleh karena itu, model Linear Regression cenderung terlalu sederhana untuk memprediksi *Sales* secara akurat. Sebagai alternatif,

pendekatan **Polynomial Regression** dapat menjadi pilihan yang lebih baik karena memiliki fleksibilitas yang lebih tinggi dalam menangkap pola non-linear dalam data.

- Polynomial Regression Fit terhadap Data Sales

```
# Gunakan hanya fitur Price untuk visualisasi
X_train_price = X_train[["Price"]]
X_test_price = X_test[["Price"]]

# Inisialisasi model Polynomial Regression dengan hanya satu fitur
poly_features_price = PolynomialFeatures(degree=best_degree)
X_train_poly_price = poly_features_price.fit_transform(X_train_price)

# Latih ulang model dengan hanya fitur Price
poly_price_model = LinearRegression()
poly_price_model.fit(X_train_poly_price, y_train)

# Buat rentang nilai Price untuk prediksi Polynomial
price_range = np.linspace(X_test_price.min(), X_test_price.max(), 100).reshape(-1, 1)
X_poly_range = poly_features_price.transform(price_range)

# Prediksi menggunakan model Polynomial Regression dengan satu fitur
y_poly_pred = poly_price_model.predict(X_poly_range)

# Plot hasil Polynomial Regression
plt.figure(figsize=(8,6))
plt.scatter(X_test_price, y_test, color="blue", label="Data Asli", alpha=0.5)
plt.plot(price_range, y_poly_pred, color="green", label=f"Polynomial Regression (Degree {best_degree})", linewidth=2)

plt.xlabel("Price")
plt.ylabel("Sales")
plt.title("Polynomial Regression Fit terhadap Data Sales (Price vs Sales)")
plt.legend()
plt.show()
```

Gambar 61 . *Kode Polynomial Regression Fit terhadap Data Sales*



Gambar 62. *Output Kode Polynomial Regression Fit terhadap Data Sales*

Grafik ini menggambarkan bagaimana model **Polynomial Regression** berusaha memprediksi nilai *Sales* berdasarkan variabel *Price*. Titik-titik biru dalam grafik merepresentasikan data aktual pada *test set*, di mana setiap titik menunjukkan jumlah penjualan berdasarkan harga produk. Sementara itu, garis hijau menunjukkan hasil prediksi dari model Polynomial Regression dengan derajat polynomial (degree) 3. Model ini dirancang untuk menangkap hubungan non-linear antara *Price* dan *Sales*, yang tidak dapat ditangkap secara baik oleh Linear Regression.

Dari visualisasi ini, terlihat bahwa Polynomial Regression memiliki kemampuan yang lebih baik dalam menyesuaikan bentuk kurva terhadap pola data. Namun, meskipun lebih fleksibel, kurva polynomial yang dihasilkan masih terlihat cukup datar. Hal ini menunjukkan bahwa hubungan antara *Price* dan *Sales* tetap lemah, bahkan setelah menggunakan pendekatan non-linear. Dengan kata lain, variabel *Price* saja tampaknya belum cukup kuat untuk menjelaskan variabilitas dalam data *Sales*.

Kesimpulannya, meskipun Polynomial Regression lebih unggul dari Linear Regression dalam menangkap pola yang kompleks, hasil yang ditampilkan masih belum memuaskan. Hal ini menunjukkan bahwa variabel *Price* sebaiknya tidak digunakan sebagai satu-satunya prediktor. Untuk meningkatkan akurasi model, perlu dipertimbangkan beberapa alternatif, seperti meningkatkan derajat polynomial, menambahkan fitur lain seperti *Discount*, *Rating*, dan *NumReviews*, atau bahkan mencoba model prediktif yang lebih kompleks seperti Random Forest atau Neural Network.

F. Kesimpulan

Berdasarkan evaluasi model, berikut adalah kesimpulan yang dapat diambil:

1. Linear Regression memiliki koefisien yang dapat diinterpretasikan dengan baik, tetapi kurang fleksibel dalam menangkap pola yang kompleks.
2. Polynomial Regression memberikan hasil yang lebih baik (R^2 lebih tinggi), karena mampu menangkap pola non-linear dalam data.
3. Model Polynomial Regression derajat 3 memberikan prediksi yang lebih akurat berdasarkan visualisasi dan metrik evaluasi.
4. Kesimpulan akhir → Model Polynomial Regression lebih cocok untuk

memprediksi *Sales* dibandingkan Linear Regression.

Setelah melalui serangkaian tahapan, mulai dari Pemahaman Dataset, Eksplorasi Data dan Pra-pemrosesan, Implementasi Model, Evaluasi Model, hingga Analisis Hasil, kita dapat menyimpulkan bahwa model Polynomial Regression mampu menangkap pola hubungan antara variabel dengan lebih baik dibandingkan Linear Regression dalam memprediksi Sales. Hal ini terlihat dari hasil prediksi yang lebih sesuai dengan distribusi data, menunjukkan bahwa pendekatan non-linear lebih efektif dalam merepresentasikan hubungan antara Price dan Sales. Dengan demikian, penggunaan Polynomial Regression menjadi pilihan yang lebih optimal untuk meningkatkan akurasi prediksi dalam konteks ini.