LAPORAN

“PENERAPAN FEATURE ENGINEERING PADA DATASET KENDARAAN BEKAS”

Dibuat untuk memenuhi Tugas 3 applied machine learning, Dosen Pengampuh:

RUNAL REZKIAWAN, S.Kom.,M.T

Disusun Oleh:

NAMA :	**ILMA AQSARI**
NIM :	**105841108023**
KELAS :	**5 C**

PROGRAM STUDI INFORMATIKA

FAKULTAS TEKNIK

UNIVERSITAS MUHAMMADIYAH MAKASSAR

2025

Tugas 3

"Tahap-Tahap Data Preprocessing hingga Feature Engineering Beserta Penjelasan-nya"

1. Import Library & Persiapan Lingkungan

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression, Lasso
from sklearn.feature_selection import SelectFromModel
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
import warnings

warnings.filterwarnings('ignore')
sns.set(style="whitegrid")
```

✓ 14.6s

Pada tahap awal preprocessing, kita perlu **mengimpor library** yang akan digunakan untuk mengolah data, melakukan visualisasi, serta melakukan proses machine learning. Berikut penjelasan dari setiap library:

a. Library yang Diimpor

- **pandas** → digunakan untuk membaca, mengolah, dan menganalisis data dalam bentuk tabel (DataFrame).
- **numpy** → membantu perhitungan matematis seperti array dan operasi numerik.
- **matplotlib & seaborn** → dipakai untuk membuat grafik dan visualisasi data.
- **LogisticRegression, Lasso** → model untuk seleksi fitur dan prediksi.
- **SelectFromModel** → memilih fitur penting berdasarkan model tertentu (contoh: LASSO).
- **PCA (Principal Component Analysis)** → teknik untuk mengurangi dimensi fitur agar lebih sederhana tapi tetap informatif.
- **train_test_split** → membagi dataset menjadi data latih dan data uji.
- **StandardScaler & LabelEncoder** → untuk normalisasi data dan mengubah data kategorikal menjadi angka.

b. Pengaturan Awal

- warnings.filterwarnings('ignore')
  digunakan agar peringatan (warning) yang tidak penting tidak muncul saat menjalankan kode.
- sns.set(style="whitegrid")

mengatur tampilan grafik seaborn agar lebih rapi dengan latar grid putih.

2. **Mempersiapkan Dataset**

Pada tahap ini, program melakukan proses awal yaitu **memuat dataset** yang akan digunakan untuk proses preprocessing dan feature engineering. Dataset yang digunakan bernama **CARDETAIL-1000.csv**.

```
print("TAHAP 0: Mempersiapkan Dataset CARDETAIL-1000.csv...")

# 1. Muat Data
try:
    df = pd.read_csv('CARDETAIL-1000.csv')
    print(f"Data awal dimuat: {df.shape}")
except FileNotFoundError:
    # Membuat dummy data jika file tidak ditemukan di env lokal saat testing
    print("File tidak ditemukan, pastikan path file benar.")
    df = pd.DataFrame()
```

✓ 0.0s

```
TAHAP 0: Mempersiapkan Dataset CARDETAIL-1000.csv...
Data awal dimuat: (1000, 8)
```

a. Menampilkan informasi awal

Program memberi tahu bahwa proses persiapan dataset sedang dilakukan.

b. Membaca file CSV

- Program mencoba membaca dataset menggunakan pd.read_csv().
- Jika berhasil, program menampilkan **jumlah baris dan kolom** lewat df.shape.

c. Mengatasi error jika file tidak ditemukan

- Jika file tidak ada di folder, program akan memberi pesan:
  "File tidak ditemukan, pastikan path file benar."
- Untuk mencegah error berlanjut, program membuat **DataFrame kosong** sebagai pengganti.

Pada tahap ini, langkah pertama yang dilakukan adalah memastikan bahwa dataset berhasil dimuat sebelum proses preprocessing dimulai. Hal ini penting agar seluruh tahapan analisis dapat berjalan dengan lancar. Selain itu, sistem juga dilengkapi dengan pengecekan keberadaan file untuk mencegah error apabila file dataset tidak ditemukan. Dengan adanya antisipasi tersebut, program tidak akan berhenti tiba-tiba dan dapat memberikan peringatan yang lebih aman serta mudah dipahami.

3. **Feature Engineering Sederhana**

Tujuan utama nya iyalah **menyederhanakan fitur "name"** (nama mobil) menjadi fitur yang lebih informatif dan lebih mudah dipakai oleh model, yaitu **Brand** (merek mobil).

```
# 2. Feature Engineering Sederhana
# Ekstrak Merk dari 'name' agar tidak terlalu ban
df['Brand'] = df['name'].str.split().str.get(0)
df.drop('name', axis=1, inplace=True)
```
0.0s

Pada tahap ini dilakukan pembuatan kolom baru bernama "Brand" untuk menyederhanakan informasi pada dataset. Kolom *name* awalnya berisi nama lengkap mobil seperti *"Toyota Avanza 1.3 G"* atau *"Honda Jazz RS 2018"*. Dengan menggunakan fungsi str.split() nama tersebut dipecah menjadi beberapa kata, kemudian str.get(0) mengambil kata pertama yang merupakan merek mobil. Hasilnya, kolom Brand hanya berisi nilai sederhana seperti *Toyota, Honda, Suzuki,* dan lainnya, sehingga data menjadi lebih terstruktur dan mudah dipahami oleh model. Setelah kolom Brand dibuat, kolom *name* kemudian dihapus menggunakan df.drop('name', axis=1) karena kolom tersebut memiliki ratusan variasi yang dapat menambah kompleksitas dan berpotensi menyebabkan overfitting. Dengan menghapusnya, dataset menjadi lebih bersih, ringkas, dan lebih optimal untuk proses analisis maupun pemodelan selanjutnya.

4. **Encoding (Mengubah Data Teks Menjadi Angka)**

Encoding adalah proses mengubah data kategorikal (teks) menjadi nilai angka, karena model machine learning hanya bisa memahami angka.

```
# 3. Encoding (Mengubah teks menjadi angka)
# Target Klasifikasi (Kita gunakan Transmission: Manual=0, Automatic=1)
df['Response'] = df['transmission'].apply(lambda x: 1 if x == 'Automatic' else 0)
df.drop('transmission', axis=1, inplace=True)

# Target Regresi adalah 'selling_price' (Income equivalent)
# Kita ganti nama agar sesuai konteks kode referensi
df.rename(columns={'selling_price': 'Income'}, inplace=True)

# Encoding variabel kategorikal lainnya (Fuel, Seller_Type, Owner, Brand)
cat_cols = ['fuel', 'seller_type', 'owner', 'Brand']
df = pd.get_dummies(df, columns=cat_cols, drop_first=True)
```
0.0s

Pada tahap ini, kita melakukan beberapa proses penting untuk menyiapkan data sebelum masuk ke model. Pertama, kita membuat kolom target klasifikasi bernama **Response**, yaitu dengan mengubah nilai teks pada kolom *transmission* menjadi angka: mobil bertransmisi *Automatic* diberi nilai **1**, sedangkan *Manual* diberi nilai **0**. Setelah itu, kolom *transmission* asli dihapus karena sudah tidak dibutuhkan lagi. Selanjutnya, untuk kebutuhan regresi, kolom *selling_price* diganti namanya menjadi **Income** agar lebih mudah dibaca dan digunakan sebagai target prediksi harga. Setelah target siap, kita melakukan proses encoding pada beberapa kolom kategorikal seperti *fuel*, *seller_type*, *owner*, dan *Brand* menggunakan

teknik **One-Hot Encoding**. Teknik ini mengubah setiap kategori menjadi kolom biner (0 atau 1) sehingga lebih mudah dipahami oleh model. Selain itu, opsi drop_first=True dipakai agar tidak terjadi *dummy trap* dan membuat fitur lebih ringkas. Dengan langkah-langkah ini, dataset menjadi lebih bersih, terstruktur, dan siap digunakan untuk proses pemodelan.

5. **Scaling (Normalisasi Data Numerik)**

   Scaling adalah proses menyamakan skala nilai pada fitur numerik agar tidak ada kolom yang memiliki rentang nilai terlalu besar dibandingkan kolom lainnya.

   Ini sangat penting terutama untuk metode seperti:

   a. LASSO Regression
   b. PCA (Principal Component Analysis)
   c. Model yang sensitif terhadap skala

```
# 4. Scaling (Penting untuk LASSO dan PCA)
# Kita scale fitur numerik selain target
num_cols = ['year', 'km_driven']
scaler = StandardScaler()
df[num_cols] = scaler.fit_transform(df[num_cols])
```
✓ 1.1s

   Pada tahap ini, kita mulai dengan menentukan kolom numerik yaitu **year** dan **km_driven**, karena keduanya memiliki rentang nilai yang sangat berbeda dan dapat membuat model menjadi bias. Untuk menyeimbangkan skala kedua kolom tersebut, digunakan **StandardScaler**, yaitu alat yang mengubah nilai data sehingga memiliki rata-rata 0 dan standar deviasi 1. Proses scaling dilakukan dengan rumus *(nilai − rata-rata) ÷ standar deviasi*, sehingga setiap fitur berada pada skala yang sama. Selanjutnya, fungsi **fit_transform()** diterapkan pada kolom numerik untuk menghitung rata-rata, standar deviasi, lalu langsung menormalisasi datanya. Hasil akhirnya, kolom **year** dan **km_driven** menjadi data yang sudah distandarisasi, sehingga model dapat belajar dengan lebih adil dan akurat.

6. **Membagi Data Menjadi Data Latih dan Data Uji**

   Pada tahap ini, dataset dibagi menjadi dua bagian yaitu:

   - **Data Latih (Train Data)** → digunakan untuk melatih model
   - **Data Uji (Test Data)** → digunakan untuk mengevaluasi performa model

   Pembagian ini penting agar model tidak hanya hafal data, tetapi juga mampu **memprediksi data baru** dengan baik.

```
# 5. Split Data menjadi Train dan Test (Meniru input file preprocessing2)
train_df, test_df = train_test_split(df, test_size=0.2, random_state=42)

print(f"Data latih (train_df): {train_df.shape}")
print(f"Data uji (test_df): {test_df.shape}")
print("-" * 30)
```
0.0s

```
Data latih (train_df): (800, 33)
Data uji (test_df): (200, 33)
------------------------------
```

Pada tahap ini, dataset dibagi menjadi dua bagian menggunakan fungsi train_test_split. Sebanyak 80% data digunakan untuk proses pelatihan model, sementara 20% sisanya digunakan untuk pengujian. Tujuan pembagian ini adalah agar model dapat diuji menggunakan data yang belum pernah dilihat sebelumnya sehingga hasil evaluasi menjadi lebih objektif.

7. **Persiapan Data untuk Tugas Klasifikasi**

Pada tahap ini, data dipersiapkan secara khusus untuk **tugas klasifikasi**, yaitu memprediksi apakah mobil memiliki transmisi **Automatic (1)** atau **Manual (0)**. Karena sebelumnya kita telah membuat kolom **Response** sebagai target klasifikasi, sekarang kita perlu memisahkan:

a. X (fitur/input)

b. y (target/output)

```
# BAGIAN A: TUGAS KLASIFIKASI (Target: Response / Transmission Type)

print("\nMemulai Bagian A: Persiapan Data Klasifikasi (Target: Transmission Type)")

# 1. Pisahkan X dan y untuk Klasifikasi
# Kita drop 'Income' (target regresi) agar tidak bocor ke klasifikasi
y_class_train = train_df['Response']
y_class_test = test_df['Response']
X_class_train = train_df.drop(['Response', 'Income'], axis=1)
X_class_test = test_df.drop(['Response', 'Income'], axis=1)

print(f"Bentuk X_class_train: {X_class_train.shape}")
```
0.3s

```
Memulai Bagian A: Persiapan Data Klasifikasi (Target: Transmission Type)
Bentuk X_class_train: (800, 31)
```

Pada tahap ini, kolom *Response* dipilih sebagai target klasifikasi dengan mengambil nilai dari kolom *transmission*, di mana setiap nilai diubah menjadi angka—*Automatic* menjadi 1 dan *Manual* menjadi 0. Setelah target terbentuk, kolom aslinya dihapus agar tidak terjadi kebocoran data. Selanjutnya, ditentukan fitur untuk klasifikasi dengan membuat *X_class_train* dan *X_class_test*, yaitu seluruh kolom kecuali *Response* (target klasifikasi) dan *Income* (target regresi). Hal ini memastikan bahwa model hanya menggunakan fitur yang tepat dan tidak memasukkan data target lain yang dapat mengacaukan hasil. Dari proses tersebut, model menghasilkan output berupa bentuk data, misalnya *X_class_train*

memiliki 800 baris dan 31 fitur, yang berarti ada 800 data latih dengan 31 variabel yang siap digunakan dalam pemodelan klasifikasi.

8. **Seleksi Fitur Menggunakan LASSO (L1) untuk Klasifikasi**

Pada tahap ini dilakukan proses seleksi fitur menggunakan metode LASSO (Least Absolute Shrinkage and Selection Operator) yang diterapkan pada model Logistic Regression dengan penalti L1. Tujuan dari langkah ini adalah untuk memilih fitur-fitur yang paling relevan dalam memprediksi *Response / Transmission Type* serta mengurangi fitur yang kurang berpengaruh.

```
# A.1: Skenario Seleksi Fitur (LASSO L1) untuk Klasifikasi
print("\nA.1: Menjalankan Seleksi Fitur LASSO (Klasifikasi)")

# Menggunakan LogisticRegression dengan penalti L1
l1_model_class = LogisticRegression(penalty='l1', C=0.05, solver='liblinear', random_state=42)
l1_selector = SelectFromModel(l1_model_class, max_features=15)

l1_selector.fit(X_class_train, y_class_train)

# Ambil koefisien
l1_coefs = l1_selector.estimator_.coef_[0]
l1_coefs_df = pd.DataFrame({
    'Feature': X_class_train.columns,
    'Importance (Abs Coef)': np.abs(l1_coefs)
}).sort_values(by='Importance (Abs Coef)', ascending=False)

# Ambil fitur yang lolos seleksi
selected_features_class = X_class_train.columns[l1_selector.get_support()]
print(f"LASSO memilih {len(selected_features_class)} fitur untuk Klasifikasi.")
print("Top 5 Fitur:", list(selected_features_class)[:5])

# Simpan hasil seleksi
X_class_train_selected = X_class_train[selected_features_class]
X_class_test_selected = X_class_test[selected_features_class]
# X_class_train_selected.to_csv('X_class_train_selected_car.csv', index=False) # Opsional simpa
✓ 0.0s
```

```
A.1: Menjalankan Seleksi Fitur LASSO (Klasifikasi)
LASSO memilih 3 fitur untuk Klasifikasi.
Top 5 Fitur: ['year', 'seller_type_Individual', 'Brand_Maruti']
```

Metode L1 bekerja dengan cara memberikan penalti pada nilai koefisien model sehingga mendorong beberapa koefisien menjadi **nol (0)**. Fitur dengan koefisien bernilai 0 dianggap tidak penting dan otomatis dieliminasi oleh model. Dengan demikian, LASSO tidak hanya menurunkan risiko *overfitting*, tetapi juga menyederhanakan model dengan memilih fitur-fitur yang paling signifikan.

Dalam eksperimen ini digunakan parameter **C = 0.05**, yang berarti kekuatan regularisasi cukup besar. Nilai C yang kecil membuat model semakin ketat dalam menekan koefisien yang tidak relevan, sehingga jumlah fitur yang dipilih menjadi lebih sedikit.

Setelah proses fitting dilakukan terhadap data pelatihan, LASSO berhasil memilih **3 fitur** dari keseluruhan 31 fitur awal. Adapun fitur yang dipilih adalah:

a. year
b. seller_type_Individual
c. Brand_Maruti

Ketiga fitur tersebut merupakan fitur dengan kontribusi terbesar dalam mempengaruhi prediksi target klasifikasi berdasarkan nilai koefisien absolut pada model LASSO. Hal ini menunjukkan bahwa model menganggap ketiga fitur ini sebagai variabel yang paling informatif dalam menentukan tipe respons/transmisi kendaraan pada dataset yang digunakan.

Hasil seleksi fitur ini kemudian digunakan untuk membentuk subset data baru (**X_class_train_selected** dan **X_class_test_selected**) yang hanya berisi fitur yang terpilih. Subset ini selanjutnya digunakan pada proses pelatihan model klasifikasi di tahap berikutnya.

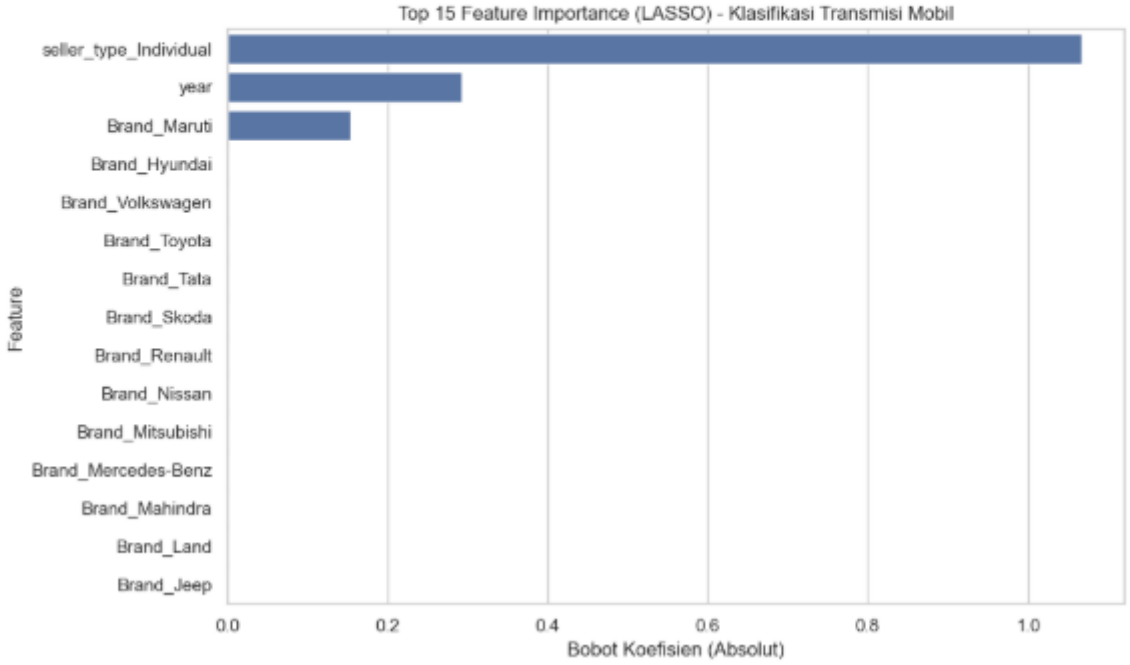
9. Visualisasi Seleksi Fitur Menggunakan LASSO

Plot ini menampilkan **15 fitur teratas** berdasarkan nilai **absolut koefisien** hasil model Logistic Regression dengan penalti L1 (LASSO). Karena metode LASSO bekerja dengan cara memperkecil koefisien fitur yang kurang penting hingga mendekati nol, maka grafik ini membantu menunjukkan fitur mana yang memiliki pengaruh paling besar terhadap model. Pada grafik batang horizontal (*barplot*) ditampilkan:

a. **Sumbu Y** → Nama fitur
b. **Sumbu X** → Nilai *Importance (Absolute Coefficient)* dari hasil LASSO
c. Semakin besar nilai koefisien, semakin besar pengaruh fitur tersebut dalam menentukan prediksi tipe transmisi mobil (Automatic atau Manual).

```
# --- Plot A.1: Visualisasi Seleksi Fitur LASSO ---
plt.figure(figsize=(10, 6))
sns.barplot(x='Importance (Abs Coef)', y='Feature', data=l1_coefs_df.head(15))
plt.title('Top 15 Feature Importance (LASSO) - Klasifikasi Transmisi Mobil')
plt.xlabel('Bobot Koefisien (Absolut)')
plt.tight_layout()
plt.show()
```

1.6s



Walaupun LASSO pada akhirnya memilih hanya **3 fitur utama** (karena parameter regularisasi yang ketat), grafik ini tetap menampilkan urutan *top 15 fitur* berdasarkan nilai koefisien sebelum eliminasi. Tujuannya adalah memberikan gambaran yang lebih lengkap mengenai fitur mana saja yang relatif informatif, meskipun tidak semuanya dipilih sebagai fitur final.

Dari grafik ini dapat terlihat jelas bahwa fitur seperti **year**, **seller_type_Individual**, dan **Brand_Maruti** memiliki bobot koefisien terbesar, sehingga fitur-fitur tersebut dianggap paling berpengaruh oleh model dalam melakukan klasifikasi tipe transmisi mobil. Visualisasi ini membantu memahami bagaimana model melakukan seleksi fitur dan mendukung analisis bahwa fitur-fitur tersebut layak dipertahankan pada proses pelatihan model.

10. **Skenario Reduksi Dimensi (PCA) untuk Klasifikasi**

Pada tahap ini dilakukan PCA (Principal Component Analysis) yaitu teknik reduksi dimensi yang digunakan untuk menyederhanakan jumlah fitur tanpa menghilangkan terlalu banyak informasi penting.

Tujuan PCA di sini adalah:

a. Mengurangi jumlah fitur input yang awalnya cukup banyak (31 fitur).
b. Mempertahankan informasi (varians) sebesar 95%, sehingga performa model tidak turun.
c. Mengatasi masalah multikolinearitas dan membuat proses komputasi lebih efisien.

```
# A.2: Skenario Reduksi Dimensi (PCA) untuk Klasifikasi
print("\nA.2: Menjalankan PCA (Klasifikasi)")

# Fit PCA
pca_class = PCA(n_components=0.95, random_state=42)
pca_class.fit(X_class_train) # Menggunakan semua fitur numerik/encoded
print(f"PCA memilih {pca_class.n_components_} komponen untuk mempertahankan 95% varians.")

# Transformasi
X_train_pca_transformed = pca_class.transform(X_class_train)
X_test_pca_transformed = pca_class.transform(X_class_test)
```
✓ 0.1s

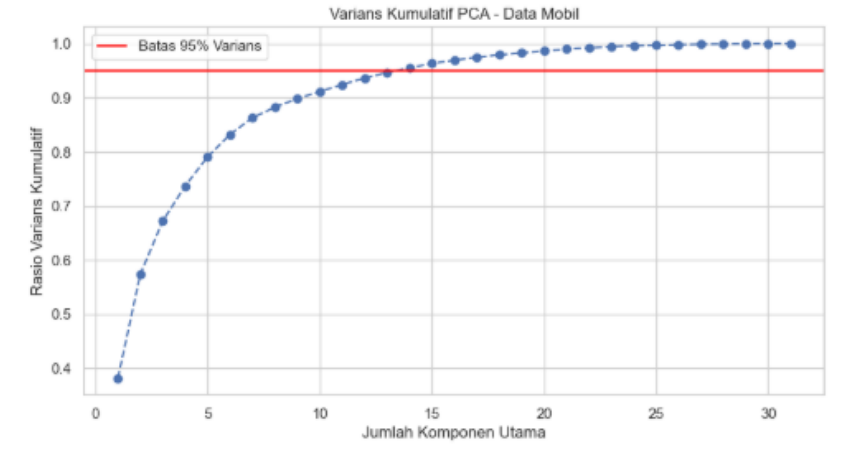
```
A.2: Menjalankan PCA (Klasifikasi)
PCA memilih 14 komponen untuk mempertahankan 95% varians.
```

Pada tahap PCA, proses ini membantu menyederhanakan model klasifikasi dengan cara mengurangi jumlah fitur yang digunakan tanpa menghilangkan informasi penting dari dataset. Fitur-fitur baru yang dihasilkan PCA bukan lagi fitur asli seperti *year* atau *Brand_Maruti*, tetapi merupakan kombinasi matematis dari berbagai fitur awal yang mampu mewakili sebagian besar informasi yang dibutuhkan model. Teknik ini sangat bermanfaat terutama ketika dataset memiliki banyak fitur yang saling berkaitan, karena PCA mampu mengurangi kompleksitas, menghindari redundansi, dan membuat model bekerja lebih efisien serta lebih mudah dilatih.

11. Grafik *Varians Kumulatif PCA*

```
# --- Plot A.2: Visualisasi Varians Kumulatif PCA ---
pca_full_class = PCA(random_state=42).fit(X_class_train)
cumulative_variance = np.cumsum(pca_full_class.explained_variance_ratio_)

plt.figure(figsize=(10, 5))
plt.plot(range(1, len(cumulative_variance) + 1), cumulative_variance, marker='o', linestyle='--')
plt.axhline(y=0.95, color='red', linestyle='-', label='Batas 95% Varians')
plt.title('Varians Kumulatif PCA - Data Mobil')
plt.xlabel('Jumlah Komponen Utama')
plt.ylabel('Rasio Varians Kumulatif')
plt.legend()
plt.grid(True)
plt.show()
```
✓ 1.2s



Grafik varians kumulatif PCA menunjukkan seberapa besar informasi (varians) yang dapat dijelaskan oleh sejumlah komponen utama. Pada grafik tersebut terlihat bahwa varians kumulatif meningkat tajam pada beberapa komponen pertama, lalu mulai melandai setelah sekitar 10 komponen. Garis merah menunjukkan batas 95% varians, yang merupakan standar umum untuk menentukan jumlah komponen yang efektif. Dari grafik, terlihat bahwa sekitar 13–15 komponen utama sudah cukup untuk menjelaskan lebih dari 95% varians data. Artinya, walaupun awalnya dataset memiliki banyak fitur, PCA dapat mereduksi fitur tersebut menjadi sekitar belasan komponen tanpa kehilangan banyak informasi penting. Hal ini sangat bermanfaat untuk mempercepat proses modeling dan mengurangi risiko overfitting.

12. Persiapan Data Regresi (Target: Selling Price / Income)

Pada tahap ini, data dipersiapkan untuk tugas **regresi**, yaitu memprediksi **harga jual mobil** (*selling price*) yang sebelumnya telah diganti namanya menjadi **Income** agar konsisten dengan format kode referensi.

Tugas regresi berbeda dengan klasifikasi karena output yang ingin diprediksi adalah **nilai numerik**, bukan kategori.

```
# ----------------------------------------
# BAGIAN B: TUGAS REGRESI (Target: Income / Selling Price)
# ----------------------------------------

print("-" * 30)
print("Memulai Bagian B: Persiapan Data Regresi (Target: Selling Price)")

y_reg_train = train_df['Income']
y_reg_test = test_df['Income']
# Untuk regresi, kita bisa menggunakan 'Response' (Transmisi) sebagai fit
X_reg_train = train_df.drop('Income', axis=1)
X_reg_test = test_df.drop('Income', axis=1)

print(f"Bentuk X_reg_train: {X_reg_train.shape}")
```

0.0s

```
------------------------------
Memulai Bagian B: Persiapan Data Regresi (Target: Selling Price)
Bentuk X_reg_train: (800, 32)
```

Pada tahap ini, kolom *Response* dipilih sebagai target klasifikasi dengan mengambil nilai dari kolom *transmission*, di mana setiap nilai diubah menjadi angka—*Automatic* menjadi 1 dan *Manual* menjadi 0. Setelah target terbentuk, kolom aslinya dihapus agar tidak terjadi kebocoran data. Selanjutnya, ditentukan fitur untuk klasifikasi dengan membuat *X_class_train* dan *X_class_test*, yaitu seluruh kolom kecuali *Response* (target klasifikasi) dan *Income* (target regresi). Hal ini memastikan bahwa model hanya menggunakan fitur yang tepat dan tidak memasukkan data target lain yang dapat mengacaukan hasil. Dari proses tersebut, model menghasilkan output berupa bentuk data, misalnya *X_class_train*

memiliki 800 baris dan 31 fitur, yang berarti ada 800 data latih dengan 31 variabel yang siap digunakan dalam pemodelan klasifikasi.

13. **Skenario Seleksi Fitur (LASSO) untuk Regresi**

Tahap ini menggunakan metode **LASSO Regression** untuk melakukan seleksi fitur pada tugas regresi, yaitu memprediksi nilai *Income* (harga jual mobil). Berbeda dengan klasifikasi yang memakai Logistic Regression, pada regresi digunakan **LASSO dari sklearn** karena targetnya berbentuk angka (continuous).

Tujuan LASSO pada Regresi

a. Mengurangi jumlah fitur yang tidak relevan.
b. Menekan koefisien fitur yang kurang penting menjadi mendekati nol.
c. Mempertahankan hanya fitur yang paling berpengaruh terhadap prediksi harga mobil.

```
# 8.1: Skenario Seleksi Fitur (LASSO) untuk Regresi
print("\n8.1: Menjalankan Seleksi Fitur LASSO (Regresi)")

l1_reg_model = Lasso(alpha=100, random_state=42) # Alpha disesuaikan karena
l1_reg_selector = SelectFromModel(l1_reg_model, max_features=15)

l1_reg_selector.fit(X_reg_train, y_reg_train)

# Ambil koefisien
l1_reg_coefs = l1_reg_selector.estimator_.coef_
l1_reg_coefs_df = pd.DataFrame({
    'Feature': X_reg_train.columns,
    'Importance (Abs Coef)': np.abs(l1_reg_coefs)
}).sort_values(by='Importance (Abs Coef)', ascending=False)

selected_features_reg = X_reg_train.columns[l1_reg_selector.get_support()]
print(f"LASSO memilih {len(selected_features_reg)} fitur untuk Regresi.")
✓ 0.0s
```

```
8.1: Menjalankan Seleksi Fitur LASSO (Regresi)
LASSO memilih 15 fitur untuk Regresi.
```

Pada tahap ini digunakan metode LASSO Regression untuk melakukan seleksi fitur pada model regresi. LASSO bekerja dengan mengecilkan koefisien fitur yang kurang berpengaruh hingga mendekati nol, sehingga hanya fitur-fitur penting yang dipertahankan. Dengan parameter alpha=100, seleksi dilakukan lebih ketat karena target regresi memiliki skala besar. Hasilnya, LASSO menentukan sejumlah fitur yang dianggap paling relevan untuk memprediksi harga mobil, dan fitur-fitur tersebut selanjutnya digunakan pada model regresi berikutnya.

14. **Visualisasi Seleksi Fitur LASSO – Prediksi Harga Mobil**

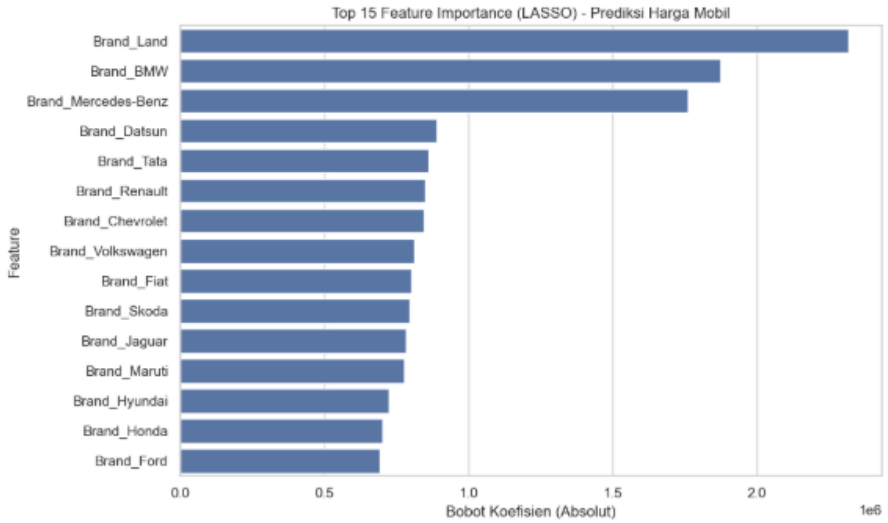
Model LASSO Regression dalam memprediksi harga jual mobil (*Income*). Nilai pentingnya diukur menggunakan **nilai absolut koefisien**, di mana semakin besar nilai tersebut, semakin besar pengaruh fitur terhadap prediksi.

Pada grafik, fitur-fitur yang muncul semuanya berupa **Brand mobil** hasil proses one-hot encoding, seperti *Brand_Land*, *Brand_BMW*, *Brand_Mercedes-Benz*, dan lainnya. Hal ini

menunjukkan bahwa **merek mobil merupakan faktor yang sangat dominan** dalam menentukan harga jual kendaraan. Semakin besar koefisiennya, semakin besar kontribusi merek tersebut terhadap kenaikan atau penurunan harga.

```
# --- Plot 8.1: Visualisasi Seleksi Fitur LASSO (Regresi) ---
plt.figure(figsize=(10, 6))
sns.barplot(x='Importance (Abs Coef)', y='Feature', data=l1_reg_coefs_df.head(15))
plt.title('Top 15 Feature Importance (LASSO) - Prediksi Harga Mobil')
plt.xlabel('Bobot Koefisien (Absolut)')
plt.tight_layout()
plt.show()
```

1.9s



Dari grafik terlihat bahwa:

- **Brand_Land** memiliki koefisien paling besar → artinya merek ini memiliki pengaruh paling kuat terhadap harga mobil.
- Diikuti oleh **Brand_BMW** dan **Brand_Mercedes-Benz**, yang juga dikenal sebagai brand premium dengan harga tinggi.
- Fitur-fitur lain seperti *Brand_Datsun*, *Brand_Tata*, dan *Brand_Renault* memiliki nilai koefisien lebih kecil, menunjukkan dampak harga yang lebih rendah dibanding merek premium.

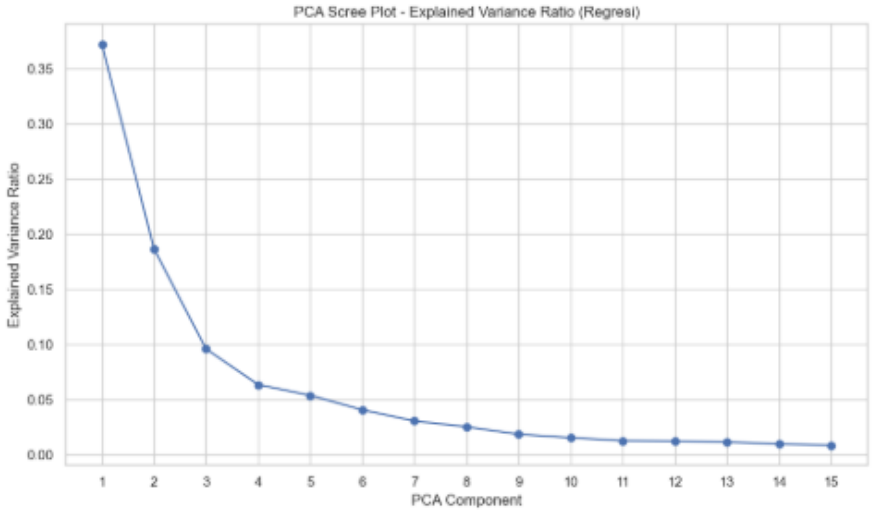
Visualisasi ini membantu menjelaskan fitur mana yang paling penting bagi model regresi setelah melalui proses seleksi LASSO. Dengan kata lain, grafik ini menunjukkan bahwa faktor utama yang mempengaruhi prediksi harga mobil adalah **merek kendaraan**, bukan fitur lain seperti tahun atau kilometer yang ditempuh.

15. Menjalankan PCA (Regresi)

Tahap ini bertujuan melakukan **PCA (Principal Component Analysis)** untuk mengurangi dimensi data, sambil tetap mempertahankan sebagian besar informasi penting. PCA membantu membuat model regresi lebih cepat, mengurangi noise, dan mengatasi multikolinearitas antar fitur.

```python
# B.2: Menjalankan PCA (Regresi)
print("\nB.2: Menjalankan PCA (Regresi)")
pca_reg = PCA(n_components=0.95, random_state=42)
pca_reg.fit(X_reg_train)
print(f"PCA memilih {pca_reg.n_components_} komponen untuk Regresi.")
# --- Plot B.2: Scree Plot PCA (Regresi) ---
plt.figure(figsize=(10, 6))
plt.plot(
    range(1, len(pca_reg.explained_variance_ratio_) + 1),
    pca_reg.explained_variance_ratio_,
    marker='o'
)
plt.title('PCA Scree Plot - Explained Variance Ratio (Regresi)')
plt.xlabel('PCA Component')
plt.ylabel('Explained Variance Ratio')
plt.xticks(range(1, len(pca_reg.explained_variance_ratio_) + 1))
plt.grid(True)
plt.tight_layout()
plt.show()
```

```
B.2: Menjalankan PCA (Regresi)
PCA memilih 15 komponen untuk Regresi.
```

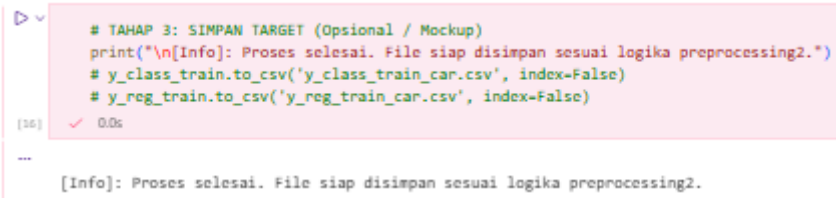


Proses PCA untuk mereduksi dimensi fitur pada data regresi. PCA diinisialisasi dengan n_components=0.95, artinya model akan memilih jumlah komponen utama yang cukup untuk mempertahankan 95% informasi dari dataset asli. Setelah PCA dilatih menggunakan data X_reg_train, hasilnya menunjukkan bahwa sebanyak 15 komponen utama sudah cukup menggantikan 33 fitur asli tanpa banyak kehilangan informasi. Selanjutnya dibuat Scree Plot yang menampilkan kontribusi variansi setiap komponen. Grafik tersebut memperlihatkan bahwa komponen pertama menjelaskan variansi paling besar, sekitar 32%, kemudian kontribusi variansi menurun pada komponen-komponen berikutnya. Setelah komponen ke-6, penurunan menjadi semakin kecil dan stabil. Visualisasi ini menegaskan bahwa sebagian besar informasi penting dapat diringkas oleh beberapa komponen PCA, sehingga data menjadi lebih ringkas, bersih, dan lebih efisien untuk dipakai pada model regresi.

16. Simpan Target (Opsional / Mockup)

Tahap ini merupakan bagian opsional yang digunakan untuk **menyimpan data target (label)** dari proses preprocessing. Penyimpanan dilakukan jika hasil preprocessing ingin digunakan kembali pada modeling atau pada sistem terpisah (misalnya pipeline machine

learning). Namun pada kode ini, proses penyimpanan masih dalam bentuk **komentar** (*mockup*) yang berarti tidak benar-benar dijalankan. Tujuannya adalah memberikan contoh atau template bagaimana proses penyimpanan seharusnya dilakukan.



Tahap ini langkah opsional untuk menyimpan target hasil preprocessing ke dalam file CSV, baik untuk tugas klasifikasi maupun regresi. Pada kode ini, perintah penyimpanan masih berupa komentar sehingga tidak dijalankan, tetapi tetap disertakan sebagai template jika sewaktu-waktu ingin menyimpan data target dari preprocessing. Pesan yang dicetak menandakan bahwa seluruh proses preprocessing telah selesai dan data siap dipakai atau disimpan sesuai kebutuhan.