Laporan Proyek Praktik: Pra-pemrosesan Data II (Feature Engineering)

1. Pendahuluan

Bagian laporan ini merupakan kelanjutan dari **Pra-pemrosesan I**, yang telah menghasilkan data latih dan data uji yang bersih (bebas dari *missing values* dan *outlier*), serta telah melalui *encoding* dan *scaling*.

Jika Pra-pemrosesan I berfokus pada **Feature Transformation** (mengubah data mentah agar *bisa dibaca* oleh model), maka Pra-pemrosesan II berfokus pada **Feature Optimization**. Tujuannya adalah untuk mengoptimalkan *dataset* yang sudah bersih tersebut agar lebih efisien dan efektif untuk *modelling*, dengan cara mengurangi jumlah fitur (dimensi) yang tidak relevan atau berlebihan (*redundant*).

1.1. Strategi Analisis Paralel

Untuk mencapai optimasi, dua skenario *feature engineering* utama dijalankan secara paralel. Alih-alih menjalankan satu *pipeline* berurutan (misalnya, Seleksi lalu PCA), kedua metode ini dijalankan secara terpisah pada data yang sama. Tujuannya adalah untuk menghasilkan dua *dataset* kandidat yang dioptimalkan secara berbeda, yang nantinya dapat dibandingkan kinerjanya saat tahap *modelling*.

- Skenario A: Seleksi Fitur (Feature Selection)

  Tujuan dari skenario ini adalah untuk mengurangi jumlah fitur dari 31 menjadi subset yang lebih kecil yang berisi fitur-fitur paling relevan dan penting. Fitur yang dianggap tidak penting atau redundant akan dibuang. Metode yang digunakan adalah LASSO (L1 Regularization).

- Skenario B: Reduksi Dimensi (PCA)

  Tujuan dari skenario ini adalah untuk merangkum informasi dari fitur-fitur numerik yang saling berkorelasi. Fitur tidak dibuang, melainkan ditransformasi menjadi satu set "Komponen Utama" (Principal Components) baru yang lebih efisien dan tidak saling berkorelasi.

1.2. Justifikasi Pipeline Ganda (Klasifikasi dan Regresi)

Agenda *supervised learning* mencakup dua tugas yang berbeda secara fundamental: **Klasifikasi** (memprediksi Response) dan **Regresi** (memprediksi Income).

Kedua tugas ini tidak dapat menggunakan satu *dataset* fitur yang sama.

- **Untuk Tugas Klasifikasi:** Targetnya adalah Response. Fitur Income adalah salah satu prediktor (fitur X) yang sangat penting untuk memprediksi Response.
- **Untuk Tugas Regresi:** Targetnya adalah Income. Oleh karena itu, Income tidak bisa menjadi fitur X (karena tidak bisa digunakan untuk memprediksi dirinya sendiri).

Karena susunan fitur X dan target y berbeda untuk kedua tugas, maka proses optimasi (Seleksi Fitur dan PCA) harus dijalankan dua kali secara terpisah. Laporan ini akan mendokumentasikan eksekusi untuk kedua *pipeline* tersebut (Bagian A untuk Klasifikasi dan Bagian B untuk Regresi).

2. Bagian A: Laporan Tugas Klasifikasi (Target: Response)

Pada bagian ini, *dataset* bersih dari Pra-pemrosesan I (data_train_preprocessed.csv) dioptimalkan untuk tugas **Klasifikasi**, yaitu memprediksi kolom target Response.

2.1. Inisialisasi Environment dan Pemuatan Data

2.1.1. Inisialisasi Environment

Langkah pertama adalah mengimpor semua library yang diperlukan untuk Pra-pemrosesan II, termasuk pandas dan numpy untuk manipulasi data, serta sklearn untuk feature selection (LASSO) dan reduksi dimensi (PCA).

```
# TAHAP 1: IMPOR DAN MUAT DATA

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
import warnings

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

2.1.2. Pemuatan Data Hasil Pra-pemrosesan I

Data latih dan uji yang sudah bersih (disimpan dari notebook Pra-pemrosesan I) dimuat ke dalam dataframe baru. Ini adalah titik awal kita untuk optimasi.

```
# 1.1. Muat data yang sudah diproses dari Tahap I
try:
    train_df = pd.read_csv('data_train_preprocessed.csv')
    test_df = pd.read_csv('data_test_preprocessed.csv')
except FileNotFoundError:
    print("Pastikan file 'data_train_preprocessed.csv' dan 'data_test_preprocessed.csv'
ada di folder yang sama.")

print(f"Data latih dimuat dengan bentuk: {train_df.shape}")
print(f"Data uji dimuat dengan bentuk: {test_df.shape}")
```

Output ini mengonfirmasi bahwa data latih (1792 baris) dan data uji (448 baris) telah berhasil dimuat. Keduanya memiliki 32 kolom (31 fitur + 1 kolom target).

```
Data latih dimuat dengan bentuk: (1792, 32)
Data uji dimuat dengan bentuk: (448, 32)
```

2.1.3. Pemisahan Data Klasifikasi

Data latih dan uji kemudian dipisahkan menjadi fitur (X) dan target (y). Untuk tugas klasifikasi ini, targetnya adalah Response. Fitur-fiturnya (X_class_train) adalah semua 31 kolom lainnya, termasuk Income, yang akan digunakan sebagai prediktor.

```
# BAGIAN A: TUGAS KLASIFIKASI (Target: Response)

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

# 1. Pisahkan X dan y untuk Klasifikasi
y_class_train = train_df['Response']
y_class_test = test_df['Response']
X_class_train = train_df.drop('Response', axis=1)
X_class_test = test_df.drop('Response', axis=1)
print(f"Bentuk X_class_train: {X_class_train.shape}")
```

Output ini mengonfirmasi bahwa data fitur latih (X_class_train) memiliki 1792 baris dan 31 kolom, siap untuk optimasi.

```
Memulai Bagian A: Persiapan Data Klasifikasi
Bentuk X_class_train: (1792, 31)
```

2.2. Skenario A: Seleksi Fitur (LASSO L1)

Skenario A berfokus pada **Seleksi Fitur**. Metode **LASSO (L1 Regularization)** digunakan, diimplementasikan melalui LogisticRegression(penalty='l1'). Metode *embedded* ini dipilih karena kemampuannya menangani multikolinearitas (fitur yang berlebihan) dengan secara otomatis "memaksa" koefisien fitur yang tidak penting menjadi nol.

```
# A.1: Skenario Seleksi Fitur (LASSO L1) untuk Klasifikasi

print("\nA.1: Menjalankan Seleksi Fitur LASSO (Klasifikasi)")

# Kita set C=0.05 (sedikit lebih kuat) untuk seleksi yang lebih ketat
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 dari model yang sudah di-fit di DALAM selector
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(list(selected_features_class))
```

```
# Buat dataframe baru
X_class_train_selected = X_class_train[selected_features_class]
X_class_test_selected = X_class_test[selected_features_class]

# Simpan hasil
X_class_train_selected.to_csv('X_class_train_selected.csv', index=False)
X_class_test_selected.to_csv('X_class_test_selected.csv', index=False)
print("Berhasil disimpan: 'X_class_train_selected.csv'")
```

Output teks menunjukkan bahwa dari 31 fitur, LASSO berhasil menguranginya menjadi **15 fitur** yang dianggap paling relevan. *Dataset* yang dioptimasi ini (X_class_train_selected.csv) kemudian disimpan.

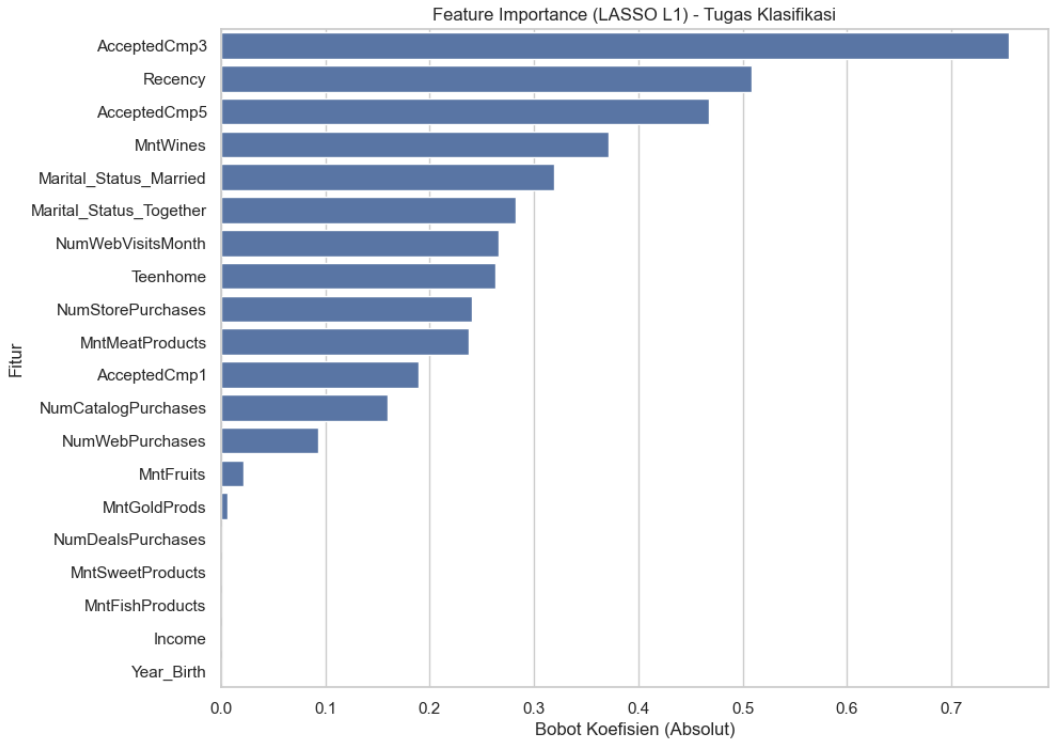
```
A.1: Menjalankan Seleksi Fitur LASSO (Klasifikasi)
LASSO memilih 15 fitur untuk Klasifikasi:
['Teenhome', 'Recency', 'MntWines', 'MntFruits', 'MntMeatProducts', 'MntGoldProds',
'NumWebPurchases', 'NumCatalogPurchases', 'NumStorePurchases', 'NumWebVisitsMonth',
'AcceptedCmp3', 'AcceptedCmp5', 'AcceptedCmp1', 'Marital_Status_Married',
'Marital_Status_Together']
Berhasil disimpan: 'X_class_train_selected.csv'
```

2.3. Visualisasi Hasil Seleksi Fitur (LASSO)

Untuk menganalisis fitur apa yang dipilih oleh LASSO, dibuatlah visualisasi "Feature Importance". Plot ini mengurutkan fitur berdasarkan nilai absolut dari koefisien (bobot) yang diberikan oleh model LASSO.

```
# --- [Plot A.1: Visualisasi Seleksi Fitur LASSO (Klasifikasi)] ---

plt.figure(figsize=(10, 8))
# Kita plot 20 fitur teratas untuk melihat mana yang dipilih dan mana yang tidak
sns.barplot(
    x='Importance (Abs Coef)',
    y='Feature',
    data=l1_coefs_df.head(20)
)
plt.title('Feature Importance (LASSO L1) - Tugas Klasifikasi')
plt.xlabel('Bobot Koefisien (Absolut)')
plt.ylabel('Fitur')
plt.show()
```

Plot ini memberikan wawasan penting. Fitur-fitur dengan bobot tertinggi (paling penting) adalah AcceptedCmp3, Recency, AcceptedCmp5, dan MntWines. Ini menunjukkan bahwa perilaku respons masa lalu dan keterlibatan terbaru adalah prediktor terbaik. Sebaliknya, fitur demografis seperti Income dan Year_Birth memiliki bobot yang sangat rendah, menunjukkan relevansi yang kecil untuk tugas klasifikasi ini.



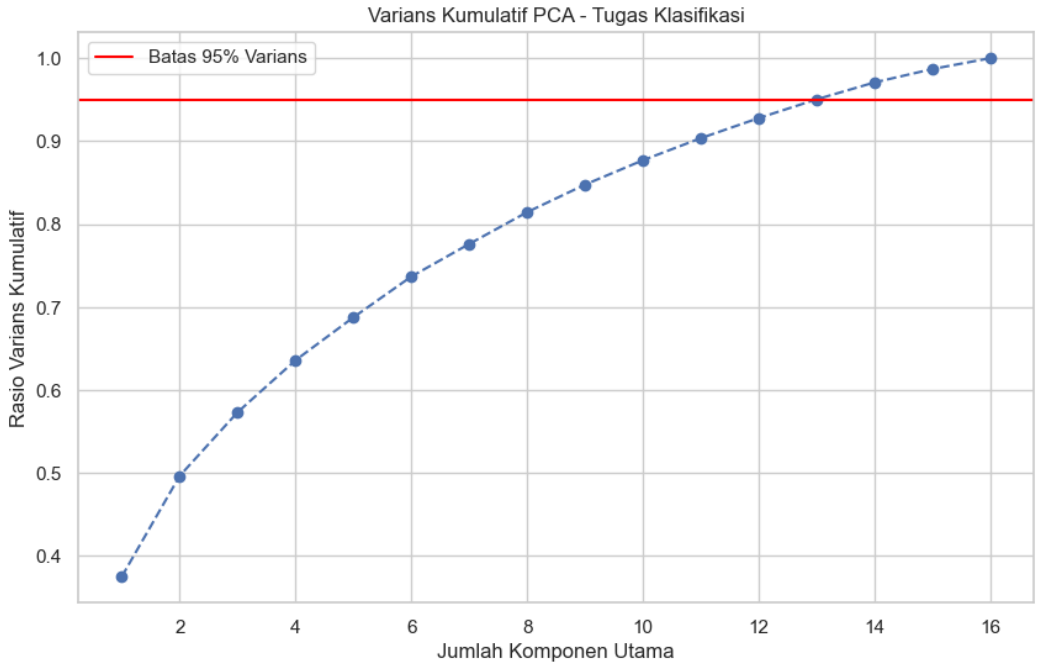
2.4. Skenario B: Reduksi Dimensi (PCA) - Analisis Komponen

Skenario B berfokus pada **Reduksi Dimensi**. Tujuannya adalah untuk **merangkum** 16 fitur numerik (termasuk Income) menjadi lebih sedikit "Komponen Utama" (PC) baru.

Untuk menentukan jumlah komponen yang optimal, PCA di-*fit* pada data latih numerik dan "Scree Plot" (Varians Kumulatif) dibuat.

```
# --- [Plot A.2: Visualisasi Varians Kumulatif PCA (Klasifikasi)] ---

# Fit PCA penuh untuk memvisualisasikan varians
pca_full_class = PCA(random_state=42).fit(X_train_num)
cumulative_variance = np.cumsum(pca_full_class.explained_variance_ratio_)

plt.figure(figsize=(10, 6))
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 - Tugas Klasifikasi')
plt.xlabel('Jumlah Komponen Utama')
plt.ylabel('Rasio Varians Kumulatif')
plt.legend(loc='best')
plt.grid(True)
plt.show()
```

Plot varians kumulatif ini adalah justifikasi visual untuk jumlah komponen yang kita pilih. Garis biru menunjukkan bahwa untuk menangkap **95% dari total informasi** (varians) dari 16 fitur numerik asli, kita membutuhkan sekitar **13 Komponen Utama**.



2.5. Penerapan PCA dan Penyimpanan Hasil

Setelah jumlah komponen (13) ditentukan, transformasi PCA diterapkan pada data numerik (latih dan uji). Data biner (15 fitur) yang tidak di-PCA kemudian digabungkan kembali dengan 13 Komponen Utama baru ini, dan hasilnya disimpan.

```
# A.2 (Lanjutan): Transformasi dan Penyimpanan PCA Klasifikasi

# 3. Transformasi data numerik
X_train_pca_transformed = pca_class.transform(X_train_num)
X_test_pca_transformed = pca_class.transform(X_test_num)

# 4. Buat dataframe PCA dan gabungkan kembali
pc_cols = [f'PC{i+1}' for i in range(pca_class.n_components_)]
X_train_pca_df = pd.DataFrame(X_train_pca_transformed, columns=pc_cols)
X_test_pca_df = pd.DataFrame(X_test_pca_transformed, columns=pc_cols)

X_class_train_pca = pd.concat([X_train_bin.reset_index(drop=True),
X_train_pca_df.reset_index(drop=True)], axis=1)
X_class_test_pca = pd.concat([X_test_bin.reset_index(drop=True),
X_test_pca_df.reset_index(drop=True)], axis=1)

# 5. Simpan hasil
X_class_train_pca.to_csv('X_class_train_pca.csv', index=False)
X_class_test_pca.to_csv('X_class_test_pca.csv', index=False)
print("Berhasil disimpan: 'X_class_train_pca.csv'")
```

Output ini mengonfirmasi bahwa *dataset* kandidat kedua (X_class_train_pca.csv), yang berisi 15 fitur biner + 13 Komponen Utama, telah berhasil disimpan.

```
Berhasil disimpan: 'X_class_train_pca.csv'
```

3. Bagian B: Laporan Tugas Regresi (Target: Income)

Bagian ini mendokumentasikan optimasi *dataset* untuk tugas **Regresi**, yaitu memprediksi kolom target Income.

3.1. Pemisahan Data Regresi

Prosesnya serupa dengan tugas klasifikasi, namun dengan satu perbedaan krusial:

1. Target (y_reg_train) sekarang adalah Income.
2. Fitur (X_reg_train) adalah semua kolom lain, **termasuk Response** (karena status respons kampanye bisa menjadi prediktor pendapatan) tetapi **tidak termasuk Income** (karena itu adalah targetnya).

```
# BAGIAN B: TUGAS REGRESI (Target: Income)

print("\nMemulai Bagian B: Persiapan Data Regresi")

y_reg_train = train_df['Income']
y_reg_test = test_df['Income']
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}")
```

Output ini mengonfirmasi bahwa X_reg_train memiliki 31 fitur, sama seperti *dataset* klasifikasi, tetapi komposisinya berbeda (sekarang berisi Response dan tidak berisi Income).

```
Memulai Bagian B: Persiapan Data Regresi
Bentuk X_reg_train: (1792, 31)
```

3.2. Skenario A: Seleksi Fitur (LASSO L1)

Untuk tugas regresi, model Lasso (versi regresi dari LASSO) digunakan untuk seleksi fitur. Model ini juga menerapkan *penalty L1* untuk "menolkan" koefisien fitur yang tidak penting.

```
# B.1: Skenario Seleksi Fitur (LASSO) untuk Regresi

print("\nB.1: Menjalankan Seleksi Fitur LASSO (Regresi)")

l1_reg_model = Lasso(alpha=0.1, random_state=42) # alpha=0.1 adalah nilai default
l1_reg_selector = SelectFromModel(l1_reg_model, max_features=15)
l1_reg_selector.fit(X_reg_train, y_reg_train)

# Ambil koefisien dari model di dalam selector
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)

# Ambil fitur yang lolos seleksi
selected_features_reg = X_reg_train.columns[l1_reg_selector.get_support()]
```

```
print(f"LASSO memilih {len(selected_features_reg)} fitur untuk Regresi:")
print(list(selected_features_reg))

# Buat dataframe baru
X_reg_train_selected = X_reg_train[selected_features_reg]
X_reg_test_selected = X_reg_test[selected_features_reg]

# Simpan hasil
X_reg_train_selected.to_csv('X_reg_train_selected.csv', index=False)
X_reg_test_selected.to_csv('X_reg_test_selected.csv', index=False)
print("Berhasil disimpan: 'X_reg_train_selected.csv'")
```

Output teks menunjukkan bahwa untuk memprediksi Income, LASSO bersikap jauh lebih "ketat" dan hanya memilih **7 fitur** dari 31 fitur yang tersedia.

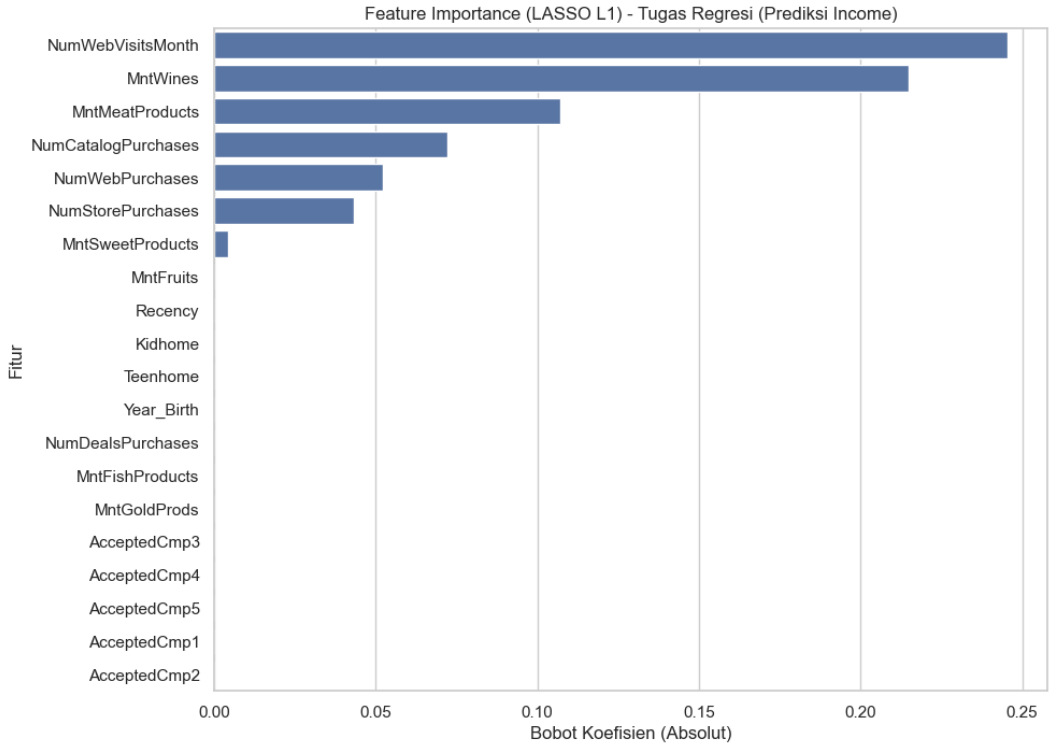
```
B.1: Menjalankan Seleksi Fitur LASSO (Regresi)
LASSO memilih 7 fitur untuk Regresi:
['MntWines', 'MntMeatProducts', 'MntSweetProducts', 'NumWebPurchases',
'NumCatalogPurchases', 'NumStorePurchases', 'NumWebVisitsMonth']
Berhasil disimpan: 'X_reg_train_selected.csv'
```

3.3. Visualisasi Hasil Seleksi Fitur (LASSO)

Plot "Feature Importance" dibuat untuk menganalisis 7 fitur yang dipilih LASSO untuk tugas regresi.

```
# --- [Plot B.1: Visualisasi Seleksi Fitur LASSO (Regresi)] ---

plt.figure(figsize=(10, 8))
# Plot 20 fitur teratas
sns.barplot(
    x='Importance (Abs Coef)',
    y='Feature',
    data=l1_reg_coefs_df.head(20)
)
plt.title('Feature Importance (LASSO L1) - Tugas Regresi (Prediksi Income)')
plt.xlabel('Bobot Koefisien (Absolut)')
plt.ylabel('Fitur')
plt.show()
```

Analisis plot ini sangat penting. Fitur-fitur yang dipilih (NumWebVisitsMonth, MntWines, MntMeatProducts, dll.) sangat berbeda dari fitur yang dipilih untuk tugas klasifikasi. Ini sangat logis: **kebiasaan belanja** (berapa banyak yang dihabiskan untuk wine/daging) adalah prediktor terbaik untuk **pendapatan (Income)**.



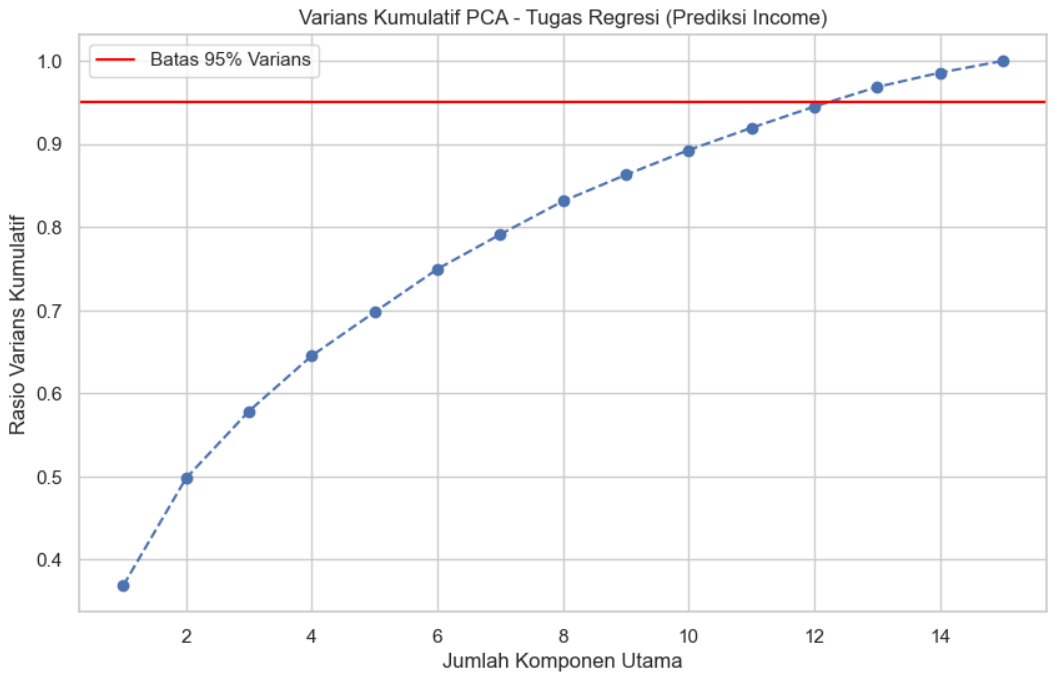
3.4. Skenario B: Reduksi Dimensi (PCA) - Analisis Komponen

Skenario B dijalankan untuk tugas regresi, merangkum 15 fitur numerik (tidak termasuk Income).

```
# B.2: Skenario Reduksi Dimensi (PCA) untuk Regresi

print("\nB.2: Menjalankan PCA (Regresi)")

# 1. Tentukan kolom numerik (TANPA Income) dan biner (TERMASUK Response)
numerical_cols_reg = [col for col in X_reg_train.columns if X_reg_train[col].nunique() >
2]
binary_cols_reg = [col for col in X_reg_train.columns if col not in numerical_cols_reg]

# 2. Pisahkan data numerik dan biner
X_train_num_reg = X_reg_train[numerical_cols_reg]
X_train_bin_reg = X_reg_train[binary_cols_reg]
X_test_num_reg = X_reg_test[numerical_cols_reg]
X_test_bin_reg = X_reg_test[binary_cols_reg]

# 3. Fit PCA untuk menangkap 95% varians
pca_reg = PCA(n_components=0.95, random_state=42)
pca_reg.fit(X_train_num_reg)
print(f"PCA memilih {pca_reg.n_components_} komponen untuk Regresi.")
```

```
B.2: Menjalankan PCA (Regresi)
PCA memilih 13 komponen untuk Regresi.
```

```
# --- [Plot B.2: Visualisasi Varians Kumulatif PCA (Regresi)] ---

# Fit PCA penuh untuk memvisualisasikan varians
pca_full_reg = PCA(random_state=42).fit(X_train_num_reg)
cumulative_variance_reg = np.cumsum(pca_full_reg.explained_variance_ratio_)

plt.figure(figsize=(10, 6))
plt.plot(range(1, len(cumulative_variance_reg) + 1), cumulative_variance_reg, marker='o',
linestyle='--')
plt.axhline(y=0.95, color='red', linestyle='-', label='Batas 95% Varians')
plt.title('Varians Kumulatif PCA - Tugas Regresi (Prediksi Income)')
plt.xlabel('Jumlah Komponen Utama')
plt.ylabel('Rasio Varians Kumulatif')
plt.legend(loc='best')
plt.grid(True)
plt.show()
```



Output teks dan plot "Scree Plot" keduanya mengonfirmasi hasil yang serupa dengan tugas klasifikasi. Untuk menangkap 95% varians dari 15 fitur numerik asli, **13 Komponen Utama** baru dipilih oleh PCA.

3.5. Penerapan PCA dan Penyimpanan Hasil

Terakhir, transformasi PCA diterapkan dan hasilnya digabungkan kembali dengan 16 fitur biner/encoded (termasuk Response) sebelum disimpan.

```
# B.2 (Lanjutan): Transformasi dan Penyimpanan PCA Regresi

# 4. Transformasi data numerik
X_train_pca_transformed_reg = pca_reg.transform(X_train_num_reg)
X_test_pca_transformed_reg = pca_reg.transform(X_test_num_reg)

# 5. Buat dataframe PCA dan gabungkan kembali
pc_cols_reg = [f'PC{i+1}' for i in range(pca_reg.n_components_)]
X_train_pca_df_reg = pd.DataFrame(X_train_pca_transformed_reg, columns=pc_cols_reg)
X_test_pca_df_reg = pd.DataFrame(X_test_pca_transformed_reg, columns=pc_cols_reg)

X_reg_train_pca = pd.concat([X_train_bin_reg.reset_index(drop=True),
X_train_pca_df_reg.reset_index(drop=True)], axis=1)
X_reg_test_pca = pd.concat([X_test_bin_reg.reset_index(drop=True),
X_test_pca_df_reg.reset_index(drop=True)], axis=1)

# 6. Simpan hasil
X_reg_train_pca.to_csv('X_reg_train_pca.csv', index=False)
X_reg_test_pca.to_csv('X_reg_test_pca.csv', index=False)
print("Berhasil disimpan: 'X_reg_train_pca.csv'")
```

Output ini mengonfirmasi bahwa *dataset* (X_reg_train_pca.csv), yang berisi16 fitur biner/encoded (termasuk Response), telah berhasil disimpan.

```
Berhasil disimpan: 'X_reg_train_pca.csv'
```

3.6. Penyimpanan File Target Pra-Pemrosesan II

Sebagai langkah terakhir dari Pra-pemrosesan II, keempat *dataset* target (latih dan uji, untuk klasifikasi dan regresi) disimpan ke dalam file CSV terpisah. Ini memastikan bahwa *dataset* fitur (X) dan target (y) dapat dimuat secara independen di agenda *modelling* selanjutnya.

```
# TAHAP 3: SIMPAN TARGET (y)

print("\nMenyimpan semua file target")

y_class_train.to_csv('y_class_train.csv', index=False, header=True)
y_class_test.to_csv('y_class_test.csv', index=False, header=True)
y_reg_train.to_csv('y_reg_train.csv', index=False, header=True)
y_reg_test.to_csv('y_reg_test.csv', index=False, header=True)

print("\nPra-pemrosesan II Selesai.")
print("Total 12 file (8 fitur, 4 target) telah disimpan.")
```

Output ini mengonfirmasi bahwa seluruh alur kerja telah selesai. Total 12 file (8 file fitur dari Skenario A dan B, ditambah 4 file target ini) telah berhasil dibuat dan disimpan. *Dataset* yang telah dioptimalkan ini sekarang siap sepenuhnya untuk digunakan dalam agenda **Supervised Learning - Klasifikasi** dan **Supervised Learning - Regresi**.

```
Menyimpan semua file target..

Pra-pemrosesan II Selesai.
Total 12 file (8 fitur, 4 target) telah disimpan.
```

4. Kesimpulan dan Panduan Penggunaan File

Proses ini bertujuan mengoptimalkan 31 fitur yang telah dibersihkan (dari Pra-pemrosesan I) menjadi dataset yang lebih efisien untuk *modelling*.

Untuk melakukan ini, dua "eksperimen" atau strategi optimasi dijalankan secara paralel:

1. **Skenario A (Seleksi Fitur)**: Menggunakan LASSO (L1) untuk memilih fitur-fitur paling penting dan membuang fitur yang tidak relevan (noise).
2. **Skenario B (Reduksi Dimensi)**: Menggunakan PCA untuk merangkum (mentransformasi) fitur-fitur numerik yang berkorelasi menjadi "Komponen Utama" yang lebih sedikit.

Strategi menjalankan Seleksi Fitur (Skenario A) dan Reduksi Dimensi (Skenario B) secara **paralel**—alih-alih berurutan—dipilih karena keduanya adalah filosofi optimasi yang berbeda secara fundamental. **Seleksi Fitur (LASSO)** bertujuan untuk **memilih** subset fitur asli yang paling penting dan **membuang** fitur yang *redundant* atau tidak relevan. Sebaliknya, **Reduksi Dimensi (PCA)** bertujuan untuk **merangkum** informasi dari fitur-fitur numerik yang saling berkorelasi (redundant) dan **mentransformasikannya** menjadi set "Komponen Utama" baru yang lebih efisien. Menjalankan kedua skenario ini secara paralel memungkinkan perbandingan langsung di tahap *modelling* untuk menentukan pendekatan optimasi mana (memilih fitur asli vs. merangkum fitur) yang menghasilkan model paling akurat untuk dataset ini.

Karena ada dua tugas *supervised learning* yang berbeda (Klasifikasi dan Regresi), yang memiliki target dan susunan fitur yang berbeda, kedua skenario ini harus dijalankan dua kali (satu kali untuk setiap tugas)..

Untuk **Agenda Klasifikasi** *(target y_class_train.csv dan y_class_test.csv*), dua model akan dilatih dan dievaluasi. **Eksperimen A** akan menggunakan fitur hasil Seleksi LASSO (*X_class_train_selected.csv dan X_class_test_selected.csv*). **Eksperimen B** akan menggunakan fitur hasil PCA (*X_class_train_pca.csv dan X_class_test_pca.csv*). Penting untuk dicatat bahwa *resampling* (seperti SMOTE) harus diterapkan pada kedua set data latih (*X_class_..._train.csv*) untuk menangani ketidakseimbangan kelas Response yang signifikan sebelum pelatihan.

Secara paralel, untuk **Agenda Regresi** (*target y_reg_train.csv dan y_reg_test.csv*), dua model Regresi Linier akan dilatih. **Eksperimen A** akan menggunakan 7 fitur yang dipilih oleh LASSO (*X_reg_train_selected.csv*). **Eksperimen B** akan menggunakan fitur gabungan hasil PCA (*X_reg_train_pca.csv*). Kinerja kedua model regresi ini kemudian akan dievaluasi pada set data uji masing-masing (*X_reg_test_...csv*) untuk menentukan strategi optimasi mana yang menghasilkan prediksi Income paling akurat.