NAMA : RAFLI NAUFAL

NIM : 105841109823

KELAS: 5 AI-A

MATKUL: APPLIED MACHINE LEARNING

LAPORAN PREPROCESSING FEATURE ENGINEERING

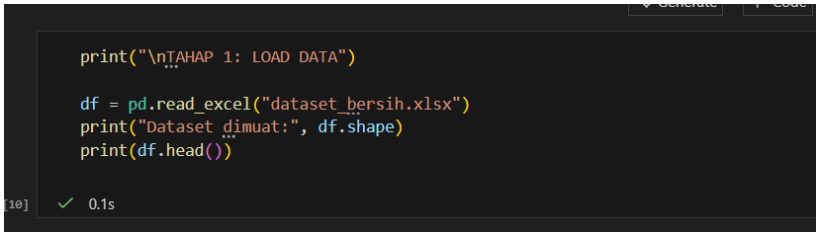
Dalam membangun model prediksi untuk risiko kredit (*credit risk*), kualitas data memegang peranan yang jauh lebih penting daripada kompleksitas algoritmanya. Data mentah seringkali mengandung banyak variabel yang tidak relevan (noise) atau memiliki dimensi yang terlalu besar yang justru bisa membingungkan model.

Oleh karena itu, laporan ini disusun untuk menjelaskan secara rinci tahapan kode program yang telah saya buat. Fokus utama dari kode ini adalah melakukan pra-pemrosesan data (*data preprocessing*) yang sistematis—mulai dari pemilahan fitur penting menggunakan metode LASSO hingga penyederhanaan struktur data menggunakan PCA. Tujuannya adalah menghasilkan dataset yang bersih, efisien, dan siap untuk diolah lebih lanjut.

1. Pemuatan dan Pengecekan Awal Data (*Data Loading*)

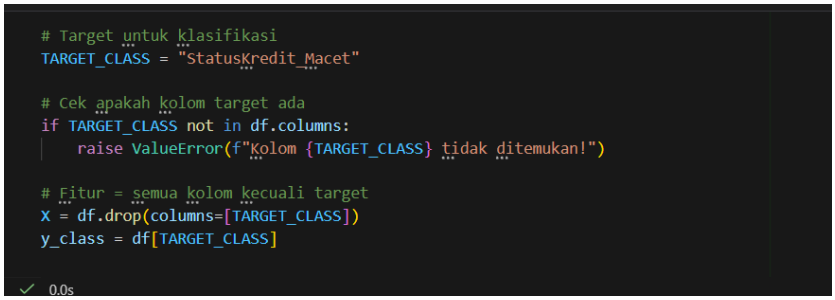
Pada tahap ini, proses dimulai dengan memanggil file sumber data yang bernama dataset_bersih.xlsx menggunakan fungsi read_excel dari pustaka Pandas. Data dari file Excel tersebut kemudian dimuat dan disimpan ke dalam variabel variabel df (DataFrame) agar dapat diproses lebih lanjut oleh sistem. Setelah proses pemuatan selesai, program akan menampilkan informasi mengenai dimensi dataset (jumlah total baris dan kolom) menggunakan perintah .shape. Selain itu, program juga menampilkan 5 baris teratas dari data tersebut menggunakan perintah .head() sebagai langkah verifikasi awal untuk memastikan bahwa data telah berhasil dibaca

dengan format yang benar sebelum masuk ke tahap analisis selanjutnya.



2. Penentuan Target dan Pemisahan Fitur (*Feature Splitting*)

Pada tahap ini, fokus utama adalah memisahkan data menjadi dua komponen yang diperlukan untuk pemodelan. Pertama, variabel target didefinisikan secara spesifik sebagai kolom 'StatusKredit_Macet'. Sistem kemudian melakukan validasi otomatis untuk memeriksa ketersediaan kolom tersebut di dalam dataset; jika kolom tidak ditemukan, program akan berhenti dan menampilkan pesan kesalahan untuk mencegah kegagalan proses selanjutnya. Setelah dipastikan ada, data kemudian dibagi menjadi variabel X (kumpulan fitur) yang berisi seluruh kolom data kecuali kolom target, dan variabel y_class yang hanya berisi data target tersebut. Pemisahan ini dilakukan agar model nantinya dapat mempelajari pola dari fitur (X) untuk memprediksi status kredit (y).



3. Pembagian Data Latih dan Data Uji (*Train-Test Split*)

Setelah fitur dan target ditentukan, tahap selanjutnya adalah membagi dataset menjadi dua bagian terpisah, yaitu data latih (*training set*) dan data uji (*testing set*). Proses ini dilakukan menggunakan fungsi train_test_split dari pustaka Scikit-Learn. Dalam kode ini, proporsi pembagian diatur sebesar 80:20, di mana 20% data (test_size=0.2) dialokasikan khusus sebagai data uji untuk mengevaluasi performa

model nantinya, sedangkan 80% sisanya digunakan untuk melatih model. Penerapan parameter random_state=42 bertujuan untuk mengunci proses pengacakan agar hasil pembagian data tetap konsisten (tidak berubah-ubah) setiap kali program dijalankan. Terakhir, program menampilkan dimensi dari keempat variabel hasil pemisahan (X_train, X_test, y_train, y_test) untuk memverifikasi bahwa distribusi jumlah data sudah sesuai."

```
from sklearn.model_selection import train_test_split

print("\nTAHAP 2: SPLIT DATA")

# Split data hanya untuk klasifikasi
X_train_c, X_test_c, y_train_c, y_test_c = train_test_split(
    X, y_class, test_size=0.2, random_state=42
)

print("Split klasifikasi:")
print("X_train:", X_train_c.shape)
print("X_test :", X_test_c.shape)
print("y_train:", y_train_c.shape)
print("y_test :", y_test_c.shape)
```

0.0s

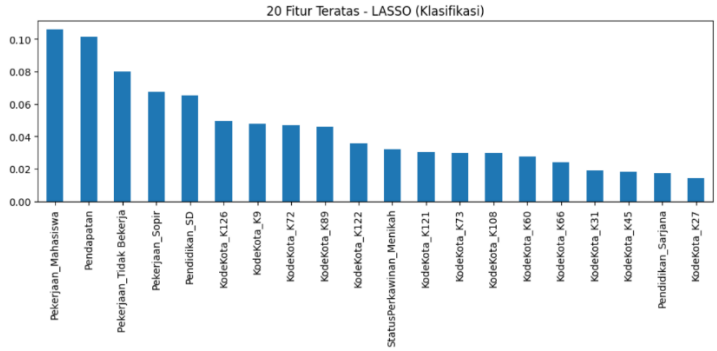
4. Seleksi Fitur Penting Menggunakan Algoritma LASSO

Pada tahap ini, dilakukan proses seleksi fitur untuk mengetahui variabel mana yang paling berpengaruh terhadap target klasifikasi. Sebuah fungsi khusus bernama run_lasso didefinisikan untuk menjalankan proses ini. Langkah pertama dalam fungsi tersebut adalah melakukan standarisasi data menggunakan StandardScaler, tujuannya menyamakan rentang nilai antar kolom agar perhitungan algoritma adil dan akurat. Selanjutnya, model LASSO (*Least Absolute Shrinkage and Selection Operator*) dilatih menggunakan data yang sudah distandarisasi tersebut. Dari hasil pelatihan, program mengambil nilai koefisien (bobot) setiap fitur; fitur dengan nilai koefisien tertinggi dianggap sebagai fitur yang paling penting. Hasil akhirnya kemudian divisualisasikan menggunakan grafik batang (*bar chart*) yang menampilkan 20 fitur dengan pengaruh terbesar, memudahkan kita untuk melihat faktor utama penyebab kredit macet.

```python
def run_lasso(X_train, y_train, alpha=0.01):
    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(X_train)

    model = Lasso(alpha=alpha)
    model.fit(X_scaled, y_train)

    coef = pd.Series(model.coef_, index=X_train.columns)
    coef_sorted = coef.sort_values(ascending=False)

    return coef_sorted


print("\nTAHAP 3: LASSO (Klasifikasi)")
lasso_class = run_lasso(X_train_c, y_train_c)
print(lasso_class.head())

plt.figure(figsize=(10,5))
lasso_class.head(20).plot(kind="bar")
plt.title("20 Fitur Teratas - LASSO (Klasifikasi)")
plt.tight_layout()
plt.show()
```

0.2s

hasil eksekusi kode di atas, program menampilkan grafik batang yang mengurutkan 20 fitur paling berpengaruh. Dari visualisasi tersebut, terlihat jelas bahwa variabel **'Pekerjaan_Mahasiswa'** dan **'Pendapatan'** memiliki bobot tertinggi dibandingkan fitur lainnya. Hal ini menunjukkan bahwa menurut model LASSO, kedua faktor tersebut adalah penentu utama dalam memprediksi apakah kredit seseorang akan macet atau tidak."



5. Reduksi Dimensi Data dengan PCA (*Principal Component Analysis*)

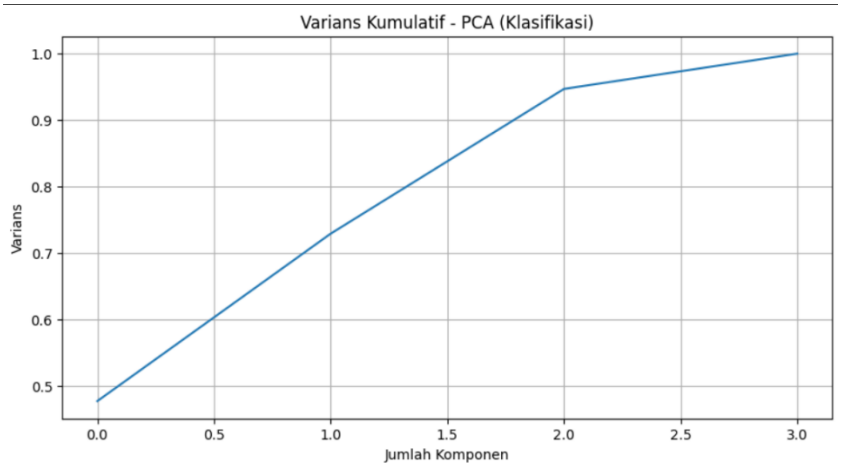
Tahap ini bertujuan untuk menyederhanakan kompleksitas dataset (reduksi dimensi) namun tetap mempertahankan informasi penting di dalamnya menggunakan teknik *Principal Component Analysis* (PCA). Di dalam fungsi run_pca, langkah awal yang dilakukan adalah memfilter data agar hanya kolom bernilai numerik yang diproses, karena PCA bekerja berdasarkan perhitungan matematis. Data numerik tersebut kemudian distandarisasi menggunakan StandardScaler agar perbedaan skala antar fitur tidak membiaskan hasil analisis. Program kemudian membuat grafik 'Varians Kumulatif' untuk memvisualisasikan

seberapa banyak informasi yang dapat dijelaskan oleh sejumlah komponen tertentu. Pada langkah akhir, transformasi PCA diterapkan dengan parameter n_components=0.95, yang artinya sistem secara otomatis memilih jumlah komponen minimal yang mampu mempertahankan 95% varians (informasi) dari data asli, membuang sisanya yang dianggap kurang signifikan

```python
def run_pca(X, title):
    numeric_cols = X.select_dtypes(include=[np.number]).columns
    X_num = X[numeric_cols]

    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(X_num)

    pca_full = PCA()
    pca_full.fit(X_scaled)

    plt.figure(figsize=(10,5))
    plt.plot(np.cumsum(pca_full.explained_variance_ratio_))
    plt.title(f"Varians Kumulatif - PCA ({title})")
    plt.xlabel("Jumlah Komponen")
    plt.ylabel("Varians")
    plt.grid()
    plt.show()

    # PCA final dengan 95% varians
    pca_final = PCA(n_components=0.95)
    X_pca = pca_final.fit_transform(X_scaled)

    print(f"PCA {title} menghasilkan {pca_final.n_components_} komponen")

    return X_pca, pca_final


print("\nTAHAP 4: PCA (Klasifikasi)")
X_pca_class, pca_model_class = run_pca(X_train_c, "Klasifikasi")
```

0.2s

Sebagai pembuktian bahwa data hasil penyederhanaan masih berkualitas, kita bisa melihat grafik varians kumulatif di bawah ini. Grafik ini menggambarkan kenaikan jumlah informasi yang didapat seiring bertambahnya jumlah komponen:



6. Penyimpanan Hasil Pemrosesan Data (*Data Export*)

Sebagai langkah penutup, seluruh hasil analisis penting yang telah diproses disimpan ke dalam file eksternal berformat CSV (*Comma Separated Values*) agar

dapat digunakan kembali tanpa harus menjalankan ulang kode dari awal. Pertama, hasil seleksi fitur dari algoritma LASSO disimpan ke dalam file bernama lasso_classification.csv, yang berisi daftar fitur beserta bobot kepentingannya. Kedua, data hasil transformasi dimensi (PCA) yang sebelumnya berbentuk *array* dikonversi terlebih dahulu menjadi format tabel (DataFrame), lalu disimpan dengan nama pca_classification.csv. Pada proses penyimpanan ini, digunakan parameter index=False untuk memastikan bahwa nomor indeks baris tidak ikut tertulis ke dalam file, sehingga struktur data tetap rapi dan bersih. Terakhir, program menampilkan pesan konfirmasi untuk memvalidasi bahwa seluruh rangkaian proses (*pipeline*) telah berhasil dijalankan tanpa adanya *error*.

7. Kesimpulan

Berdasarkan seluruh tahapan kode yang telah dijalankan, dapat disimpulkan bahwa proses pra-pemrosesan (*preprocessing*) data memegang peranan kunci dalam menyiapkan data kredit yang berkualitas. Proses ini tidak hanya sekadar memuat data, tetapi melibatkan serangkaian teknik filtrasi cerdas untuk memastikan model nantinya bekerja optimal.Secara spesifik, penerapan algoritma LASSO terbukti sangat membantu dalam menyaring puluhan fitur yang ada, membuang informasi sampah, dan hanya mempertahankan variabel yang benar-benar mempengaruhi status kredit (seperti pendapatan dan pekerjaan). Selanjutnya, penggunaan metode PCA berhasil menyederhanakan data yang kompleks menjadi format yang jauh lebih ringkas tanpa menghilangkan inti informasinya.Hasil akhirnya, kita mendapatkan kumpulan data yang bersih, relevan, dan efisien yang telah tersimpan rapi dalam format CSV. Dengan data yang sudah 'dimatangkan' ini, proses pembuatan model prediksi kedepannya akan menjadi lebih cepat secara komputasi dan berpotensi menghasilkan akurasi yang jauh lebih tinggi dibandingkan jika kita menggunakan data mentah.