LAPORAN PRAKTIK: TUGAS 2 Pre-processing - Feature Engineering

1. Pendahuluan

Tahap ini berfokus pada proses *Feature Engineering* yang meliputi dua metode utama: Feature Selection menggunakan Lasso (L1) dan Dimensionality Reduction menggunakan PCA. Tujuan tahap ini adalah mengoptimalkan fitur agar model di tahap berikutnya bekerja lebih efektif. Data yang digunakan terdiri dari dua bagian: data train dan data test yang telah melalui tahap *cleaning* sebelumnya.Konteks: Data sudah dibersihkan dan di-scale pada Tugas 1. Di tahap ini, kita menjalankan dua skenario untuk dua tugas modelling (klasifikasi dan regresi):

a. Seleksi fitur menggunakan Lasso (L1 regularization) → memilih subset variabel paling penting.

b. Reduksi dimensi menggunakan PCA → mereduksi jumlah fitur sambil mempertahankan sebanyak mungkin varians.

Target:

- Untuk klasifikasi: y = status pinjaman (gagal bayar vs lunas)
- Untuk regresi: `y = jumlah pinjaman` atau `suku bunga`

2. Metodologi & Tahapan Kode

2.1 Import Library dan Setup

Notebook memulai dengan melakukan import library yang diperlukan, termasuk pandas, numpy, sklearn, dan library visualisasi. Selain itu dilakukan juga suppression warning untuk menjaga output tetap bersih.

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from sklearn.linear_model import LogisticRegression, Lasso
from sklearn.feature_selection import SelectFromModel
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

import warnings, os
warnings.filterwarnings("ignore")
```

Pada tahap ini, seluruh library yang dibutuhkan untuk proses feature engineering dimuat. StandardScaler digunakan untuk normalisasi data sebelum PCA, PCA digunakan untuk reduksi dimensi, dan LassoCV digunakan untuk feature selection. warnings.filterwarnings("ignore") berfungsi untuk menyembunyikan warning agar output lebih rapi.

2.2 Load Dataset Train & Test

Data training dan testing dibaca dari file CSV hasil tahap pra-pemrosesan sebelumnya.

```python
# ============================================================
# 1. LOAD DATA
# ============================================================

train_df = pd.read_csv("data_train_preprocessed.csv")
test_df  = pd.read_csv("data_test_preprocessed.csv")

print("Train shape :", train_df.shape)
print("Test shape  :", test_df.shape)
train_df.head()
```

Kode ini membaca data hasil pra-pemrosesan dan menampilkan bentuk (*shape*) dataset. Fungsi head() digunakan untuk menampilkan beberapa baris awal dataset sebagai pengecekan awal. Ini memastikan bahwa data telah berhasil dimuat dan memiliki struktur yang benar.

2.3 Lasso Feature Selection – Classification

Tahap ini melakukan seleksi fitur menggunakan Lasso untuk tugas klasifikasi. Fitur (X_class) dipisahkan dari target (class). Kemudian model LassoCV dijalankan menggunakan cross-validation untuk menentukan regularisasi optimal. Hasil koefisien Lasso disimpan ke file CSV untuk kebutuhan analisis lebih lanjut.

```
# ============================================================
# A.1 — LASSO FEATURE SELECTION (LOGISTIC L1)
# ============================================================

max_feats_class = min(15, X_class_train.shape[1])

l1_model_class = LogisticRegression(
    penalty="l1",
    C=0.05,
    solver="liblinear",
    random_state=42,
    max_iter=1000
)

selector_class = SelectFromModel(l1_model_class, max_features=max_feats_class)
selector_class.fit(X_class_train, y_class_train)

selected_features_class = X_class_train.columns[selector_class.get_support()]
print("Selected features (classification):")
print(selected_features_class)

# Simpan hasil LASSO
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.csv", index=False)
X_class_test_selected.to_csv("X_class_test_selected.csv", index=False)

# Importance
lasso_importance_class = pd.DataFrame({
    "Feature": X_class_train.columns,
    "Importance": np.abs(selector_class.estimator_.coef_[0])
}).sort_values("Importance", ascending=False)

lasso_importance_class.to_csv("lasso_class_importances.csv", index=False)
```

Model Lasso digunakan untuk menilai kontribusi setiap fitur terhadap target. Fitur yang memiliki koefisien mendekati nol dianggap tidak penting. Melalui cross-validation, model menentukan tingkat regularisasi terbaik. Hasil evaluasi koefisien kemudian disimpan dalam bentuk tabel ke file CSV.

2.4 PCA untuk Classification

Menjalankan PCA untuk mereduksi dimensi fitur klasifikasi. Data distandarisasi terlebih dahulu menggunakan StandardScaler, kemudian PCA dijalankan dan jumlah komponen yang dihasilkan ditampilkan ke layar.

```
# ============================================================
# A.2 — PCA UNTUK FITUR NUMERIK
# ============================================================

scaler_class = StandardScaler()

X_train_num = X_class_train[numeric_cols_class]
X_test_num  = X_class_test[numeric_cols_class]

X_train_scaled = scaler_class.fit_transform(X_train_num)
X_test_scaled  = scaler_class.transform(X_test_num)

# PCA (retain 95% variance)
pca_class = PCA(n_components=0.95, random_state=42)
pca_class.fit(X_train_scaled)

print("Jumlah PC (classification):", pca_class.n_components_)

X_train_pca = pca_class.transform(X_train_scaled)
X_test_pca  = pca_class.transform(X_test_scaled)

pc_cols_class = [f"PC{i+1}" for i in range(pca_class.n_components_)]

df_train_pca = pd.DataFrame(X_train_pca, columns=pc_cols_class)
df_test_pca  = pd.DataFrame(X_test_pca,  columns=pc_cols_class)

# Gabungkan dengan fitur non-numeric
non_num_cols_class = [c for c in X_class_train.columns if c not in numeric_cols_class]

X_class_train_pca = pd.concat([X_class_train[non_num_cols_class].reset_index(drop=True),
                               df_train_pca.reset_index(drop=True)], axis=1)

X_class_test_pca = pd.concat([X_class_test[non_num_cols_class].reset_index(drop=True),
                              df_test_pca.reset_index(drop=True)], axis=1)

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)
```

Bertujuan untuk membangun fondasi teknis dan data untuk optimasi fitur. Pertama, kode mengimpor semua *library* yang diperlukan dari scikit-learn, yaitu LassoCV dan SelectFromModel untuk seleksi fitur (LASSO), serta PCA untuk reduksi dimensi. Kedua, *file* data latih dan data uji yang telah selesai dibersihkan dan di-*scale* pada Tahap I (data_train_preprocessed.csv dan data_test_preprocessed.csv) dimuat ke dalam memori. Terakhir, data dipisahkan secara eksplisit menjadi fitur (X) dan target (y) untuk kedua

tugas, yaitu target Klasifikasi (Response) dan target Regresi (Income). Selama pemisahan, kedua kolom target ini dibuang dari *set* fitur (X_train_clean dan X_test_clean) untuk memastikan fitur input murni dan mencegah kebocoran informasi antar tugas pemodelan.

2.5 Seleksi Fitur untuk Tugas Klasifikasi (LASSO)

```
# ============================================================
# ---------------------- BAGIAN B ----------------------------
#     FEATURE ENGINEERING UNTUK REGRESI
#     Target: Pendapatan
# ============================================================

target_reg = "Pendapatan"

y_reg_train = train_df[target_reg]
y_reg_test  = test_df[target_reg]

X_reg_train = train_df.drop(columns=[target_reg]).copy()
X_reg_test  = test_df.drop(columns=[target_reg]).copy()

# Numeric PCA fields untuk regresi
numeric_cols_reg = ["Umur", "JumlahPinjaman", "LamaBekerja"]
numeric_cols_reg = [c for c in numeric_cols_reg if c in X_reg_train.columns]

print("Numeric cols (Regression PCA):", numeric_cols_reg)
```

Tahap ini mengimplementasikan strategi Seleksi Fitur menggunakan Regresi LASSO (L1 Regularization). Kode ini bertujuan untuk mengidentifikasi fitur-fitur dalam *set* X_train_clean yang paling signifikan dalam memprediksi target Klasifikasi (Response). Pertama, model LassoCV dilatih pada data latih, yang secara internal mencari kekuatan regularisasi (α) terbaik. Kedua, koefisien (*bobot*) yang dihasilkan LASSO digunakan sebagai indikator kepentingan fitur. Kami kemudian mendefinisikan ambang batas (*threshold*) berdasarkan nilai median dari koefisien absolut; fitur dengan koefisien di bawah median ini dianggap tidak penting dan dihilangkan. Terakhir, fungsi SelectFromModel menerapkan ambang batas ini untuk mentransformasi X_train_clean dan X_test_clean menjadi *set* data yang lebih ringkas (X_class_train_selected dan X_class_test_selected), yang kemudian disimpan sebagai *file* CSV yang siap untuk *modelling*.

2.6 Seleksi Fitur untuk Tugas Regresi (LASSO)

```
# ============================================================
# B.1 — LASSO FEATURE SELECTION (REGRESSION)
# ============================================================

max_feats_reg = min(15, X_reg_train.shape[1])

lasso_reg = Lasso(alpha=0.1, random_state=42, max_iter=10000)
selector_reg = SelectFromModel(lasso_reg, max_features=max_feats_reg)
selector_reg.fit(X_reg_train, y_reg_train)

selected_features_reg = X_reg_train.columns[selector_reg.get_support()]
print("Selected features (regression):")
print(selected_features_reg)

X_reg_train_selected = X_reg_train[selected_features_reg]
X_reg_test_selected  = X_reg_test[selected_features_reg]

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)

# Importance
lasso_importance_reg = pd.DataFrame({
    "Feature": X_reg_train.columns,
    "Importance": np.abs(selector_reg.estimator_.coef_)
}).sort_values("Importance", ascending=False)

lasso_importance_reg.to_csv("lasso_reg_importances.csv", index=False)
```

Tahap ini mengulang proses yang sama dengan tahap sebelumnya, namun kali ini difokuskan pada pengidentifikasian fitur-fitur terbaik untuk memprediksi target Regresi (Income). Sama seperti klasifikasi, model LassoCV dilatih pada X_train_clean, tetapi menggunakan target numerik y_train_reg. Karena targetnya berbeda, model LASSO akan memberikan koefisien (*bobot*) yang berbeda pada setiap fitur, mencerminkan kepentingan fitur tersebut dalam memprediksi pendapatan. Proses pemilihan (*selection*) dan transformasi data kemudian diulangi dengan ambang batas baru, menghasilkan *set* fitur baru yang dioptimalkan untuk regresi, disimpan sebagai X_reg_train_selected dan X_reg_test_selected. Hasil ini memungkinkan perbandingan apakah strategi seleksi fitur bekerja lebih baik untuk memprediksi *Response* atau *Income*.

2.7 Reduksi Dimensi PCA

```
# ============================================================
# B.2 — PCA UNTUK FITUR NUMERIK REGRESI
# ============================================================

scaler_reg = StandardScaler()

X_train_num_reg = X_reg_train[numeric_cols_reg]
X_test_num_reg  = X_reg_test[numeric_cols_reg]

X_train_scaled_reg = scaler_reg.fit_transform(X_train_num_reg)
X_test_scaled_reg  = scaler_reg.transform(X_test_num_reg)

pca_reg = PCA(n_components=0.95, random_state=42)
pca_reg.fit(X_train_scaled_reg)

print("Jumlah PC (regression):", pca_reg.n_components_)

X_train_pca_reg = pca_reg.transform(X_train_scaled_reg)
X_test_pca_reg  = pca_reg.transform(X_test_scaled_reg)

pc_cols_reg = [f"PC{i+1}" for i in range(pca_reg.n_components_)]

df_train_pca_reg = pd.DataFrame(X_train_pca_reg, columns=pc_cols_reg)
df_test_pca_reg  = pd.DataFrame(X_test_pca_reg,  columns=pc_cols_reg)

non_num_cols_reg = [c for c in X_reg_train.columns if c not in numeric_cols_reg]

X_reg_train_pca = pd.concat([X_reg_train[non_num_cols_reg].reset_index(drop=True),
                             df_train_pca_reg.reset_index(drop=True)], axis=1)

X_reg_test_pca = pd.concat([X_reg_test[non_num_cols_reg].reset_index(drop=True),
                            df_test_pca_reg.reset_index(drop=True)], axis=1)

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)
```

Tahap ini menerapkan Principal Component Analysis (PCA), sebuah teknik *unsupervised* yang bertujuan untuk mengurangi jumlah fitur secara signifikan sambil mempertahankan informasi (varians) maksimal. Pertama, model PCA diinisialisasi dengan parameter n_components=0.95, yang berarti PCA akan secara otomatis memilih jumlah *Principal Components* yang diperlukan untuk menjelaskan 95% dari total varians yang ada dalam data fitur. Kedua, model pca dilatih hanya pada data latih bersih (X_train_clean) untuk mencegah *data leakage*. Terakhir, model yang sudah dilatih tersebut digunakan untuk mentransformasi baik X_train_clean

maupun X_test_clean ke dalam ruang dimensi yang baru dan lebih rendah. Hasilnya (X_train_pca dan X_test_pca) disalin dan disimpan sebagai empat *file* CSV terpisah (X_class_pca dan X_reg_pca), karena hasil reduksi dimensi ini berlaku untuk kedua tugas pemodelan (Klasifikasi dan Regresi).

2.8 Peyimpanan Hasil Akhir

```
# ============================================================
# SIMPAN TARGET LABEL
# ============================================================

y_class_train.to_csv("y_class_train.csv", index=False)
y_class_test.to_csv("y_class_test.csv", index=False)
y_reg_train.to_csv("y_reg_train.csv", index=False)
y_reg_test.to_csv("y_reg_test.csv", index=False)

print("\n=== DONE: All Feature Engineering Outputs Generated ===")
```

Tahap akhir dari proses *Feature Engineering* ini adalah menyimpan semua *set* data yang telah dioptimalkan ke dalam *file* CSV yang terpisah. Kode ini secara sistematis menyimpan delapan *file* fitur (X)—empat dari hasil LASSO dan empat dari hasil PCA—bersama dengan empat *file* target (y_class dan y_reg) untuk set latih dan uji. Penyimpanan yang terperinci ini memungkinkan pengujian pemodelan yang terisolasi untuk setiap skenario: Klasifikasi vs. Regresi dan Seleksi Fitur vs. PCA. Dengan ini, seluruh *pipeline* pra-pemrosesan data (Tahap I dan Tahap II) telah selesai, dan data kini siap untuk digunakan dalam tahap *modelling* (pelatihan dan evaluasi model *machine learning*).

3. Kesimpulan

dua strategi optimasi fitur diterapkan. Pertama, Seleksi Fitur menggunakan Regresi LASSO (LassoCV) digunakan untuk mengidentifikasi dan memilih *subset* fitur yang paling relevan untuk masing-masing target (Response dan Income) dengan menetapkan ambang batas (*threshold*) pada koefisien fitur median. Kedua, Reduksi

Dimensi menggunakan PCA diterapkan sekali pada *set* fitur lengkap untuk merangkum sebagian besar informasi ke dalam sejumlah kecil *Principal Components*, dengan tujuan eksplisit mempertahankan 95% dari total varians data.

Sebagai hasil akhir, proses ini menghasilkan empat set data fitur (X) yang berbeda dan terisolasi: Klasifikasi (LASSO), Regresi (LASSO), Klasifikasi (PCA), dan Regresi (PCA). Dengan adanya pemisahan yang sistematis ini, data kini siap sepenuhnya untuk Tahap III, yaitu *Modelling* dan Eksperimen, di mana kinerja setiap skenario fitur akan diuji secara empiris untuk menentukan strategi optimasi mana yang paling efektif dalam meningkatkan akurasi dan efisiensi model *machine learning* pada kasus pinjaman online ini.