# IMPLEMENTASI PIPELINE GENERASI DATA SINTETIS MENGGUNAKAN CTGAN

Dosen Pengampu Mata Kuliah: Ir. Sidik Prabowo, S.T., M.T., CEH, IDPP

Universitas Telkom


Disusun Oleh:

| | |
|---|---|
| Muhammad Karov Ardava Barus | 103052300001 |
| Muhammad Al Fayyedh Denof | 103052330042 |
| Avatar Bintang Ramadhan | 103052300007 |
| Runa Raditya Rizki Hidayat | 103052300037 |

PROGRAM STUDI S1 SAINS DATA
FAKULTAS INFORMATIKA
UNIVERSITAS TELKOM
BANDUNG

2025

# Daftar Isi

# I. Pendahuluan

## 1.1 Latar Belakang

Pada minggu sebelumnya, kami telah merancang alur kerja (*pipeline*) untuk pembuatan data sintetis guna mengatasi masalah privasi dan ketidakseimbangan kelas pada dataset *Telco Customer Churn*. Minggu ke-4 ini berfokus pada tahap eksekusi teknis, yaitu mengimplementasikan rancangan tersebut ke dalam kode program menggunakan bahasa Python dan pustaka *Synthetic Data Vault* (SDV).

## 1.2 Tujuan

Tujuan dari laporan progres minggu ke-4 ini adalah:

1. Melakukan instalasi dan konfigurasi lingkungan pengembangan.
2. Melakukan *preprocessing* data untuk persiapan pelatihan model.
3. Melatih model *Conditional Tabular GAN* (CTGAN) menggunakan data asli.
4. Membangkitkan (*generate*) dataset sintetis sebanyak 2.000 sampel.

# II. Implementasi Teknis

## 2.1 Preprocessing Data

Sebelum data dimasukkan ke dalam model, dilakukan beberapa tahapan pembersihan:

1. Penghapusan Identitas Langsung: Kolom `customerID` dihapus karena merupakan *Direct Identifier* yang unik untuk setiap pengguna dan tidak boleh dipelajari oleh model generatif.
2. Konversi Tipe Data: Kolom `TotalCharges` dipastikan bertipe numerik, dan nilai kosong (*missing values*) diisi dengan nilai rata-rata (*mean imputation*).

```python
# Hapus Direct Identifiers
df_train = df.drop(columns=['customerID'])

# Handling Missing Values
df_train['TotalCharges'] = pd.to_numeric(df_train['TotalCharges'], errors='coerce')
df_train['TotalCharges'].fillna(df_train['TotalCharges'].mean(), inplace=True)
```

## 2.2 Pelatihan Model (Training)

Kami menggunakan algoritma `CTGANSynthesizer`. Metadata tabel (tipe data setiap kolom) dideteksi secara otomatis oleh library `sdv`. Model dilatih selama 100 epoch untuk memastikan model mempelajari distribusi data dengan cukup baik.

```python
from sdv.single_table import CTGANSynthesizer
from sdv.metadata import SingleTableMetadata

# Deteksi Metadata
metadata = SingleTableMetadata()
```

```python
metadata.detect_from_dataframe(df_train)

# Inisialisasi dan Training
synthesizer = CTGANSynthesizer(metadata, epochs=100, verbose=True)
synthesizer.fit(df_train)
```

### 2.3 Generasi Data (Sampling)

Setelah model dilatih, kami membangkitkan data sintetis. Sesuai dengan persyaratan tugas (minimal 1.500 sampel), kami membangkitkan 2.000 baris data.

```python
# Generate 2000 baris data
n_samples = 2000
synthetic_data = synthesizer.sample(num_rows=n_samples)

# Simpan ke CSV
synthetic_data.to_csv('output/synthetic_telco_churn.csv', index=False)
```

## III. Hasil Implementasi

### 3.1 Proses Training

Proses pelatihan berjalan lancar dengan indikator *loss* untuk Generator dan Discriminator yang terpantau stabil selama 100 epoch.

### 3.2 Sampel Data Sintetis

Berikut adalah cuplikan 5 baris pertama dari dataset sintetis yang berhasil dibangkitkan. Data ini memiliki struktur kolom yang sama persis dengan data asli (kecuali `customerID` yang memang sengaja tidak dibangkitkan).

### 3.3 Output File

File hasil generasi telah disimpan dengan nama `synthetic_telco_churn.csv` di dalam folder `output/`. File ini siap digunakan untuk tahap evaluasi utilitas dan privasi pada minggu berikutnya.

## IV. Kesimpulan

Implementasi *pipeline* generasi data sintetis pada Minggu 4 telah berhasil dilakukan. Model CTGAN sukses dilatih menggunakan dataset *Telco Customer Churn* yang telah diproses, dan menghasilkan 2.000 baris data sintetis. Langkah selanjutnya adalah melakukan evaluasi mendalam untuk membandingkan kualitas statistik dan privasi antara data asli dan data sintetis.