Topik : 3.1. Dataset Tabular Simulasi

Objective : Buat dataset dummy subsidi (Dinsos, Dukcapil, Kemenkes)

Task : Format CSV per client → load ke TFF

Source : https://www.w3schools.com/python/pandas/default.asp

PANDAS

- Pandas Adalah library python yang digunakan untuk bekerja dengan Kumpulan data
- Library ini memiliki fungsi untuk menganalisis, membersihkan , mengeksplorasi, dan memanipulasi data
- Nama “Pandas” merujuk pada “Panel Data” dan “Python Data Analysis” yang diciptakan oleh Wes McKinney pada tahun 2008.

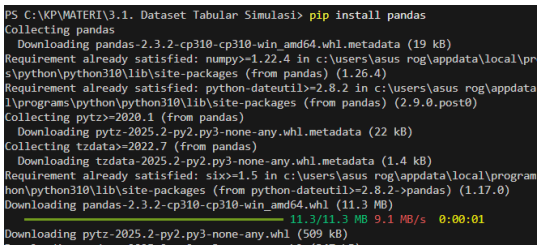
Mengapa menggunakan Pandas ?

- Pandas memungkinkan kita untuk menganalisis data besar dan menarik Kesimpulan berdasarkan teori statistic
- Pandas dapat membersihkan dataset yang berantakan dan membuatnya menjadi mudah dibaca dan relevan.
- Data yang relevan sangat penting dalam ilmu data

Apa yang dapat dilakukan oleh pandas ?

- Pandas dapat memberikan jawaban tentang korelasi data antara dua kolom atau lebih kolom
- Pandas dapat memberikan jawaban tentang data yang memiliki nilai rata – rata , nilai maksimum, dan nilai minimum.
- Pandas juga dapat menghapus baris yang tidak relevan atau mengandung nilai yang salah seperti nilai kosong atau null (ini disebut pembersihan data).

Installation of pandas



coba_pandas.py 1

```python
import pandas as pd

mydataset = {
    'cars' : ["bmw", "volvo", "ford"],
    'passing' : [3,7,2]
}

myvar = pd.DataFrame(mydataset)
print(myvar)
print(pd.__version__)
```

```
PS C:\KP\MATERI\3.1. Dataset Tabular Simulasi> python coba_pandas.py
    cars  passing
0    bmw        3
1  volvo        7
2   ford        2
2.3.2
PS C:\KP\MATERI\3.1. Dataset Tabular Simulasi>
```

Pandas Series

Panda series mirip dengan kolom dalam table. Ini Adalah array satu dimensi yang menyimpan data dari jenis apapun.

pandas_series.py 1

```python
import pandas as pd

a = [1,7,2]
myvar = pd.Series(a)
print(myvar)
print("")

#label
print("label = ",myvar[0])
print("")


#create label
myvar = pd.Series(a,index=["x","y","z"])
print("create label : ")
print(myvar)
print("")

#retrun value y
print("retrun value y : ", myvar["y"])
print("")

#key/value objects as series
calories = {"day1": 420, "day2":220, "day3":180}
myvar = pd.Series(calories)
print(myvar)
print("")


#only frame day 1 and day 2
myvar = pd.Series(calories, index=["day1", "day2"])
print(myvar)

```

```
PS C:\KP\MATERI\3.1. Dataset Tabular Simulasi> python pandas_series.py
0    1
1    7
2    2
dtype: int64

label =  1

create label :
x    1
y    7
z    2
dtype: int64

retrun value y :  7

day1    420
day2    220
day3    180
dtype: int64

day1    420
day2    220
dtype: int64
PS C:\KP\MATERI\3.1. Dataset Tabular Simulasi>
```

Pandas Read CSV

Cara sederhana untuk menyimpan set data besar Adalah dengan menggunakan berkas CSV (berkas yang dipisahkan koma). Berkas CSV berisi teks biasa dan merupakan format yang umum digunakan yang dapat dibaca oleh semua orang , termasuk pandas.

Contoh data_dummy.csv :

	A
1	ID,Nama,Umur,Kota,Pendapatan
2	1,Andi,23,Jakarta,5000000
3	2,Budi,30,Bandung,7000000
4	3,Citra,25,Surabaya,5500000
5	4,Dewi,28,Medan,6200000
6	5,Eka,22,Makassar,4800000
7	6,Fajar,35,Yogyakarta,8000000
8	7,Gina,27,Bali,5300000
9	8,Hadi,40,Semarang,9000000
10	9,Indah,26,Palembang,6000000
11	10,Joko,29,Bogor,7500000

```
pandas_ReadCSV.py 1
pandas_ReadCSV.py > ...
1   import pandas as pd
2
3   df = pd.read_csv('data_dummy.csv')
4   print(df.to_string())
5
6   #Jika Anda memiliki DataFrame besar dengan banyak baris,
7   # Pandas hanya akan mengembalikan
8   #  5 baris pertama dan 5 baris terakhir:
9   print("")
10  print(df)
11
12
```

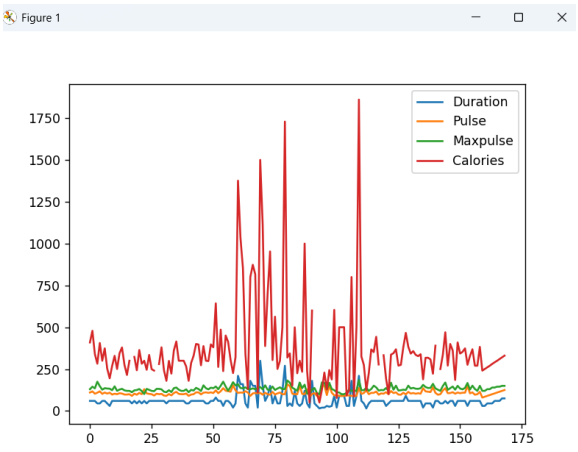
```
PS C:\KP\MATERI\3.1. Dataset Tabular Simulasi> python pandas_ReadCSV.py
   ID   Nama  Umur        Kota  Pendapatan
0   1   Andi    23     Jakarta     5000000
1   2   Budi    30     Bandung     7000000
2   3  Citra    25    Surabaya     5500000
3   4   Dewi    28       Medan     6200000
4   5    Eka    22    Makassar     4800000
5   6  Fajar    35  Yogyakarta     8000000
6   7   Gina    27        Bali     5300000
7   8   Hadi    40    Semarang     9000000
8   9  Indah    26   Palembang     6000000
9  10   Joko    29       Bogor     7500000

   ID   Nama  Umur        Kota  Pendapatan
0   1   Andi    23     Jakarta     5000000
1   2   Budi    30     Bandung     7000000
2   3  Citra    25    Surabaya     5500000
3   4   Dewi    28       Medan     6200000
4   5    Eka    22    Makassar     4800000
5   6  Fajar    35  Yogyakarta     8000000
6   7   Gina    27        Bali     5300000
7   8   Hadi    40    Semarang     9000000
8   9  Indah    26   Palembang     6000000
9  10   Joko    29       Bogor     7500000
PS C:\KP\MATERI\3.1. Dataset Tabular Simulasi>
```

Pandas Plotting

1. Plotting
   Pandas menggunakan metode plot() untuk membuat diagram. Kita dapat menggunakan Pyplot, sebuah submodule dari perpustakaan Matplotlib untuk menampilkan diagram di layer.

```
pandas_plotting.py > ...
1   import pandas as pd
2   import matplotlib.pyplot as plt
3
4   df = pd.read_csv("angka_plotting.csv")
5   df.plot()
6   plt.show()
7
```



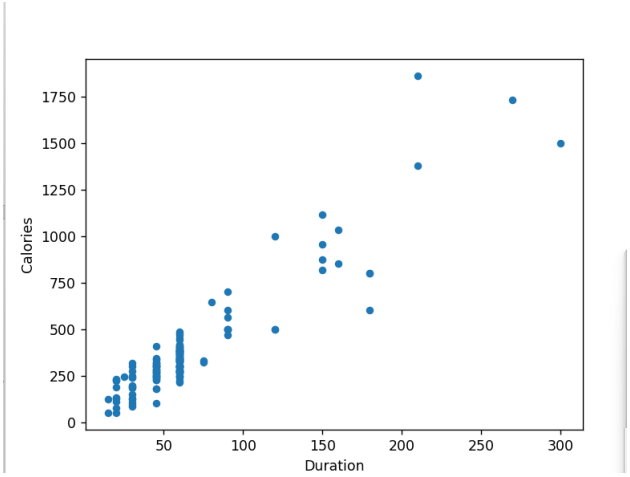
2. Scatter plot

   Tentukan bahwa kita ingin membuat scatter plot dengan kind argument :

   - Kind = ‘Scatter’

   Scatter plot memerlukan sumbu x dan y. dalam contoh ini, kita akan menggunakan “durasi” untuk sumbu x dan “kalori” untuk sumbu y.

```
9   df = pd.read_csv("angka_plotting.csv")
10  df.plot(kind = 'scatter', x = 'Duration', y = 'Calories')
11  plt.show()
```



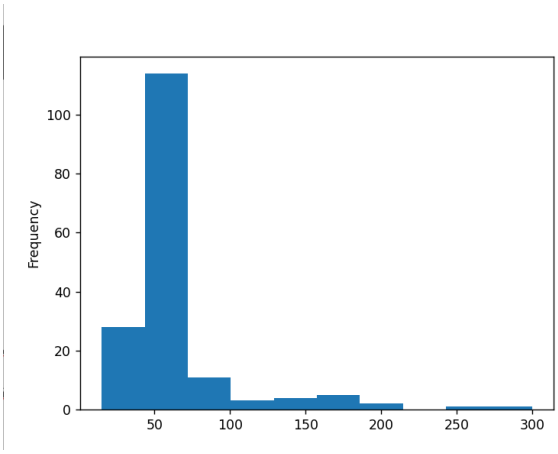
3. Histogram

   Gunakan argument ‘kind’ untuk menentukan bahwa anda ingin membuat histogram :

   - Kind = ‘hist’

   Histogram hanya memerlukan satu kolom. Histogram menunjukkan frekuensi setiap interval, misalnya, berapa banyak latihan yang berlangsung antara 50 dan 60 menit?

```
12
13  df = pd.read_csv("angka_plotting.csv")
14  df["Duration"].plot(kind= 'hist')
15  plt.show()
16
```



Source : https://scikit-learn.org/stable/api/sklearn.datasets.html

sklearn.datasets

LOADERS	
clear_data_home	Hapus semua akonten dari chace data home
dump_svmlight_file	Simpan dataset dalam format svmlight / libsvm
fetch_20newsgroups	Muat nama file dan data dari dataset 20 newsgroup (klasifikasi)
fetch_20newsgroups_vectorized	Muat dan vektorkan dataset 20 newsgroup (klasikasi)
fetch_california_housing	muat dataset perumahan California (regresi)
fetch_covtype	Muat dataset covtype (klasifikasi)
fetch_file	Ambil file dari web jika belum tersedia di folder local
fetch_kddcup99	Muat dataset kddcup99 (klasifikasi)
fetch_lfw_pairs	Muat dataset pasangan wajah Labeled Faces in the Wild (LFW) (klasifikasi)
fetch_lfw_people	Muat dataset orang Labeled Facess in the Wild (LFW) (klasifikasi)
fetch_olivetti_faces	Muat dataset wajah olivetti dari AT&T (klasifikasi)
fetch_openml	Ambil dataset dari openml berdasarkan nama atau id dataset
fetch_rcv1	Mulai dataset multilabel RCV1 (klasifikasi)
fetch_species_distributions	Loader untuk dataset distribusi spesies dari Phillips dkk/
get_data_home	Kembalikan path direktori data scikit-learn
load_breast_cancer	Muat dan kembalikan dataset kanker payudara Wisconsin (klasifikasi)
load_diabetes	Muat dan kembalikan dataset diabetes (regresi)
load_digits	Muat dan kembalikan dataset digit (klasifikasi)
load_files	Muat file teks dengan kategori sebagai nama subfolder
load_iris	Muat dan kembalikan dataset iris (klasifikasi)
load_linnerud	Muat dan kembalikan dataset Latihan fisik Linnerud
load_sample_image	Muat array numpy dari satu gambar sampel
load_sample_images	Muat gambar contoh untuk manipulasi gambar
load_svmlight_file	Muat dataset dalam format svmlight / libsvm ke dalam matriks CSR sparse
load_svmlight_files	Muat dataset dari beberapa file dalam format SVMlight

load_wine	Muat dan kembalikan dataset wine (klasifikasi)

Sample Generators	
make_biclusters	Hasilkan array struktur blog diagonal konstan untuk biclustering
make_blobs	Hasilkan blob Gaussian isotropic untuk clustering
make_checkerboard	Hasilkan array dengan structure papan kotak – kotak untuk biclustering
make_circles	Buat lingkaran besar yang berisi lingkaran kecil di dalamnya dalam 2D
make_classification	Hasilkan masalah klasifikasi acak dengan n kelas
make_friedman1	Hasilkan masalah regresi “Friedman #1”
make_friedman2	Hasilkan masalah regresi “Friedman #2”
make_friedman3	Hasilkan masalah regresi “Friedman #3”
make_gaussian_quantiles	Hasilkan Gaussian isotropic dan beri label sampel berdasarkan kuantil
make_hastie_10_2	Hasilkan data untuk klasifikasi biner yang digunakan pada Hestie dkk. 2009
make_low_rank_matrix	Hasilkan matriks peringkat rendah dengan nilai singular berbentuk lonceng
make_moons	Buat dua setengah lingkaran yang saling bertautan
make_multilabel_classification	Hasilkan masalah klasifikasi multilabel acak
make_regression	Hasilkan masalah regresi acak
make_s_curve	Hasilkan dataset kurva berbentuk S
make_sparse_coded_signal	Hasilkan sinyal sebagai kombinasi jarang (sparse) dari elemen kamus
make_sparse_spd_matrix	Hasilkan matriks jarang yang simetris dan positif-definit
make_sparse_uncorrelated	Hasilkan masalah regresi acak dengan desain jarang yang tidak berkorelasi
make_spd_matrix	Hasilkan matriks simetris acak yang positif-definit
make_swiss_roll	Hasilkan dataset berbentuk gulungan Swiss

Source : https://www.youtube.com/watch?v=I5mntelNnsM

https://www.kaggle.com/code/faviansulthanwafi/tugas-eda-kelompok-b/notebook

PREPROCESSING TABULAR DATA DENGAN PYTHON MENGGUNAKAN PANDAS DAN SCIKIT-LEARN :

1. Menggunakan data Dummy

PreProcessing.py | data_dummy.csv

data_dummy.csv > data

```
ID,Nama,Umur,Kota,Pendapatan
1,Andi,23,Jakarta,5000000
2,Budi,30,Bandung,7000000
3,Citra,25,Surabaya,5500000
4,Dewi,28,Medan,6200000
5,Eka,22,Makassar,4800000
6,Fajar,35,Yogyakarta,8000000
7,Gina,27,Bali,5300000
8,Hadi,40,Semarang,9000000
9,Indah,26,Palembang,6000000
10,Joko,29,Bogor,7500000
```

2. Import Data

PreProcessing.py 2 | pandas_ReadCSV.py 1

PreProcessing.py > ...

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder, StandardScaler

df = pd.read_csv('data_dummy.csv')
print(df)
```

```
PS C:\KP\MATERI\3.1. Dataset Tabular Simulasi> python pandas_plotting.py
PS C:\KP\MATERI\3.1. Dataset Tabular Simulasi> python PreProcessing.py
   ID   Nama  Umur        Kota  Pendapatan
0   1   Andi    23     Jakarta     5000000
1   2   Budi    30     Bandung     7000000
2   3  Citra    25    Surabaya     5500000
3   4   Dewi    28       Medan     6200000
4   5    Eka    22    Makassar     4800000
5   6  Fajar    35  Yogyakarta     8000000
6   7   Gina    27        Bali     5300000
7   8   Hadi    40    Semarang     9000000
8   9  Indah    26   Palembang     6000000
9  10   Joko    29       Bogor     7500000
PS C:\KP\MATERI\3.1. Dataset Tabular Simulasi>
```

3. Encoding column categorical (kota)
   Ada 2 pilihan :
   1. Label Encoding → setiap kota jadi angka unik
   2. One-Hot Encoding → tiap kota jadi kolom biner

```
# --- Label Encoding ---
le = LabelEncoder()
df['kota_label'] = le.fit_transform(df['kota'])

# --- One-Hot Encoding ---
df = pd.get_dummies(df, columns=['kota'])
```

4. Normalisasi / Standarisasi Data Numerik
   Kita scaling Umur dan Pendapatan biar punya skala yang sama

```
17  scaler = StandardScaler()
18  df[['Umur', 'Pendapatan']] = scaler.fit_transform(df[['Umur', 'Pendapatan']])
19
```

5. Output akhir

```
21  print(df)
```

```
PS C:\KP\MATERI\3.1. Dataset Tabular Simulasi> python PreProcessing.py
   ID   Nama      Umur  Pendapatan  ...  Kota_Palembang  Kota_Semarang  Kota_Surabaya  Kota_Yogyakarta
0   1   Andi -1.057497   -1.077261  ...           False          False          False            False
1   2   Budi  0.288408    0.429398  ...           False          False          False            False
2   3  Citra -0.672952   -0.700596  ...           False          False           True            False
3   4   Dewi -0.096136   -0.173266  ...           False          False          False            False
4   5    Eka -1.249769   -1.227926  ...           False          False          False            False
5   6  Fajar  1.249769    1.182727  ...           False          False          False             True
6   7   Gina -0.288408   -0.851262  ...           False          False          False            False
7   8   Hadi  2.211130    1.936056  ...           False           True          False            False
8   9  Indah -0.480680   -0.323931  ...            True          False          False            False
9  10   Joko  0.096136    0.806062  ...           False          False          False            False

[10 rows x 14 columns]
PS C:\KP\MATERI\3.1. Dataset Tabular Simulasi>
```

Kalau mau kelihatan semua outputnya , maka :

```
22  print(df.to_string())
```

```
PS C:\KP\MATERI\3.1. Dataset Tabular Simulasi> python PreProcessing.py
   ID   Nama      Umur  Pendapatan  Kota_Bali  Kota_Bandung  Kota_Bogor  Kota_Jakarta  Kota_Makassar  Kota_Medan  Kota_Palembang  Kota_Semarang  Kota_Surabaya  Kota_Yogyakarta
0   1   Andi -1.057497   -1.077261      False         False       False          True          False       False           False          False          False            False
1   2   Budi  0.288408    0.429398      False          True       False         False          False       False           False          False          False            False
2   3  Citra -0.672952   -0.700596      False         False       False         False          False       False           False          False           True            False
3   4   Dewi -0.096136   -0.173266      False         False       False         False          False        True           False          False          False            False
4   5    Eka -1.249769   -1.227926      False         False       False         False           True       False           False          False          False            False
5   6  Fajar  1.249769    1.182727      False         False       False         False          False       False           False          False          False             True
6   7   Gina -0.288408   -0.851262       True         False       False         False          False       False           False          False          False            False
7   8   Hadi  2.211130    1.936056      False         False       False         False          False       False           False          False           True            False
8   9  Indah -0.480680   -0.323931      False         False       False         False          False       False            True          False          False            False
9  10   Joko  0.096136    0.806062      False         False        True         False          False       False           False          False          False            False
PS C:\KP\MATERI\3.1. Dataset Tabular Simulasi>
```

Task : CSV per Client → Load ke TFF

1. Format CSV per client

   Dalam federated learning, data biasanya tidak digabung menjadi satu, tetapi tersimpan di masing – masing client. Artinya :

   - Dinsos.csv (100 baris data) → client 1
   - Dukcapil.csv (100 baris data) → client 2
   - Kemenkes.csv (100 baris data) → client 3

2. Buat File CSV nya

```python
df = pd.read_csv('dinsos_100.csv')
df2 = pd.read_csv('dukcapil_100.csv')
df3 = pd.read_csv('kemenkes_100.csv')

print("DINSOS")
print(df)
print("")
print("DUKCAPIL")
print(df2)
print("")
print("KEMENKES")
print(df3)
```

```
PS C:\KP\MATERI\3.1. Dataset Tabular Simulasi> python pandas_ReadCSV.py
DINSOS
    jumlah_tanggungan  penghasilan     kondisi_rumah  layak_subsidi
0                   4      1925472       tidak layak              1
1                   5       548984             layak              1
2                   3      2900070  sangat sederhana              1
3                   5      3765879  sangat sederhana              0
4                   5      2004384  sangat sederhana              0
..                ...          ...               ...            ...
95                  4      1619043       tidak layak              1
96                  4      3829462  sangat sederhana              1
97                  5      2706903             layak              0
98                  1      2241093  sangat sederhana              0
99                  5      3058231  sangat sederhana              1

[100 rows x 4 columns]

DUKCAPIL
    umur status_pekerjaan status_pernikahan
0     43     pengangguran            cerai
1     54        pensiunan            cerai
2     43     pengangguran     belum menikah
3     40     pengangguran     belum menikah
4     26     pengangguran           menikah
..   ...              ...              ...
95    68         karyawan           menikah
96    67        pensiunan           menikah
97    52            buruh           menikah
98    40          pelajar            cerai
99    34            buruh     belum menikah

[100 rows x 3 columns]

KEMENKES
   riwayat_penyakit status_gizi  tinggi_cm  berat_kg
0       hipertensi        baik        146        51
1         diabetes        baik        142        86
2            sehat        baik        186        40
3       hipertensi        baik        162        65
4             asma       lebih        185        53
..             ...         ...        ...       ...
95            asma        baik        158        94
96           sehat        baik        159        41
97           sehat      kurang        172        53
98            asma        baik        159        92
99            asma        baik        188        79

[100 rows x 4 columns]
PS C:\KP\MATERI\3.1. Dataset Tabular Simulasi>
```

3. Preprocessing : Encoding + samakan jumlah fitur

```
# ============================================================
# 2. Preprocessing: Encoding kategori + samakan jumlah fitur
# ============================================================

# ---- DINSOS ----
enc_dinsos = LabelEncoder()
dinsos['kondisi_rumah'] = enc_dinsos.fit_transform(dinsos['kondisi_rumah'])
# label sudah ada: layak_subsidi (0/1)
dinsos['dummy'] = 0   # biar total 4 fitur

# ---- DUKCAPIL ----
enc_pekerjaan = LabelEncoder()
enc_nikah = LabelEncoder()
dukcapil['status_pekerjaan'] = enc_pekerjaan.fit_transform(dukcapil['status_pekerjaan'])
dukcapil['status_pernikahan'] = enc_nikah.fit_transform(dukcapil['status_pernikahan'])
dukcapil['layak_subsidi'] = (dukcapil['umur'] > 40).astype(int)  # dummy rule
dukcapil['dummy'] = 0   # biar total 4 fitur

# ---- KEMENKES ----
enc_penyakit = LabelEncoder()
enc_gizi = LabelEncoder()
kemenkes['riwayat_penyakit'] = enc_penyakit.fit_transform(kemenkes['riwayat_penyakit'])
kemenkes['status_gizi'] = enc_gizi.fit_transform(kemenkes['status_gizi'])
kemenkes['layak_subsidi'] = (kemenkes['berat_kg'] > 60).astype(int)  # dummy rule
# sudah 4 fitur: riwayat_penyakit, status_gizi, tinggi_cm, berat_kg

print("Data siap (contoh Dinsos):\n", dinsos.head())

```

- Dinsos → encode kondisi_rumah (kategori teks jadi angka) + tambahkan kolom dummy biar total fiturnya 4.
- Dukcapil → encode status_pekerjaan & status_pernikahan, buat label layak_subsidi (rule: umur > 40), tambah kolom dummy.
- Kemenkes → encode riwayat_penyakit & status_gizi, buat label layak_subsidi (rule: berat_kg > 60), sudah punya 4 fitur jadi tidak perlu dummy.

4. Fungsi convert DataFrame → tf.dataset

```
# ============================================================
# 3. Fungsi Convert DataFrame -> tf.data.Dataset
# ============================================================
def make_dataset_from_df(df, label_col):
    labels = df[label_col].astype('int32').values   # pastikan int32
    features = df.drop(columns=[label_col]).astype('float32').values
    dataset = tf.data.Dataset.from_tensor_slices((features, labels))
    dataset = dataset.shuffle(buffer_size=len(df))
    dataset = dataset.batch(20)
    return dataset
```

- Mengubah pandas.DataFrame jadi tf.data.Dataset → format yang dipahami oleh TFF
- Dataset di-shuffle agar training tidak bias urutan
- Dibagi batch ukuran 20
- Output per elemen: (fitur, label) ⟶ (4 angka, 1 target biner).

5. Buat Federated Data (3 client)

```
# ======================================================
# 4. Buat Federated Data (3 Client)
# ======================================================
client_dinsos = make_dataset_from_df(dinsos, 'layak_subsidi')
client_dukcapil = make_dataset_from_df(dukcapil, 'layak_subsidi')
client_kemenkes = make_dataset_from_df(kemenkes, 'layak_subsidi')

federated_train_data = [client_dinsos, client_dukcapil, client_kemenkes]
print(f"Total clients: {len(federated_train_data)}")
```

```
.
Skipping registering GPU devices...
Total clients: 3
```

   - Tiap dataset dijadikan 1 client
   - Semua client digabungkan ke list federated_train_data
   - Jadi, federated learning akan melatih 3 model

6. Definisikan model

```
# ======================================================
# 5. Definisikan Model TFF
# ======================================================
input_dim = federated_train_data[0].element_spec[0].shape[1]  # 4 fitur

def model_fn():
    keras_model = tf.keras.Sequential([
        tf.keras.layers.Input(shape=(input_dim,)),  # 4 fitur
        tf.keras.layers.Dense(16, activation='relu'),
        tf.keras.layers.Dense(2, activation='softmax')  # biner: 0/1
    ])

    return tff.learning.models.from_keras_model(
        keras_model,
        input_spec=federated_train_data[0].element_spec,
        loss=tf.keras.losses.SparseCategoricalCrossentropy(),
        metrics=[tf.keras.metrics.SparseCategoricalAccuracy()]
    )
```

   - Model Keras Sederhana
     - Input : 4 fitur
     - Hidden Layer : Dense 16 neuron dengan aktivasi ReLU.
     - Output Layer : Dense 2 neuron dengan softmax → klasifikasi biner (tidak layak=0, layak=1).

7. Federated Training

```
# ======================================================
# 6. Federated Training
# ======================================================
iterative_process = tff.learning.algorithms.build_weighted_fed_avg(
    model_fn,
    client_optimizer_fn=tff.learning.optimizers.build_sgdm(learning_rate=0.01),
    server_optimizer_fn=tff.learning.optimizers.build_sgdm(learning_rate=1.0)
)

state = iterative_process.initialize()
```

   - Menggunakan Federated Averaging (FedAvg).
   - Optimizer :
     - **Client** → SGD learning rate 0.02 (update model lokal).
     - **Server** → SGD learning rate 1.0 (gabungkan model dari client).
   - state = iterative_process.initialize() → inisialisasi model global.\

8. Training loop

```python
for round_num in range(1, 6):
    result = iterative_process.next(state, federated_train_data)
    state, metrics = result.state, result.metrics
    acc = metrics['client_work']['train']['sparse_categorical_accuracy']
    loss = metrics['client_work']['train']['loss']
    print(f'Round {round_num} - Loss: {loss:.4f}, Accuracy: {acc:.4f}')
```

```
Skipping registering GPU devices...
Round 1 - Loss: 5.3894, Accuracy: 0.5433
Round 2 - Loss: 5.1882, Accuracy: 0.5500
Round 3 - Loss: 5.0325, Accuracy: 0.5633
Round 4 - Loss: 5.1956, Accuracy: 0.5700
Round 5 - Loss: 5.1236, Accuracy: 0.5700
(venv) ezranahumury@DESKTOP-8003BIM:/mnt/c/KP/MATERI/3.1. Dataset Tabular Simulasi/Task$
```

   - Round Federated Learning
     - Model global dikirim ke client
     - Client melatih model dengan datanya masing – masing
     - Update model dengan datanya masing – masing
     - Server menggabungkan (average) update → model global baru