Topik : 3.3. Training Federated dengan Tabular Data

Objective : Latih FL model dengan dataset dummy 3 client

Task : Simulasikan 5-10 round + logging akurasi

Source : https://medium.com/@irfandy.thalib/menggabungkan-2-dataset-dengan-pandas-dan-eksport-ke-csv-167b535c4f45

Menggabungkan 3 Dataset Dengan Pandas dan Ekspor ke CSV

OPSI 1 : Pandas akan meng-*union* semua kolom dari ketiga file

Hasil akhirnya: file gabungan.csv memang punya semua kolom, tapi sebagian besar nilai jadi NaN karena tiap instansi hanya punya subset kolom tertentu.

gabungan.py 1

```python
import pandas as pd

#Memanggil Dataset
data1 = pd.read_csv('dinsos_100.csv')
data2 = pd.read_csv('dukcapil_100.csv')
data3 = pd.read_csv('kemenkes_100.csv')


#Merge
frames = [data1, data2, data3]
result = pd.concat(frames).drop_duplicates().reset_index(drop=True)

#ekspor ke new file csv
result.to_csv(r'gabungan.csv', index=False)
```

```
   jumlah_tanggungan  penghasilan     kondisi_rumah  layak_subsidi  umur status_pekerjaan status_pernikahan riwayat_penyakit status_gizi  tinggi_cm  berat_kg
0                4.0    1925472.0       tidak layak            1.0   NaN            NaN              NaN             NaN         NaN        NaN       NaN
1                5.0     548984.0             layak            1.0   NaN            NaN              NaN             NaN         NaN        NaN       NaN
2                3.0    2900070.0  sangat sederhana            1.0   NaN            NaN              NaN             NaN         NaN        NaN       NaN
3                5.0    3765879.0  sangat sederhana            0.0   NaN            NaN              NaN             NaN         NaN        NaN       NaN
4                5.0    2004384.0  sangat sederhana            0.0   NaN            NaN              NaN             NaN         NaN        NaN       NaN
..               ...          ...               ...            ...   ...            ...              ...             ...         ...        ...       ...
288              NaN          NaN               NaN            NaN   NaN            NaN              NaN           asma        baik      158.0      94.0
289              NaN          NaN               NaN            NaN   NaN            NaN              NaN          sehat        baik      159.0      41.0
290              NaN          NaN               NaN            NaN   NaN            NaN              NaN          sehat      kurang      172.0      53.0
2                3.0    2900070.0  sangat sederhana            1.0   NaN            NaN              NaN             NaN         NaN        NaN       NaN
3                5.0    3765879.0  sangat sederhana            0.0   NaN            NaN              NaN             NaN         NaN        NaN       NaN
4                5.0    2004384.0  sangat sederhana            0.0   NaN            NaN              NaN             NaN         NaN        NaN       NaN
..               ...          ...               ...            ...   ...            ...              ...             ...         ...        ...       ...
288              NaN          NaN               NaN            NaN   NaN            NaN              NaN           asma        baik      158.0      94.0
289              NaN          NaN               NaN            NaN   NaN            NaN              NaN          sehat        baik      159.0      41.0
290              NaN          NaN               NaN            NaN   NaN            NaN              NaN          sehat      kurang      172.0      53.0
291              NaN          NaN               NaN            NaN   NaN            NaN              NaN           asma        baik      159.0      92.0
..               ...          ...               ...            ...   ...            ...              ...             ...         ...        ...       ...
288              NaN          NaN               NaN            NaN   NaN            NaN              NaN           asma        baik      158.0      94.0
289              NaN          NaN               NaN            NaN   NaN            NaN              NaN          sehat        baik      159.0      41.0
290              NaN          NaN               NaN            NaN   NaN            NaN              NaN          sehat      kurang      172.0      53.0
291              NaN          NaN               NaN            NaN   NaN            NaN              NaN           asma        baik      159.0      92.0
292              NaN          NaN               NaN            NaN   NaN            NaN              NaN           asma        baik      188.0      79.0
289              NaN          NaN               NaN            NaN   NaN            NaN              NaN          sehat        baik      159.0      41.0
290              NaN          NaN               NaN            NaN   NaN            NaN              NaN          sehat      kurang      172.0      53.0
291              NaN          NaN               NaN            NaN   NaN            NaN              NaN           asma        baik      159.0      92.0
292              NaN          NaN               NaN            NaN   NaN            NaN              NaN           asma        baik      188.0      79.0
291              NaN          NaN               NaN            NaN   NaN            NaN              NaN           asma        baik      159.0      92.0
292              NaN          NaN               NaN            NaN   NaN            NaN              NaN           asma        baik      188.0      79.0
292              NaN          NaN               NaN            NaN   NaN            NaN              NaN           asma        baik      188.0      79.0

[293 rows x 11 columns]
```

OPSI 2 : Column-Wise → ketiga client digabung menjadi baris gabungan

```python
import pandas as pd

# Load CSV per instansi
dinsos = pd.read_csv("dinsos_100.csv")
dukcapil = pd.read_csv("dukcapil_100.csv")
kemenkes = pd.read_csv("kemenkes_100.csv")


# Pastikan jumlah baris sama
assert len(dinsos) == len(dukcapil) == len(kemenkes), "Jumlah baris dataset berbeda!"

# Gabung berdasarkan index (side by side)
gabungan = pd.concat([dinsos.reset_index(drop=True),
                      dukcapil.reset_index(drop=True),
                      kemenkes.reset_index(drop=True)], axis=1)

# Simpan ke CSV baru
gabungan.to_csv("dataset_gabungan.csv", index=False)
```

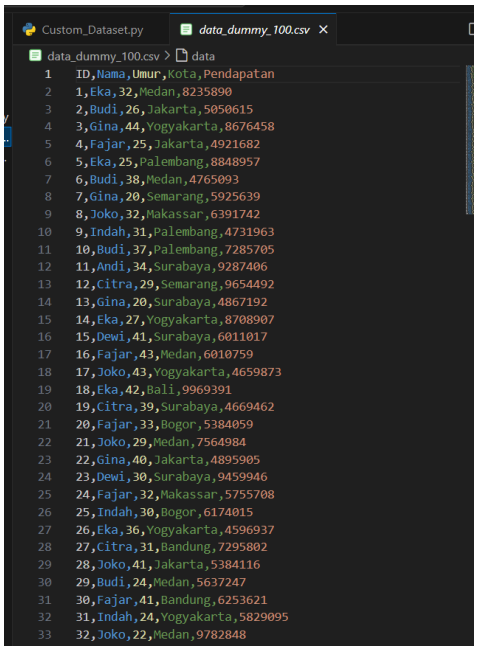
```
    jumlah_tanggungan  penghasilan     kondisi_rumah  layak_subsidi  umur status_pekerjaan status_pernikahan riwayat_penyakit status_gizi  tinggi_cm  berat_kg
0                   4     1925472       tidak layak              1    43     pengangguran            cerai       hipertensi         baik        146        51
1                   5      548984             layak              1    54        pensiunan            cerai         diabetes         baik        142        86
2                   3     2900070  sangat sederhana              1    43     pengangguran     belum menikah           sehat         baik        186        40
3                   5     3765879  sangat sederhana              0    40     pengangguran     belum menikah       hipertensi         baik        162        65
4                   5     2004384  sangat sederhana              0    26     pengangguran          menikah            asma        lebih        185        53
..                ...         ...               ...            ...   ...              ...              ...             ...         ...        ...       ...
95                  4     1619043       tidak layak              1    68         karyawan          menikah            asma         baik        158        94
96                  4     3829462  sangat sederhana              1    67        pensiunan          menikah           sehat         baik        159        41
97                  5     2706903             layak              0    52            buruh          menikah           sehat       kurang        172        53
98                  1     2241093  sangat sederhana              0    40           pelajar            cerai            asma         baik        159        92
99                  5     3058231  sangat sederhana              1    34            buruh     belum menikah            asma         baik        188        79

[100 rows x 11 columns]
(venv) ezranahumury@DESKTOP-8003BIM:/mnt/c/KP/MATERI/3.3. Training Federated dengan Tabular Data$
```

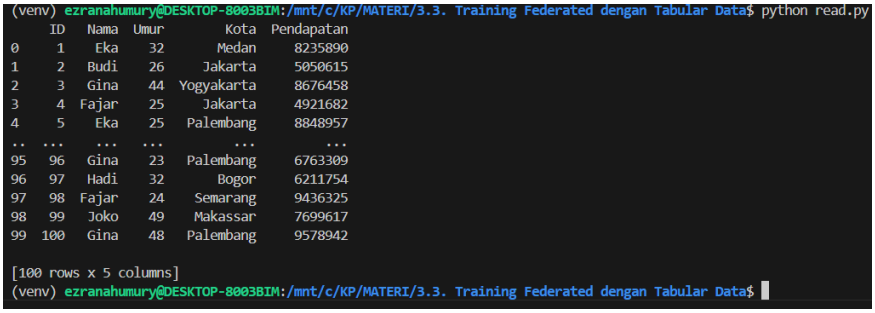
Pandas + TFF custom dataset examples

Referensi :
https://colab.research.google.com/github/tensorflow/federated/blob/v0.88.0/docs/tutorials/federated_learning_for_image_classification.ipynb

Federated Learning for custom datasets, bagian custom dataset itu Adalah Ketika kita tidak memakai dataset bawaan TFF (Seperti EMNIST, Shakespeare, StackOverflow) , tapi justru mempersiapkan sendiri dataset kita (misalnya CSV dengan Pandas) lalu mengubahnya menjadi tf.data.Dataset untuk tiap klient.

Menggunakan Dataset Dummy :





1. Menyiapkan Dataset dengan pandas

```python
# 1. Load & Label
df = pd.read_csv("data_dummy_100.csv")
df['layak_subsidi'] = (df["Pendapatan"] > 6000000).astype(int)

# 2. Ambil kolom numerik
df_num = df[["Umur", "Pendapatan", "layak_subsidi"]]

# Normalisasi fitur (Umur & Pendapatan)
df_num["Umur"] = (df_num["Umur"] - df_num["Umur"].mean()) / df_num["Umur"].std()
df_num["Pendapatan"] = (df_num["Pendapatan"] - df_num["Pendapatan"].mean()) / df_num["Pendapatan"].std()

x = df_num.drop("layak_subsidi", axis=1)
y = df_num["layak_subsidi"]
```

2. Konversi ke Tensorflow Dataset

```python
def make_tf_dataset(x, y, batch_size=8):
    feats = tf.cast(x.values, tf.float32)
    labels = tf.cast(y.values, tf.float32)
    ds = tf.data.Dataset.from_tensor_slices((feats, labels))
    ds = ds.shuffle(buffer_size=len(x)).batch(batch_size)
    return ds
```

- tf.data.Dataset dipakai TFF untuk baca data.

3. Bungkus menjadi federated Dataset

```
clients = []
num_clients = 3
split_df = np.array_split(df_num, num_clients)
for client_df in split_df:
    Xc = client_df[["Umur", "Pendapatan"]]
    Yc = client_df["layak_subsidi"]
    clients.append(make_tf_dataset(Xc, Yc))
```

- Dalam Federated Learning, data dibagi per klien.
- Misalnya kita punya 3 klien → dataset dibagi jadi 3.
- Setiap klien punya tf.data.Dataset masing-masing.
- Hasil akhirnya: clients = list of dataset per client.

4. Definisikan model keras

```
def create_keras_model():
    model = keras.Sequential([
        layers.Input(shape=(2,), dtype=tf.float32, name="features"),
        layers.Dense(16, activation='relu'),
        layers.Dense(1, activation='sigmoid')
    ])
    return model
```

- Model sederhana: input → hidden layer → output biner.
- Input shape (2,) karena kita pakai **2 fitur** (Umur, Pendapatan).
- Hidden layer: 16 neuron dengan aktivasi ReLU.
- Output: sigmoid → cocok untuk klasifikasi biner.

5. Wrap ke TFF

```
input_spec = clients[0].element_spec

def model_fn():
    keras_model = create_keras_model()
    return tff.learning.models.from_keras_model(
        keras_model,
        input_spec=input_spec,
        loss=tf.keras.losses.BinaryCrossentropy(),
        metrics=[tf.keras.metrics.BinaryAccuracy()]
    )
```

- input_spec diambil dari dataset klien → memberitahu TFF bentuk data.
- from_keras_model membungkus model Keras ke dalam format TFF.
- Kita tentukan **loss** (Binary Crossentropy) dan **metrics** (Binary Accuracy).

6. Training dengan Federated Averaging

```
iterative_process = tff.learning.algorithms.build_weighted_fed_avg(
    model_fn,
    client_optimizer_fn=tff.learning.optimizers.build_sgdm(learning_rate=0.05),
    server_optimizer_fn=tff.learning.optimizers.build_sgdm(learning_rate=1.0)
)

state = iterative_process.initialize()

for round_num in range(1, 11):
    state, metrics = iterative_process.next(state, clients)
    print(f"Round {round_num}, metrics={metrics}")
```

Ringkasan Alur

1. **Pandas** → load CSV, bersihkan, buat label.
2. **TensorFlow Dataset** → konversi tabular ke tf.data.Dataset.
3. **Federated Dataset** → split dataset per klien.
4. **Model Keras** → desain NN sederhana.
5. **Wrapper TFF** → bungkus model supaya TFF paham input spec.
6. **FedAvg Training** → jalankan federated learning dengan beberapa round.

Task :

Ringkasan Dataset

1. **Dinsos**
    - Fitur: jumlah_tanggungan, penghasilan, kondisi_rumah
    - Label: layak_subsidi
2. **Dukcapil**
    - Fitur: umur, status_pekerjaan, status_pernikahan
    - Label: layak_subsidi
3. **Kemenkes**
    - Fitur: riwayat_penyakit, status_gizi, tinggi_cm, berat_kg
    - Label: layak_subsidi

Menggunakan 5 Round :

```python
trainer = tff.learning.algorithms.build_weighted_fed_avg(
    model_fn,
    client_optimizer_fn=tff.learning.optimizers.build_sgdm(learning_rate=0.05),
    server_optimizer_fn=tff.learning.optimizers.build_sgdm(learning_rate=1.0)
)

state = trainer.initialize()

for round_num in range(1, 6):  # ganti ke range(1, 11) kalau mau 10 round
    result = trainer.next(state, federated_train_data)
    state = result.state
    metrics = result.metrics
    acc = float(metrics["client_work"]["train"]["binary_accuracy"])
    loss = float(metrics["client_work"]["train"]["loss"])
    print(f"Round {round_num} -> acc={acc:.4f}, loss={loss:.4f}")
```

```
Skipping registering GPU devices...
Round 1 -> acc=0.4993, loss=0.6985
Round 2 -> acc=0.5122, loss=0.6951
Round 3 -> acc=0.5118, loss=0.6941
Round 4 -> acc=0.5182, loss=0.6936
Round 5 -> acc=0.5169, loss=0.6933
(venv) ezranahumury@DESKTOP-8003BIM:/mnt/c/KP/MATERI/3.3. Training Federated dengan Tabular Data$
```

Menggunakan 10 Round :

```python
trainer = tff.learning.algorithms.build_weighted_fed_avg(
    model_fn,
    client_optimizer_fn=tff.learning.optimizers.build_sgdm(learning_rate=0.05),
    server_optimizer_fn=tff.learning.optimizers.build_sgdm(learning_rate=1.0)
)

state = trainer.initialize()

for round_num in range(1, 11):
    result = trainer.next(state, federated_train_data)
    state = result.state
    metrics = result.metrics
    acc = float(metrics["client_work"]["train"]["binary_accuracy"])
    loss = float(metrics["client_work"]["train"]["loss"])
    print(f"Round {round_num} -> acc={acc:.4f}, loss={loss:.4f}")
```

```
Skipping registering GPU devices...
Round 1 -> acc=0.5176, loss=0.7020
Round 2 -> acc=0.5182, loss=0.6966
Round 3 -> acc=0.5262, loss=0.6951
Round 4 -> acc=0.5191, loss=0.6941
Round 5 -> acc=0.5193, loss=0.6935
Round 6 -> acc=0.5178, loss=0.6933
Round 7 -> acc=0.5218, loss=0.6927
Round 8 -> acc=0.5193, loss=0.6928
Round 9 -> acc=0.5229, loss=0.6926
Round 10 -> acc=0.5309, loss=0.6923
(venv) ezranahumury@DESKTOP-8003BIM:/mnt/c/KP/MATERI/3.3. Training Federated dengan Tabular Data$
```

- **Simulasi 5–10 round** → berarti kita latih model federated selama beberapa putaran (round).
- **Logging akurasi** → setiap round, kita cetak akurasi (dan loss) supaya bisa memantau apakah model makin bagus.