

## Soal Mentoring 1 - Machine Learning Process

Job Preparation Program, Data Science, Pacmann AI

### Catatan:

- Make a copy docs ini sebelum menjawab.
- Studi kasus pada soal ini akan **digunakan** untuk mentoring selanjutnya.
- Tulis jawaban Anda dalam word processor dan ekspor ke PDF untuk soal nomor 1 dan 2
- Tulis jawaban Anda dalam jupyter notebook dan python script untuk soal nomor 3 dan 4
- Archive file PDF dan keseluruhan root folder project Anda ke ZIP, **kecuali folder venv**
- Beri nama [NAMA LENGKAP]\_MLPROCESS\_1 file archive Anda
- Submit file archive Anda melalui link submission yang telah disediakan

### Soal Pengantar

- Posisikan diri Anda sebagai Data Scientist yang bekerja dibidang perbankan.
- Produk pemberian pinjaman (loan) adalah salah satu sumber revenue di industri perbankan.
- Bank akan mengalami kerugian apabila customer gagal mengembalikan pinjaman (gagal bayar – default).
- Anda diminta untuk membuat sistem untuk prediksi customer akan gagal bayar atau tidak.

### Soal

1. [25 points] Project Environment Setup
  - a. [5 points] Buat folder project
    - Buat folder dengan nama berformat [NAMA\_ANDA]\_MLPROCESS  
Gunakan 1 kata pertama dari nama Anda  
Folder ini akan menjadi ROOT FOLDER PROJECT Anda  
Contoh: ADIT\_MLPROCESS
    - Buat subfolder bernama data, models, dan src dalam folder project Anda

- Buat subfolder bernama raw, interim, dan processed dalam folder data
- **Lampirkan screenshot folder tree yang telah Anda buat**

```
This message is shown once a day. To disable it please create the
/home/chalimasadiah/.hushlogin file.
chalimasadiah@LAPTOP-CH00NPOB:/mnt/c/WINDOWS/system32$ cd
chalimasadiah@LAPTOP-CH00NPOB:~$ mkdir CHALIMA_MLPROCESS
chalimasadiah@LAPTOP-CH00NPOB:~$ cd CHALIMA_MLPROCESS/
chalimasadiah@LAPTOP-CH00NPOB:~/CHALIMA_MLPROCESS$ mkdir data data/raw data/interim data/processed models src
chalimasadiah@LAPTOP-CH00NPOB:~/CHALIMA_MLPROCESS$ tree
.
├── data
│   ├── interim
│   ├── processed
│   └── raw
└── models
└── src

6 directories, 0 files
chalimasadiah@LAPTOP-CH00NPOB:~/CHALIMA_MLPROCESS$
```

- b. [5 points] Buat python virtual environment (venv)

- Buatlah venv dengan nama berformat [NAMA\_ANDA]\_ENV

Buat venv di dalam folder project Anda

Gunakan 1 kata pertama dari nama Anda

Contoh: ADIT\_VENV

- **Lampirkan screenshot terminal yang menampilkan command saat Anda membuat venv tersebut**

```
chalimasadiah@LAPTOP-CH00NPOB:~/CHALIMA_MLPROCESS$ python3 -m venv .CHALIMA_VENV
chalimasadiah@LAPTOP-CH00NPOB:~/CHALIMA_MLPROCESS$
```

- **Lampirkan screenshot terminal yang menampilkan command saat Anda mengaktifkan venv tersebut**

```
chalimasadiah@LAPTOP-CH00NPOB:~/CHALIMA_MLPROCESS$ source .CHALIMA_VENV/bin/activate
(.CHALIMA_VENV) chalimasadiah@LAPTOP-CH00NPOB:~/CHALIMA_MLPROCESS$
```

c. [5 points] Update PIP

- **Lampirkan screenshot terminal yang menampilkan command saat Anda melakukan update PIP**

```
(.CHALIMA_VENV) chalimasadiah@LAPTOP-CH00NPOB:~/CHALIMA_MLPROCESS$ pip install --upgrade pip
Requirement already satisfied: pip in ./CHALIMA_VENV/lib/python3.12/site-packages (24.0)
Collecting pip
  Using cached pip-25.3-py3-none-any.whl.metadata (4.7 kB)
Using cached pip-25.3-py3-none-any.whl (1.8 MB)
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 24.0
    Uninstalling pip-24.0:
      Successfully uninstalled pip-24.0
Successfully installed pip-25.3
(.CHALIMA_VENV) chalimasadiah@LAPTOP-CH00NPOB:~/CHALIMA_MLPROCESS$
```

d. [5 points] Install dependencies

- Install package berikut:

- pandas
- scikit-learn
- imblearn
- joblib
- numpy
- scipy
- seaborn
- fastapi
- jupyterlab
- requests

- **Lampirkan screenshot terminal yang menampilkan command saat Anda melakukan pemasangan packages tersebut**

```
(.CHALIMA_VENV) chalimasadiah@LAPTOP-CH00NPOB:~/CHALIMA_MLPROCESS$ touch requirements.txt
(.CHALIMA_VENV) chalimasadiah@LAPTOP-CH00NPOB:~/CHALIMA_MLPROCESS$ nano requirements.txt
(.CHALIMA_VENV) chalimasadiah@LAPTOP-CH00NPOB:~/CHALIMA_MLPROCESS$ nano requirements.txt
(.CHALIMA_VENV) chalimasadiah@LAPTOP-CH00NPOB:~/CHALIMA_MLPROCESS$ pip install -r requirements.txt
Collecting pandas (from -r requirements.txt (line 1))
  Using cached pandas-3.0.0-cp312-cp312-manylinux_2_24_x86_64.manylinux_2_28_x86_64.whl.metadata (79 kB)
Collecting scikit-learn (from -r requirements.txt (line 2))
  Using cached scikit_learn-1.8.0-cp312-cp312-manylinux_2_27_x86_64.manylinux_2_28_x86_64.whl.metadata (11 kB)
Collecting imbalanced-learn (from -r requirements.txt (line 3))
  Downloading imbalanced_learn-0.14.1-py3-none-any.whl.metadata (8.9 kB)
Collecting joblib (from -r requirements.txt (line 4))
  Using cached joblib-1.5.3-py3-none-any.whl.metadata (5.5 kB)
Collecting numpy (from -r requirements.txt (line 5))
  Using cached numpy-2.4.1-cp312-cp312-manylinux_2_27_x86_64.manylinux_2_28_x86_64.whl.metadata (6.6 kB)
Collecting scipy (from -r requirements.txt (line 6))
  Using cached scipy-1.17.0-cp312-cp312-manylinux_2_27_x86_64.manylinux_2_28_x86_64.whl.metadata (62 kB)
Collecting seaborn (from -r requirements.txt (line 7))
  Downloading seaborn-0.13.2-py3-none-any.whl.metadata (5.4 kB)
Collecting fastapi (from -r requirements.txt (line 8))
  Downloading fastapi-0.128.0-py3-none-any.whl.metadata (30 kB)
Collecting jupyterlab (from -r requirements.txt (line 9))
  Using cached jupyterlab-4.5.3-py3-none-any.whl.metadata (16 kB)
Collecting requests (from -r requirements.txt (line 10))
  Using cached requests-2.32.1-py3-none-any.whl.metadata (4.9 kB)
```

e. [5 points] Non aktifkan venv Anda

- **Lampirkan screenshot terminal yang menampilkan command saat Anda menonaktifkan venv**

```
(.CHALIMA_VENV) chalimasadiah@LAPTOP-CH00NPOB:~/CHALIMA_MLPROCESS$ deactivate
chalimasadiah@LAPTOP-CH00NPOB:~/CHALIMA_MLPROCESS$
```

2. [25 points] Rangkum permasalahan bisnis

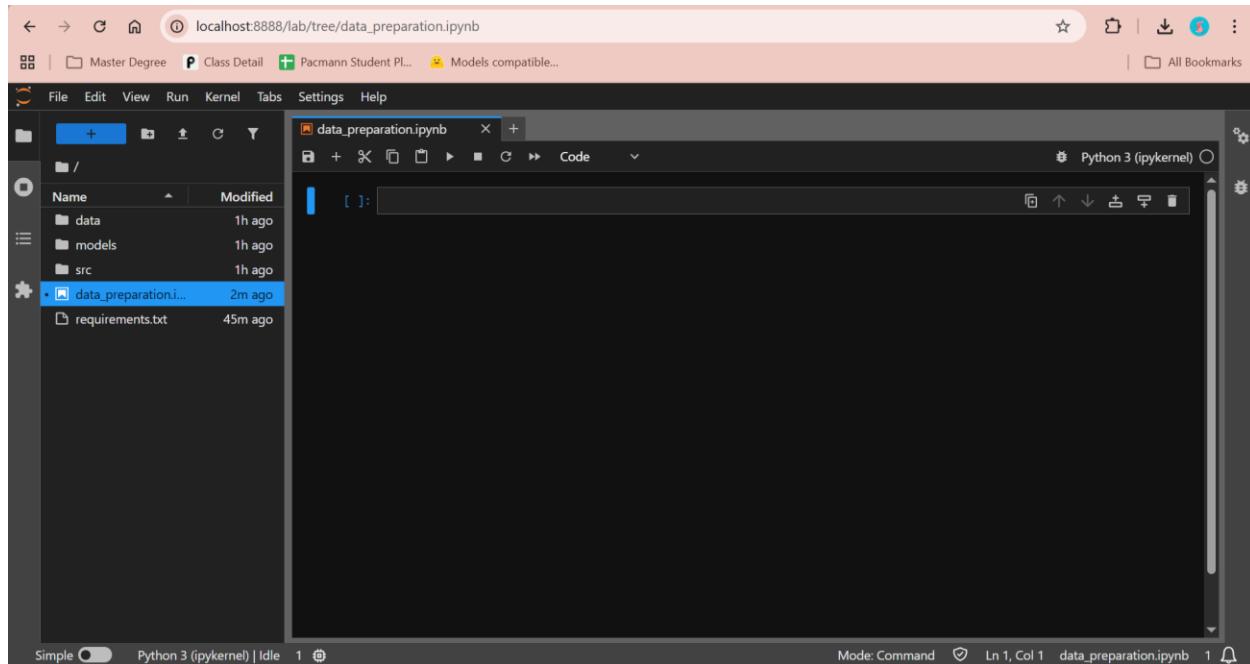
- Rangkum mockup interview antara user dan DS di bagian akhir dari dokumen ini
- **Rangkuman Anda harus bisa menjawab pertanyaan berikut:**

PERTANYAAN	JAWABAN
Latar belakang permasalahan bisnis	<i>Produk pinjaman (loan) merupakan salah satu sumber pendapatan utama bank. Namun, bank menghadapi risiko kerugian ketika nasabah gagal membayar pinjaman atau mengalami Non-Performing Loan (NPL). Saat ini, bank membutuhkan sistem yang mampu</i>

	<i>mendeteksi potensi gagal bayar lebih awal agar dapat mengambil tindakan preventif sebelum pinjaman benar-benar bermasalah.</i>
Objektif bisnis yang ingin dicapai	<i>Membangun sistem prediksi risiko kredit yang dapat membantu bank mengidentifikasi nasabah dengan potensi gagal bayar sejak dini, sehingga bank dapat menurunkan jumlah NPL dan meminimalkan potensi kerugian.</i>
Metrik pengukuran bisnis yang akan digunakan	<i>Keberhasilan solusi akan diukur berdasarkan penurunan jumlah atau persentase Non-Performing Loan (NPL) setelah sistem prediksi diterapkan dalam proses pemberian dan monitoring kredit.</i>
Kandidat solusi machine learning yang akan dibangun	<i>Solusi yang akan dibangun adalah model machine learning berbasis klasifikasi untuk memprediksi apakah seorang nasabah berpotensi mengalami gagal bayar atau tidak. Beberapa algoritma klasifikasi dapat dieksplorasi dan dibandingkan performanya untuk mendapatkan model terbaik sesuai kebutuhan bisnis.</i>
Metrik pengukuran machine learning yang akan digunakan	<i>Metrik evaluasi utama yang digunakan adalah <b>Recall</b>, khususnya untuk kelas gagal bayar (NPL). Hal ini dikarenakan bank lebih memprioritaskan untuk meminimalkan nasabah berisiko yang tidak terdeteksi oleh sistem. False positive masih dapat ditoleransi karena nasabah yang terdeteksi berisiko akan melalui proses verifikasi lanjutan.</i>

3. [25 points] Persiapan Dataset

- a. [1 points] Buat satu file bernama data\_preparation.ipynb dan letakkan pada root folder project Anda



b. [1 points] Letakkan file CSV dataset yang bernama credit\_risk\_dataset.csv ke folder raw

Sertakan screenshot hasil dari pemindahan file tersebut.

```
(.CHALIMA_VENV) chalimasadiah@LAPTOP-CH00NPOB:~/CHALIMA_MLPROCESS$ mv /mnt/c/Users/ASUS/Downloads/credit_risk_dataset.csv data/raw/
(.CHALIMA_VENV) chalimasadiah@LAPTOP-CH00NPOB:~/CHALIMA_MLPROCESS$
```

c. Folder raw terletak di dalam folder data

```
(.CHALIMA_VENV) chalimasadiah@LAPTOP-CH00NPOB:~/CHALIMA_MLPROCESS$ mv /mnt/c/Users/ASUS/Downloads/credit_risk_dataset.csv data/raw/
(.CHALIMA_VENV) chalimasadiah@LAPTOP-CH00NPOB:~/CHALIMA_MLPROCESS$ tree
.
├── data
│   ├── interim
│   ├── processed
│   └── raw
│       └── credit_risk_dataset.csv
└── data_preparation.ipynb

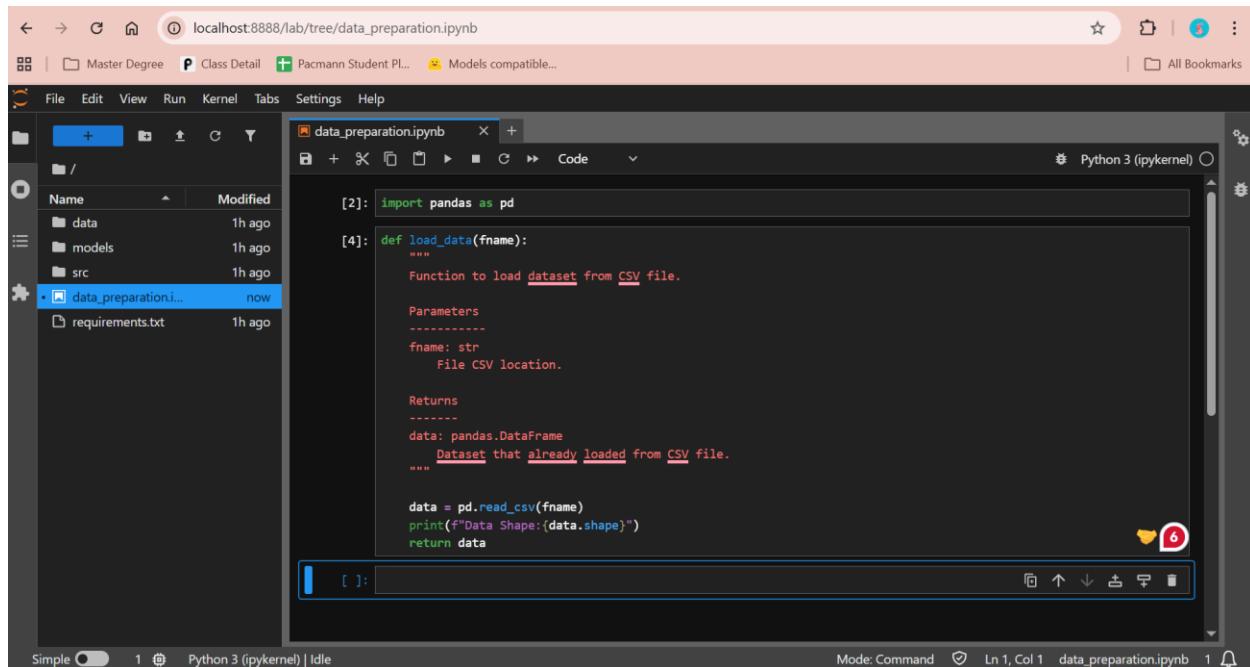
6 directories, 3 files
(.CHALIMA_VENV) chalimasadiah@LAPTOP-CH00NPOB:~/CHALIMA_MLPROCESS$
```

d. Muat data ke notebook

i.[3 points] Buat fungsi untuk memuat dataset

- Import library pandas sebelum membuat fungsi ini
- Buat fungsi dengan nama load\_data
- Fungsi load\_data memiliki parameter bernama fname
- Parameter fname adalah lokasi dimana file CSV dataset berada

- Parameter fname bertipe data string
- Muat data CSV dengan menggunakan fungsi `read_csv` dari pandas dan simpan dalam variabel bernama data
- Print shape dari data dengan keluaran "Data Shape: [SHAPE\_DATAFRAME]"
- Return data dari fungsi `load_data`
- Buat docstring untuk fungsi tersebut
- Docstring harus berisi:
  - Penjelasan tentang fungsi tersebut
  - Daftar nama parameter beserta tipe data dan penjelasannya
  - Daftar nama return value beserta tipe data dan penjelasannya



The screenshot shows a Jupyter Notebook interface with the following details:

- URL:** localhost:8888/lab/tree/data\_preparation.ipynb
- File Explorer:** Shows a directory structure with 'data', 'models', 'src', and 'data\_preparation.ipynb' (selected).
- Code Cell (Cell 4):**

```
import pandas as pd
def load_data(fname):
    """
    Function to load dataset from CSV file.

    Parameters
    -----
    fname: str
        File CSV location.

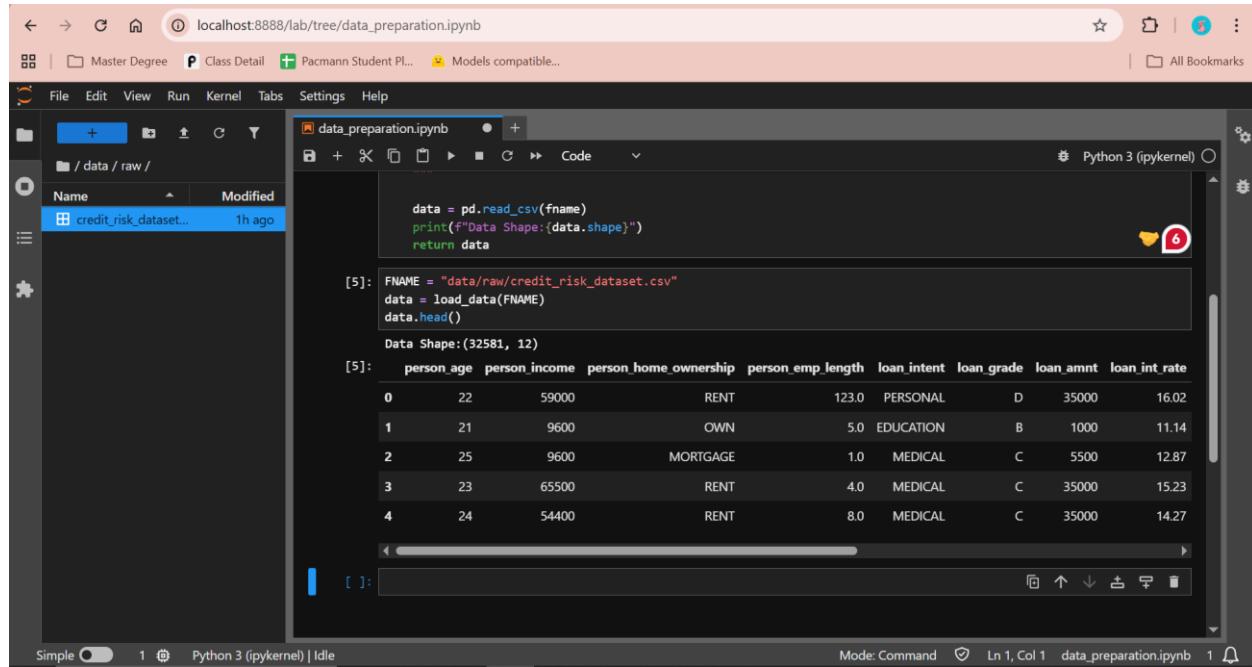
    Returns
    -----
    data: pandas.DataFrame
        Dataset that already loaded from CSV file.
    """
    data = pd.read_csv(fname)
    print(f"Data Shape:{data.shape}")
    return data
```
- Output Cell (Cell 5):** An empty cell with a progress bar.
- Status Bar:** Mode: Command, Ln 1, Col 1, data\_preparation.ipynb, 1

ii. [3 points] Jalankan fungsi

- Buat constant variabel bernama `FNAME`
- Assign lokasi file dataset ke `FNAME`
- Panggil fungsi `load_data`

- Passing FNAME sebagai argumen pada fungsi load\_data
- Simpan hasil fungsi load\_data ke variabel bernama data
- Cek data yang dimuat dengan menggunakan fungsi head()

### Sertakan screenshot hasil dari fungsi yang dijalankan



The screenshot shows a Jupyter Notebook interface with the following details:

- Title Bar:** localhost:8888/lab/tree/data\_preparation.ipynb
- File Explorer:** Shows a folder named "data / raw /" containing a file named "credit\_risk\_dataset.csv" modified 1h ago.
- Code Cell:** Contains Python code to read a CSV file and print its shape.
 

```
data = pd.read_csv(fname)
print(f'Data Shape:{data.shape}')
return data
```
- Output Cell:** Shows the result of the code execution, which is the shape of the dataset and a preview of the first 5 rows.
 

```
[5]: FNAME = "data/raw/credit_risk_dataset.csv"
data = load_data(FNAME)
data.head()
Data Shape:(32581, 12)
```

	person_age	person_income	person_home_ownership	person_emp_length	loan_intent	loan_grade	loan_amnt	loan_int_rate
0	22	59000	RENT	123.0	PERSONAL	D	35000	16.02
1	21	9600	OWN	5.0	EDUCATION	B	1000	11.14
2	25	9600	MORTGAGE	1.0	MEDICAL	C	5500	12.87
3	23	65500	RENT	4.0	MEDICAL	C	35000	15.23
4	24	54400	RENT	8.0	MEDICAL	C	35000	14.27

### e. Split input dan output dataset

i.[3 points] Buat fungsi untuk split input dan output dataset

- Buat fungsi dengan nama split\_input\_output
- Fungsi memiliki parameter bernama data dan target\_col
- Parameter data adalah dataset utuh bertipe data DataFrame
- Parameter target\_col adalah nama kolom yang berisi data target, parameter ini bertipe data string
- Buang kolom bernama target\_col dan simpan dalam variabel bernama X (upper case)
- Pilih kolom bernama target\_col dan simpan dalam variabel bernama y (lower case)
- Print shape dari data dengan keluaran:

"Original data shape: [SHAPE\_DATAFRAME]"

- Print shape dari X dengan keluaran:  
"X data shape: [SHAPE\_DATAFRAME]"
- Print shape dari y dengan keluaran:  
"y data shape: [SHAPE\_DATAFRAME]"
- Kembalikan X dan y
- Buat docstring untuk fungsi tersebut
- Docstring harus berisi:
  - Penjelasan tentang fungsi tersebut
  - Daftar nama parameter beserta tipe data dan penjelasannya
  - Daftar nama return value beserta tipe data dan penjelasannya

The screenshot shows a Jupyter Notebook interface with the following details:

- Header:** localhost:8888/lab/tree/data\_preparation.ipynb
- File Explorer:** Shows a directory structure with files: data, models, src, data\_preparation.ipynb (selected), and requirements.txt.
- Code Cell:** Contains the following Python code:

```
def split_input_output(data, target_col):
    """
    Function to split dataset to be input (X) and output (y).

    Parameters
    -----
    data: pandas.DataFrame
        Whole dataset that contain feature and target.

    target_col: str
        Target column name on dataset.

    Returns
    -----
    X: pandas.DataFrame
        Input dataset that contains feature.
    y: pandas.Series
        Output dataset that contains target.

    """
    X = data.drop(columns=[target_col])
    y = data[target_col]

    print(f"Original data shape: {data.shape}")
    print(f"X data shape: {X.shape}")
    print(f"y data shape: {y.shape}")

    return X, y
```

The code cell has a docstring explaining the purpose of the function, its parameters (data and target\_col), and its return values (X and y). It also includes print statements to show the original data shape and the shapes of the resulting X and y datasets.

ii. [3 points] Jalankan fungsi

- Siapkan constant variabel bernama TARGET\_COL
- Assign nama kolom yang menjadi target pada fungsi TARGET\_COL
- Panggil fungsi split\_input\_output

- Passing dataset utuh dan TARGET\_COL sebagai argumen pada fungsi split\_input\_output
- Simpan kembalian dari fungsi split\_input\_output ke variabel bernama X dan y

### Sertakan screenshot hasil dari fungsi yang dijalankan

```

y = data[target_col]

print(f"Original data shape: {data.shape}")
print(f"X data shape: {X.shape}")
print(f"y data shape: {y.shape}")

return X, y

[19]: TARGET_COL = "loan_status"

[21]: X, y = split_input_output(data, TARGET_COL)

Original data shape: (32581, 12)
X data shape: (32581, 11)
y data shape: (32581,)

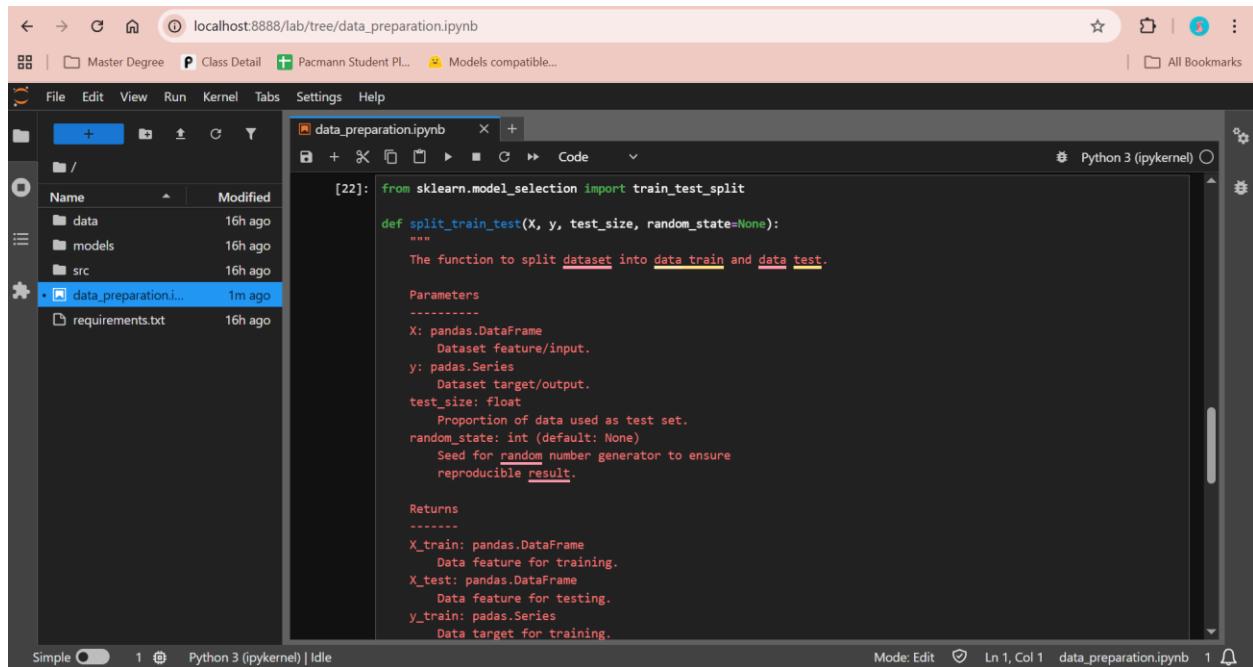
```

### f. Split train dan test dataset

#### i.[3 points] Buat fungsi untuk split train dan test dataset

- Import fungsi train\_test\_split dari sklearn.model\_selection sebelum membuat fungsi
- Buat fungsi dengan nama split\_train\_test
- Fungsi memiliki parameter bernama X, y, test\_size dan random\_state
- Parameter random\_state memiliki default value None
- Gunakan fungsi train\_test\_split untuk memisahkan X dan y menjadi train set dan test set
- Passing nilai test\_size dan random\_state pada parameter fungsi ke fungsi train\_test\_split
- Simpan kembalian dari fungsi train\_test\_split ke variabel X\_train, X\_test, y\_train, dan y\_train
- Gunakan stratify ada fungsi train\_test\_split
- Print shape dari X\_train, X\_test, y\_train, dan y\_train dengan keluaran:

- “X train shape: [SHAPE\_X\_TRAIN]” untuk X\_train
- “X test shape: [SHAPE\_X\_TEST]” untuk X\_test
- “y train shape: [SHAPE\_Y\_TRAIN]” untuk y\_train
- “y test shape: [SHAPE\_Y\_TEST]” untuk y\_test
- Kembalikan X\_train, X\_test, y\_train, dan y\_train
- Buat docstring untuk fungsi tersebut
- Docstring harus berisi:
  - Penjelasan tentang fungsi tersebut
  - Daftar nama parameter beserta tipe data dan penjelasannya
  - Daftar nama return value beserta tipe data dan penjelasannya



The screenshot shows a Jupyter Notebook interface with a dark theme. On the left, there's a file tree showing a directory structure with 'data', 'models', 'src', and 'requirements.txt'. The main area displays a code cell (cell 22) containing Python code:

```

from sklearn.model_selection import train_test_split

def split_train_test(X, y, test_size, random_state=None):
    """
    The function to split dataset into data_train and data_test.

    Parameters
    -----
    X: pandas.DataFrame
        Dataset feature/input.
    y: pandas.Series
        Dataset target/output.
    test_size: float
        Proportion of data used as test set.
    random_state: int (default: None)
        Seed for random number generator to ensure
        reproducible result.

    Returns
    -----
    X_train: pandas.DataFrame
        Data feature for training.
    X_test: pandas.DataFrame
        Data feature for testing.
    y_train: pandas.Series
        Data target for training.
    """

```

The code cell is associated with 'Python 3 (ipykernel)'.

The screenshot shows a Jupyter Notebook interface with the following details:

- URL:** localhost:8888/lab/tree/data\_preparation.ipynb
- File Explorer:** Shows a directory structure with 'data', 'models', 'src', and 'requirements.txt' files.
- Code Editor:** Displays Python code for data splitting:

```
X_test: pandas.DataFrame
    Data feature for testing.
y_train: pandas.Series
    Data target for training.
y_test: pandas.Series
    Data target for testing.

X_train, X_test, y_train, y_test = train_test_split (
    X,
    y,
    test_size=test_size,
    random_state=random_state,
    stratify=y
)

print(f"X train shape: {X_train.shape}")
print(f"X test shape: {X_test.shape}")
print(f"y train shape: {y_train.shape}")
print(f"y test shape: {y_train.shape}")

return X_train, X_test, y_train, y_test
```
- Kernel:** Python 3 (ipykernel)
- Status Bar:** Mode: Command, Ln 1, Col 1, data\_preparation.ipynb, 1

ii. [3 points] Jalankan fungsi

- Panggil fungsi `split_train_test`
- Passing `X` dan `y` ke fungsi `split_train_test`
- Isi argumen `test_size = 0.2` dan `random_state = 42` ke fungsi `split_train_test`
- Simpan keluaran fungsi `split_train_test` ke variabel bernama `X_train`, `X_non_train`, `y_train`, dan `y_not_train`
- Panggil lagi fungsi `split_train_test`
- Passing `X_not_train` dan `y_not_train` ke fungsi `split_train_test`
- Isi argumen `test_size = 0.5` dan `random_state = 42` ke fungsi `split_train_test`
- Simpan keluaran fungsi `split_train_test` ke variabel bernama `X_valid`, `X_test`, `y_valid`, dan `y_test`

Sertakan screenshot hasil dari fungsi yang dijalankan

The screenshot shows a Jupyter Notebook interface with the following details:

- URL:** localhost:8888/lab/tree/data\_preparation.ipynb
- File Tree:** Shows a directory structure with 'data', 'models', 'src', and 'data\_preparation.ipynb' (selected).
- Code Cell 23:** Contains code for splitting data into training and test sets. It prints the shapes of X\_train, X\_test, y\_train, and y\_test.

```
[23]: X_train, X_non_train, y_train, y_non_train = split_train_test(  
    X,  
    y,  
    test_size=0.2,  
    random_state=42  
)  
X train shape: (26064, 11)  
X test shape: (6517, 11)  
y train shape: (26064,)  
y test shape: (26064,)
```
- Code Cell 24:** Contains code for splitting data into validation and test sets. It prints the shapes of X\_valid, X\_test, y\_valid, and y\_test.

```
[24]: X_valid, X_test, y_valid, y_test = split_train_test(  
    X_non_train,  
    y_non_train,  
    test_size=0.5,  
    random_state=42  
)  
X train shape: (3258, 11)  
X test shape: (3259, 11)  
y train shape: (3258,)  
y test shape: (3258,)
```
- Status Bar:** Shows 'Simple' mode, Python 3 (ipykernel) | Idle, Mode: Edit, Ln 1, Col 1, data\_preparation.ipynb, and a line number 1.

#### g. Serialize data

i.[2 points] Buat fungsi untuk melakukan serialization

- Import library joblib sebelum membuat fungsi
- Buat fungsi dengan nama serialize\_data
- Fungsi memiliki parameter bernama data dan path
  - Parameter data adalah instance yang ingin diserialisasi
  - Parameter path adalah alamat dimana data tersebut ingin diserialisasi
- Gunakan fungsi dump() dari library joblib untuk melakukan serialisasi
- Passing parameter data dan path ke fungsi dump()
- Buat docstring untuk fungsi tersebut
- Docstring harus berisi:
  - Penjelasan tentang fungsi tersebut
  - Daftar nama parameter beserta tipe data dan penjelasannya

- Daftar nama return value beserta tipe data dan penjelasannya

```

localhost:8888/lab/tree/data_preparation.ipynb
File Edit View Run Kernel Tabs Settings Help
data_preparation.ipynb Python 3 (ipykernel)
Name Modified
data 16 ago
models 16 ago
src 16 ago
data_preparation.ipynb now
requirements.txt 16 ago
[26]: import joblib
def serialize_data(data, path):
    """
    The function to save serialize (object) Python into file.

    Parameters
    -----
    data: object
        The Python object to be serialized (e.g. DataFrame, Series, array).
    path: str
        The location or file name where the object will be saved.

    Returns
    -----
    None
        This function does not return any value.
    """
    joblib.dump(data, path)
X train shape: (3258, 11)
X test shape: (3259, 11)
y train shape: (3258,)
y test shape: (3258,)

Mode: Edit Ln 20, Col 28 data_preparation.ipynb 1

```

ii. [2 points] Jalankan fungsi

- Panggil fungsi serialize\_data()
  - Untuk serialize data X\_train, passing X\_train untuk parameter data dan “X\_train.pkl” untuk parameter path
  - Untuk serialize data y\_train, passing y\_train untuk parameter data dan “y\_train.pkl” untuk parameter path
  - Untuk serialize data x\_test, passing x\_test untuk parameter data dan “x\_test.pkl” untuk parameter path
  - Untuk serialize data y\_test, passing y\_test untuk parameter data dan “y\_test.pkl” untuk parameter path
  - Untuk serialize data x\_valid, passing x\_valid untuk parameter data dan “x\_valid.pkl” untuk parameter path
  - Untuk serialize data y\_valid, passing y\_valid untuk parameter data dan “y\_valid.pkl” untuk parameter path

**Sertakan screenshot hasil dari fungsi yang dijalankan**

The screenshot shows a Jupyter Notebook interface with the following details:

- Header:** localhost:8888/lab/tree/data\_preparation.ipynb
- File Bar:** File, Edit, View, Run, Kernel, Tabs, Settings, Help
- Left Sidebar:** A file tree showing the directory structure:
  - /
  - Name Modified
  - data 16h ago
  - models 16h ago
  - src 16h ago
  - data\_preparation.ipynb now (selected)
  - requirements.txt 16h ago
  - X\_test.pkl 7s ago
  - X\_train.pkl 7s ago
  - X\_valid.pkl 7s ago
  - y\_test.pkl 7s ago
  - y\_train.pkl 7s ago
  - y\_valid.pkl 7s ago
- Code Cell:** [27]:

```
"""
joblib.dump(data, path)

[27]: serialize_data(X_train, "X_train.pkl")
serialize_data(y_train, "y_train.pkl")

serialize_data(X_test, "X_test.pkl")
serialize_data(y_test, "y_test.pkl")

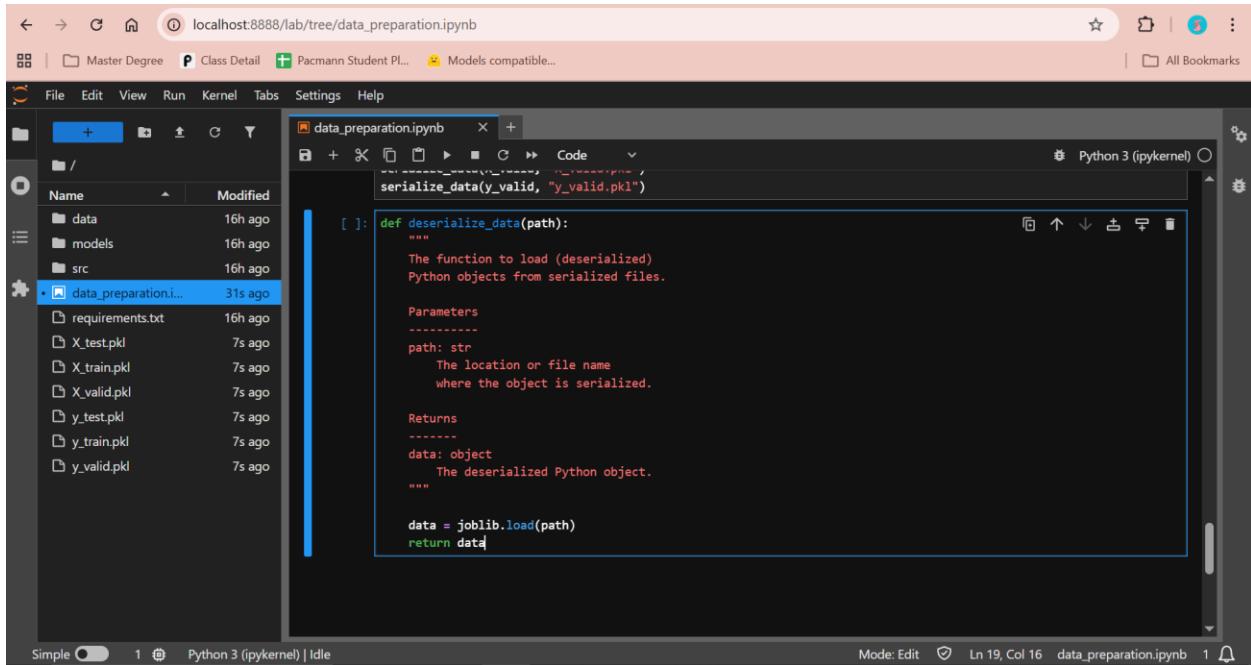
serialize_data(X_valid, "X_valid.pkl")
serialize_data(y_valid, "y_valid.pkl")
```
- Kernel:** Python 3 (ipykernel)
- Status Bar:** Simple 1 Python 3 (ipykernel) | Idle Mode: Edit Ln 1, Col 1 data\_preparation.ipynb 1

## h. Deserialize data

i.[1 points] Buat fungsi untuk melakukan deserialization

- Import library joblib sebelum membuat fungsi jika belum (dapat dijadikan satu dengan fungsi serialize\_data())
- Buat fungsi dengan nama deserialize\_data
- Fungsi memiliki parameter bernama path
  - Parameter path adalah alamat dimana data tersebut ingin dideserialisasi
- Gunakan fungsi load() dari library joblib untuk melakukan serialisasi
- Passing parameter path ke fungsi dump()
- Simpan kembalian dari fungsi load() ke variabel bernama data
- Kembalikan data
- Buat docstring untuk fungsi tersebut
- Docstring harus berisi:
  - Penjelasan tentang fungsi tersebut

- Daftar nama parameter beserta tipe data dan penjelasannya
- Daftar nama return value beserta tipe data dan penjelasannya



The screenshot shows a Jupyter Notebook interface with the following details:

- URL:** localhost:8888/lab/tree/data\_preparation.ipynb
- File Explorer:** Shows a directory structure with folders data, models, and src, and files requirements.txt, X\_test.pkl, X\_train.pkl, X\_valid.pkl, y\_test.pkl, y\_train.pkl, and y\_valid.pkl.
- Code Cell:** Displays Python code for serializing and deserializing data using joblib.

```
def serialize_data(path):
    """
    The function to save serialized Python objects to files.
    Parameters
    -----
    path: str
        The location or file name where the object is serialized.
    Returns
    -----
    data: object
        The serialized Python object.
    """
    joblib.dump(data, path)
    return data

def deserialize_data(path):
    """
    The function to load (deserialized) Python objects from serialized files.
    Parameters
    -----
    path: str
        The location or file name where the object is serialized.
    Returns
    -----
    data: object
        The deserialized Python object.
    """
    data = joblib.load(path)
    return data
```

- Status Bar:** Shows "Python 3 (ipykernel) | Idle".
- Bottom Right:** Shows "Mode: Edit" and "Ln 19, Col 16" along with a bell icon.

4. [25 points] Buat python utility script
  - a. [5 points] Buat satu file baru bernama utils.py pada folder src

```
[...]
(.CHALIMA_VENV) chalimasadiyah@LAPTOP-CH00NPOB:~/CHALIMA_MLPROCESS$ touch src/utils.py
(.CHALIMA_VENV) chalimasadiyah@LAPTOP-CH00NPOB:~/CHALIMA_MLPROCESS$ tree
.
├── X_test.pkl
├── X_train.pkl
├── X_valid.pkl
├── data
│   ├── interim
│   ├── processed
│   └── raw
│       └── credit_risk_dataset.csv
├── data_preparation.ipynb
└── models
    └── requirements.txt
└── src
    └── utils.py
y_test.pkl
y_train.pkl
y_valid.pkl

6 directories, 10 files
(.CHALIMA_VENV) chalimasadiyah@LAPTOP-CH00NPOB:~/CHALIMA_MLPROCESS$
```

b. [5 points] Copy dan paste ketiga library yang digunakan pada data\_preparation.ipynb ke utils.py

- i.Fungsi DataFrame dari library pandas
- ii.Fungsi train\_test\_split dari library sklearn.model\_selection
- iii.Library joblib

```
chalimasadiyah@LAPTOP-CH00NPOB: ~/CHALIMA_MLPROCESS
GNU nano 4.8
import pandas as pd
from sklearn.model_selection import train_test_split
import joblib
```

c. [5 points] Copy dan paste ke lima fungsi yang telah dibuat di data\_preparation.ipynb ke utils.py

i.load\_data()

```
chalmasadiah@LAPTOP-CH00NPOB: ~/CHALIMA_MLPROCESS
GNU nano 4.8                                         src/utils.py

import pandas as pd
from sklearn.model_selection import train_test_split
import joblib

def load_data(fname):
    """
    The function to load dataset from CSV file.

    Parameters
    -----
    fname: str
        File CSV location.

    Returns
    -----
    data: pandas.DataFrame
        Dataset that already loaded from CSV file.
    """

    data = pd.read_csv(fname)
    print(f'Data Shape:{data.shape}')
    return data

def split_input_output(data, target_col):
    """
    The function to split dataset into input (X) and output (y).

    Parameters
    -----
    data: pandas.DataFrame
        Whole dataset that contain feature and target.

    target_col: str
        Target column name on dataset.

    Returns
    -----
    X: pandas.DataFrame
        Input dataset that contains feature.

    Y: pandas.Series
        Output dataset that contains target.
    """

    X = data.drop(columns=[target_col])
    y = data[target_col]

    print(f'Original data shape: {data.shape}')
    print(f'X data shape: {X.shape}')
    print(f'y data shape: {y.shape}')

    return X, y
```

## ii.split\_input\_output()

```
chalmasadiah@LAPTOP-CH00NPOB: ~/CHALIMA_MLPROCESS
GNU nano 4.8                                         src/utils.py

def split_input_output(data, target_col):
    """
    The function to split dataset into input (X) and output (y).

    Parameters
    -----
    data: pandas.DataFrame
        Whole dataset that contain feature and target.

    target_col: str
        Target column name on dataset.

    Returns
    -----
    X: pandas.DataFrame
        Input dataset that contains feature.

    Y: pandas.Series
        Output dataset that contains target.
    """

    X = data.drop(columns=[target_col])
    y = data[target_col]

    print(f'Original data shape: {data.shape}')
    print(f'X data shape: {X.shape}')
    print(f'y data shape: {y.shape}')

    return X, y

def split_train_test(X, y, test_size, random_state=None):
    """
    The function to split dataset into data train and data test.

    Parameters
    -----
    X: pandas.DataFrame
        Dataset feature/input.

    Y: pandas.Series
    """

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test_size, random_state=random_state)

    return X_train, X_test, y_train, y_test
```

## iii.split\_train\_test()

```
chalmasadiah@LAPTOP-CH00NPOB: ~/CHALIMA_MLPROCESS
GNU nano 4.8                                         src/utils.py

def split_train_test(X, y, test_size, random_state=None):
    """
    The function to split dataset into data train and data test.

    Parameters
    -----
    X: pandas.DataFrame
        Dataset feature/input.
    y: pandas.Series
        Dataset target/output.
    test_size: float
        Proportion of data used as test set.
    random_state: int (default: None)
        Seed for random number generator to ensure
        reproducible result.

    Returns
    -----
    X_train: pandas.DataFrame
        Data feature for training.
    X_test: pandas.DataFrame
        Data feature for testing.
    y_train: pandas.Series
        Data target for training.
    y_test: pandas.Series
        Data target for testing.
    """

    X_train, X_test, y_train, y_test = train_test_split (
        X,
        y,
        test_size=test_size,
        random_state=random_state,
        stratify=y
    )

    print(f"X train shape: {X_train.shape}")
    print(f"X test shape: {X_test.shape}")

Get Help   ^C Write Out   ^W Where Is   ^K Cut Text   ^J Justify   ^C Cur Pos   M-U Undo   M-A Mark Text   M-T To Bracket   M-Q Previous
Exit      ^R Read File   ^R Replace   ^U Paste Text  ^T To Spell   ^G Go To Line  M-E Redo   M-B Copy Text   M-Q Where Was  M-W Next
```

## iv.serialize\_data()

```
chalmasadiah@LAPTOP-CH00NPOB: ~/CHALIMA_MLPROCESS
GNU nano 4.8                                         src/utils.py

def serialize_data(data, path):
    """
    The function to save serialize (object) Python into file.

    Parameters
    -----
    data: object
        The Python object to be serialized (e.g. DataFrame, Series, array).
    path: str
        The location or file name where the object will be saved.

    Returns
    -----
    None
        This function does not return any value.
    """

    joblib.dump(data, path)

def deserialize_data(path):
    """
    The function to load (deserialized)
    Python objects from serialized files.

    Parameters
    -----
    path: str
        The location or file name
        where the object is serialized.

    Returns
    -----
    data: object
        The deserialized Python object.
    """

    data = joblib.load(path)
    return data

Get Help   ^C Write Out   ^W Where Is   ^K Cut Text   ^J Justify   ^C Cur Pos   M-U Undo   M-A Mark Text   M-T To Bracket   M-Q Previous
Exit      ^R Read File   ^R Replace   ^U Paste Text  ^T To Spell   ^G Go To Line  M-E Redo   M-B Copy Text   M-Q Where Was  M-W Next
```

## v.deserialize\_data()

```
chalmasadiah@LAPTOP-CH00NPOB: ~/CHALIMA_MLPROCESS
GNU nano 4.8                                         src/utils.py

def serialize_data(data, path):
    """
    The function to save serialize (object) Python into file.

    Parameters
    -----
    data: object
        The Python object to be serialized (e.g. DataFrame, Series, array).
    path: str
        The location or file name where the object will be saved.

    Returns
    -----
    None
        This function does not return any value.
    """

    joblib.dump(data, path)

def deserialize_data(path):
    """
    The function to load (deserialized)
    Python objects from serialized files.

    Parameters
    -----
    path: str
        The location or file name
        where the object is serialized.

    Returns
    -----
    data: object
        The deserialized Python object.
    """

    data = joblib.load(path)
    return data

Get Help   ^C Write Out   ^W Where Is   ^K Cut Text   ^A Justify   ^C Cur Pos   M-U Undo
Exit      ^R Read File   ^R Replace   ^U Paste Text  ^T To Spell   A Go To Line  M-E Redo
                                                M-A Mark Text  M-] To Bracket M-Q Previous
                                                M-G Copy Text  ^C Where Was   M-W Next
```

d. [10 points] Guna pengecekan apakah utils.py bisa digunakan, silahkan lakukan hal berikut:

- i. Buat satu file baru bernama notebook data\_preprocessing\_utils.ipynb di root folder project Anda
- ii. Lakukan ulang apa yang Anda lakukan di data\_preprocessing.ipynb namun dengan menggunakan utils.py dibanding Anda menggunakan fungsinya langsung

**Sertakan screenshot hasil dari fungsi yang dijalankan (sertakan screenshot terminal)**

localhost:8888/lab/tree/data\_preprocessing\_utils.ipynb

File Edit View Run Kernel Tabs Settings Help

[2]: from src.utils import (  
 load\_data,  
 split\_input\_output,  
 split\_train\_test,  
 serialize\_data,  
 deserialize\_data  
)

[3]: FNAME = "data/raw/credit\_risk\_dataset.csv"  
data = load\_data(FNAME)  
data.head()

Data Shape:(32581, 12)

	person_age	person_income	person_home_ownership	person_emp_length	loan_intent	loan_grade	loan_amnt	loan_int_rate
0	22	59000	RENT	123.0	PERSONAL	D	35000	16.02
1	21	9600	OWN	5.0	EDUCATION	B	1000	11.14
2	25	9600	MORTGAGE	1.0	MEDICAL	C	5500	12.87
3	23	65500	RENT	4.0	MEDICAL	C	35000	15.22
4	24	54400	RENT	8.0	MEDICAL	C	35000	14.22

Would you like to get notified about official Jupyter news?

Open privacy policy Yes No

Simple 2 Python 3 (ipykernel) | Idle Mode: Edit Ln 1, Col 1 data\_preprocessing\_utils.ipynb 1

localhost:8888/lab/tree/data\_preprocessing\_utils.ipynb

File Edit View Run Kernel Tabs Settings Help

[3]: FNAME = "data/raw/credit\_risk\_dataset.csv"  
data = load\_data(FNAME)  
data.head()

Data Shape:(32581, 12)

	person_age	person_income	person_home_ownership	person_emp_length	loan_intent	loan_grade	loan_amnt	loan_int_rate
0	22	59000	RENT	123.0	PERSONAL	D	35000	16.02
1	21	9600	OWN	5.0	EDUCATION	B	1000	11.14
2	25	9600	MORTGAGE	1.0	MEDICAL	C	5500	12.87
3	23	65500	RENT	4.0	MEDICAL	C	35000	15.22
4	24	54400	RENT	8.0	MEDICAL	C	35000	14.22

[5]: TARGET\_COL = "loan\_status"  
X, y = split\_input\_output(data, TARGET\_COL)

Original data shape: (32581, 12)  
X data shape: (32581, 11)  
y data shape: (32581,)

[6]: X\_train, X\_non\_train, y\_train, y\_non\_train = split\_train\_test(  
 X,

G

Simple 2 Python 3 (ipykernel) | Idle Mode: Command Ln 1, Col 1 data\_preprocessing\_utils.ipynb 1

localhost:8888/lab/tree/data\_preprocessing\_utils.ipynb

File Edit View Run Kernel Tabs Settings Help

data\_preparation.ipynb data\_preprocessing\_utils.ipynb

[6]: `X_train, X_non_train, y_train, y_non_train = split_train_test(  
 X,  
 y,  
 test_size=0.2,  
 random_state=42  
)`

X train shape: (26064, 11)  
X test shape: (6517, 11)  
y train shape: (26064,)  
y test shape: (6517,)

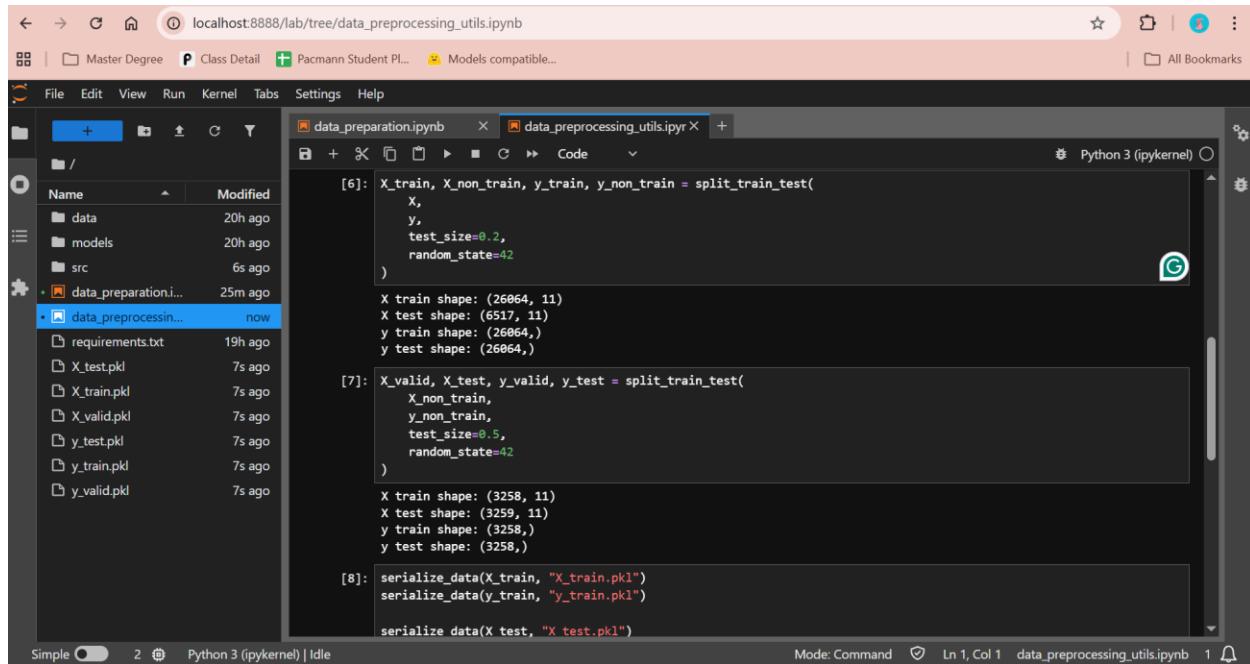
[7]: `X_valid, X_test, y_valid, y_test = split_train_test(  
 X_non_train,  
 y_non_train,  
 test_size=0.5,  
 random_state=42  
)`

X train shape: (3258, 11)  
X test shape: (3259, 11)  
y train shape: (3258,)  
y test shape: (3259,)

[8]: `serialize_data(X_train, "X_train.pkl")  
serialize_data(y_train, "y_train.pkl")  
  
serialize_data(X_test, "X_test.pkl")`

Mode: Command Ln 1, Col 1 data\_preprocessing\_utils.ipynb 1

Simple Python 3 (ipykernel) | Idle



localhost:8888/lab/tree/data\_preprocessing\_utils.ipynb

File Edit View Run Kernel Tabs Settings Help

data\_preparation.ipynb data\_preprocessing\_utils.ipynb

[6]: `X_train, X_non_train, y_train, y_non_train = split_train_test(  
 X,  
 y,  
 test_size=0.2,  
 random_state=42  
)`

X train shape: (3258, 11)  
X test shape: (3259, 11)  
y train shape: (3258,)  
y test shape: (3259,)

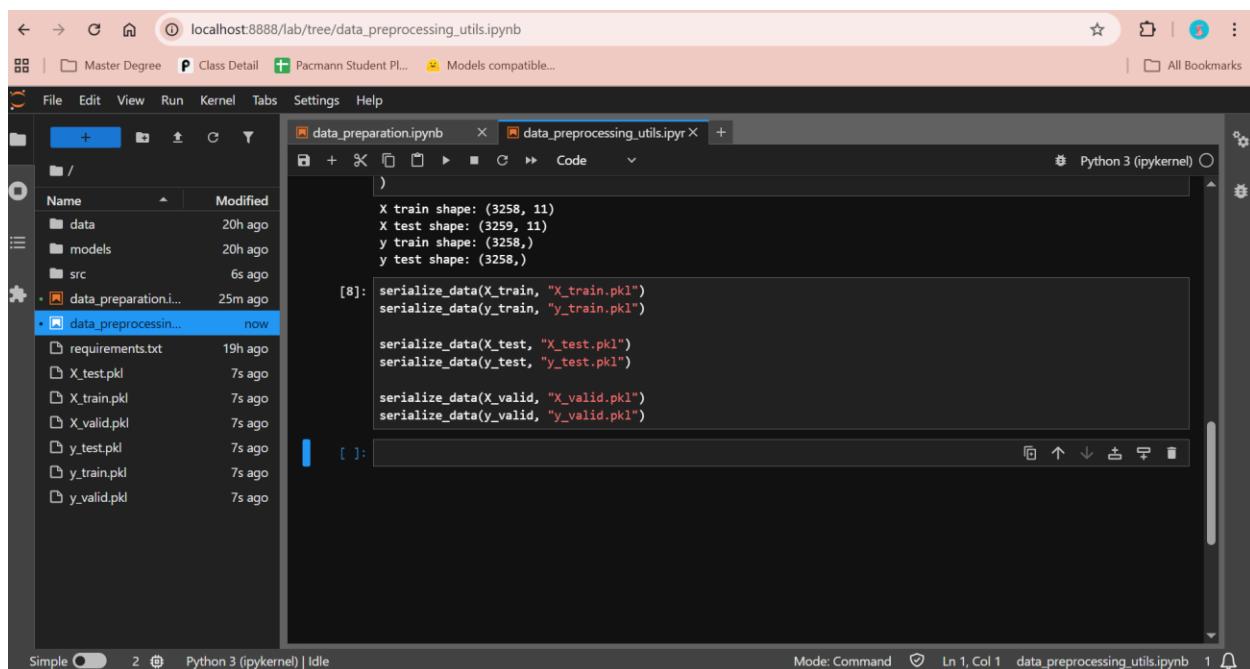
[7]: `X_valid, X_test, y_valid, y_test = split_train_test(  
 X_non_train,  
 y_non_train,  
 test_size=0.5,  
 random_state=42  
)`

X train shape: (3258, 11)  
X test shape: (3259, 11)  
y train shape: (3258,)  
y test shape: (3259,)

[8]: `serialize_data(X_train, "X_train.pkl")  
serialize_data(y_train, "y_train.pkl")  
  
serialize_data(X_test, "X_test.pkl")  
serialize_data(y_test, "y_test.pkl")  
  
serialize_data(X_valid, "X_valid.pkl")  
serialize_data(y_valid, "y_valid.pkl")`

Mode: Command Ln 1, Col 1 data\_preprocessing\_utils.ipynb 1

Simple Python 3 (ipykernel) | Idle



## Mockup Percakapan

Pengguna : Hai, Pak David! Terima kasih sudah menyempatkan waktu untuk bertemu.

David : Hai! Tentu saja, tidak masalah. Saya senang bisa membantu. Ada yang bisa saya bantu?

Pengguna Jadi begini, Pak David, kami di bank sedang memiliki masalah dengan risiko kredit, nih. Kami ingin bikin sistem prediksi yang bisa bantu kita lebih cepet nangkep pinjaman yang berpotensi jadi masalah, gitu.

David Ah, mengerti. Jadi tujuannya adalah untuk lebih akurat dalam memprediksi pinjaman bermasalah. Keren, tujuan bisnisnya apa nih?

Pengguna Ya, tentu aja. Kami pengen turunin jumlah NPL (Non-Performing Loan) dan lebih cepet deteksi pinjaman yang berisiko biar bisa langkah preventif lebih awal.

David Oke, paham. Jadi pengukuran keberhasilannya nanti gimana?

Pengguna Gampang, nih. Yang jelas pengen liat jumlah NPL yang beneran turun.

David Nah, kalo soal solusi machine learning, ada preferensi khusus gak?

Pengguna Hmm, nggak terlalu sih. Yang penting bisa handle data dalam jumlah besar dan kita bisa coba beberapa model yang beda-beda.

David Bagus. Terus kalo pengen evaluasi kinerjanya gimana?

Pengguna Ya, kita memperbolehkan adanya false alarm. Ini karena nantinya kita akan saring lagi yang diprediksi sebagai NPL, apakah emang benar akan NPL? Dan kita juga akan beri treatment agar meminimalisir nasabah yang akan NPL.

David Terakhir, ada insight apa aja dari eksplorasi data awalnya?

Pengguna      Udah ada beberapa pola menarik terkait perilaku pembayaran pelanggan dan faktor risiko lainnya. Mudah-mudahan nanti model bisa nangkep pola-pola itu dengan baik.

David      Keren, terima kasih udah sharing. Dengan info yang udah kamu kasih, saya bakal buat solusi yang sesuai dengan kebutuhan kamu.

Pengguna      Mantap, Pak David! Kami bener-bener berharap bisa kerja sama dengan Anda.

David      Sama-sama, semoga bisa ngasih hasil yang memuaskan. Kalo ada pertanyaan atau hal lain yang perlu dibahas, jangan ragu buat hubungi gue lagi ya.

Pengguna      Deal! Makasih banyak, Pak David. Sampai ketemu lagi!

David      Sama-sama! Sampai jumpa dan semoga proyeknya lancar ya!

Dataset

Dataset yang digunakan adalah Credit Risk Dataset.

Hyperlink:

<https://www.kaggle.com/datasets/laotse/credit-risk-dataset>

Mirror hyperlink:

<https://drive.google.com/file/d/1DrYNdjQkPiDxeZzUsaaloWNxWs71DEO0/view?usp=sharing>