

## Soal Mentoring Meet 2 Machine Learning Process

Job Preparation Program, Data Science, Pacmann AI

### Introduction

Kelanjutan dari Week 1, dengan menggunakan dataset yang sama, Anda diminta untuk melanjutkan EDA, preprocessing dan Feature Engineering.

#### Catatan:

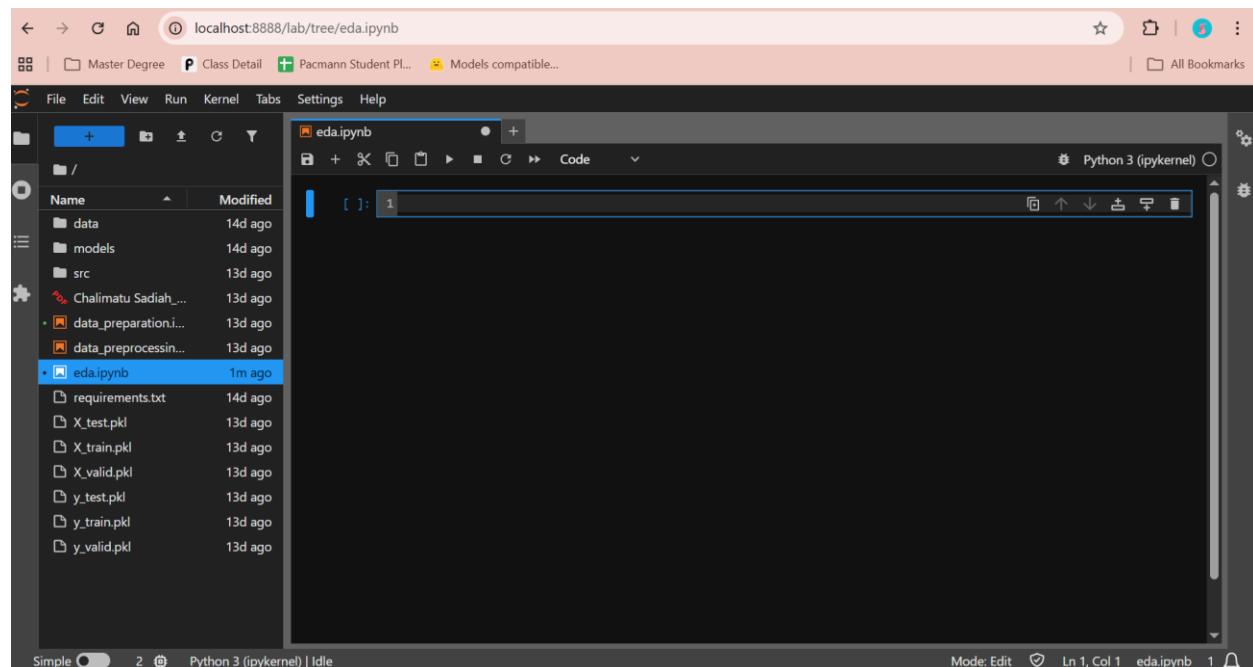
- Tulis jawaban Anda dalam jupyter notebook
- Copy file docs ini untuk menjawab dan sertakan screenshotnya
- Ekspor docs yang telah ada jawabannya ke PDF
- Archive file notebook dan file PDF Anda ke ZIP, **kecuali folder venv**
- Beri nama [NAMA LENGKAP]\_MLPROCESS\_2 file archive Anda

#### Soal

##### 1. [50 poin] EDA

- a. [1 poin] Buatlah satu file bernama eda.ipynb di root folder project Anda.

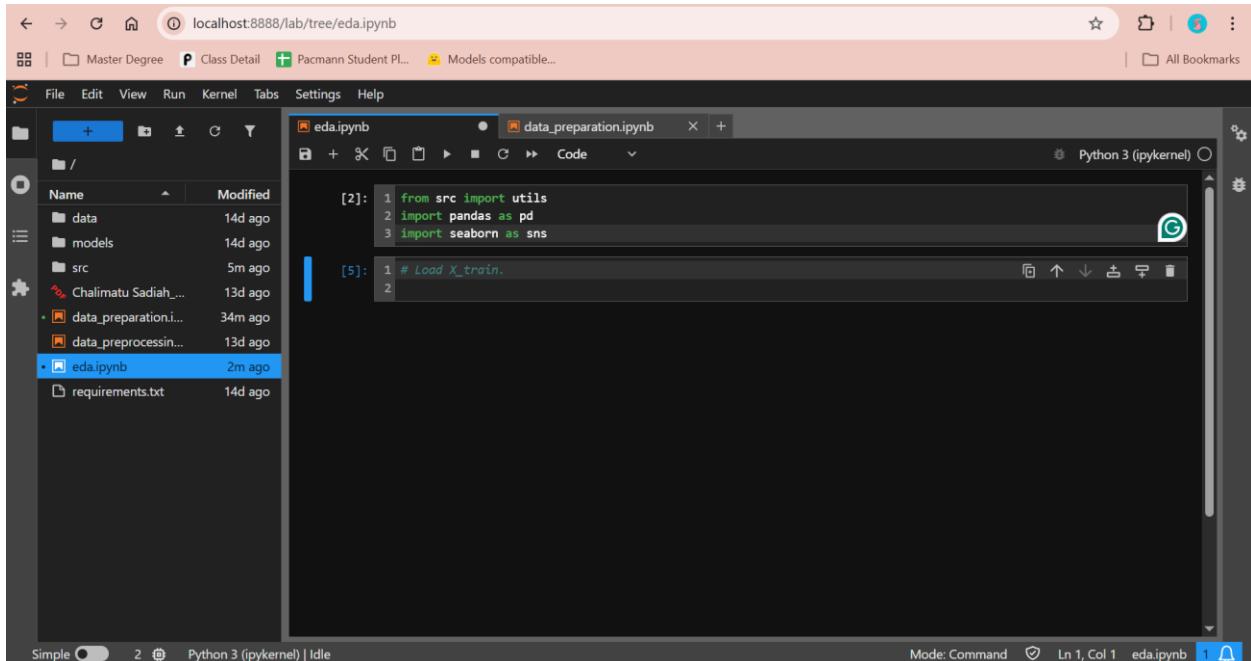
#### Lampirkan screenshot



b. [1 poin] Import library yang dibutuhkan

Buka eda.ipynb dan import library utils dari folder src, pandas yang memiliki alias pd dan seaborn yang memiliki alias sns

Lampirkan screenshot



The screenshot shows a Jupyter Notebook interface with two tabs open: 'eda.ipynb' and 'data\_preparation.ipynb'. The 'eda.ipynb' tab is active, displaying the following code in cell [2]:

```
from src import utils
import pandas as pd
import seaborn as sns
```

Cell [5] contains:

```
# Load X_train.
```

The left sidebar shows a file tree with the following structure:

- /
- Name Modified
- data 14d ago
- models 14d ago
- src 5m ago
- Chalimatu Sadiyah... 13d ago
- data\_preparation.i... 34m ago
- data\_preprocessin... 13d ago
- eda.ipynb 2m ago
- requirements.txt 14d ago

c. [1 poin] Muat train set data

i. Muat X\_train data

- Siapkan variabel konstan bernama X\_TRAIN\_PATH
- Isi X\_TRAIN\_PATH dengan “data/interim/X\_train.pkl”, TANPA PETIK!
- Panggil fungsi deserialize\_data() dan berikan X\_TRAIN\_PATH sebagai argumen pertama
- Simpan keluaran dari fungsi deserialize\_data() ke variabel bernama X\_train

Lampirkan screenshot

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

- Header:** localhost:8888/lab/tree/eda.ipynb
- File Explorer:** Shows a directory structure with files like data, models, src, Chalimatu Sadiyah..., data\_preparation.ipynb, data\_preprocessin..., eda.ipynb, and requirements.txt.
- Code Cells:**
  - [2]:

```
1 from src import utils
2 import pandas as pd
3 import seaborn as sns
```
  - [\*]:

```
1 # Load X_train
2 X_TRAIN_PATH = "data/interim/X_train.pkl"
3
4 X_train = deserialize_data(X_TRAIN_PATH)
```
- Kernel:** Python 3 (ipykernel)
- Status Bar:** Mode: Command, Ln 4, Col 41, eda.ipynb

ii. Muat y\_train data

- Siapkan variabel konstan bernama Y\_TRAIN\_PATH
- Isi Y\_TRAIN\_PATH dengan “data/interim/y\_train.pkl”, TANPA PETIK!
- Panggil fungsi deserialize\_data() dan berikan Y\_TRAIN\_PATH sebagai argumen pertama
- Simpan keluaran dari fungsi deserialize\_data() ke variabel bernama y\_train

Lampirkan screenshot

The screenshot shows a Jupyter Notebook interface with two code cells. The left sidebar lists files: 'data', 'models', 'src', 'Chalimatu Sadiah...', 'data\_preparation.ipynb' (selected), 'data\_preprocessing.ipynb', and 'requirements.txt'. The right pane has tabs for 'eda.ipynb' and 'data\_preparation.ipynb'. Cell [2] contains:

```
1 from src import utils
2 import pandas as pd
3 import seaborn as sns
```

Cell [\*]:

```
1 # Load X_train.
2 X_TRAIN_PATH = "data/interim/X_train.pkl"
3
4 X_train = deserialize_data(X_TRAIN_PATH)
```

Cell [\*]:

```
1 # Load y_train.
2 Y_TRAIN_PATH = "data/interim/y_train.pkl"
3
4 y_train = deserialize_data(Y_TRAIN_PATH)
```

d. [7 poin] Identifikasi kolom kategorik dan numerik

i. Panggil fungsi head() dari variabel X\_train

Lampirkan screenshot

The screenshot shows a Jupyter Notebook interface with code cells. The left sidebar lists files: 'data', 'models', 'src', 'Chalimatu Sadiah...', 'data\_preparation.ipynb' (selected), 'data\_preprocessing.ipynb', and 'requirements.txt'. The right pane has tabs for 'eda.ipynb' and 'data\_preparation.ipynb'. Cell [15] contains:

```
1 X_TRAIN_PATH = "data/interim/X_train.pkl"
2
3 X_train = u.deserialize_data(X_TRAIN_PATH)
```

Cell [16]:

```
1 # X_train head
2 X_train.head()
```

Cell [16] displays the output:

	person_age	person_income	person_home_ownership	person_emp_length	loan_intent	loan_grade	loan_amnt	loan_int
15884	25	241875	MORTGAGE	4.0	EDUCATION	A	16000	1
15138	21	18000	RENT	5.0	PERSONAL	B	1500	1
7474	25	53000	MORTGAGE	10.0	MEDICAL	B	16000	1
18212	28	16800	OWN	NaN	MEDICAL	C	5000	1
6493	25	50000	MORTGAGE	2.0	VENTURE	A	10000	1

The screenshot shows a Jupyter Notebook interface with two tabs open: 'eda.ipynb' and 'data\_preparation.ipynb'. The 'data\_preparation.ipynb' tab is active, showing code in cell [15] and cell [16]. Cell [15] contains code to load training data from a pickle file. Cell [16] displays the first few rows of the 'X\_train' DataFrame, which has columns: h, loan\_intent, loan\_grade, loan\_amnt, loan\_int\_rate, loan\_percent\_income, cb\_person\_default\_on\_file, cb\_person\_cred\_hist\_length. The data preview shows several rows of loan information.

ii. Secara visual, catat nama kolom yang seharusnya bertipe data

- integer dan float ke variabel baru bernama num\_col
- object ke variabel baru bernama cat\_col

### Lampirkan screenshot

The screenshot shows a Jupyter Notebook interface with two tabs open: 'eda.ipynb' and 'data\_preparation.ipynb'. The 'data\_preparation.ipynb' tab is active, showing code in cell [18]. The code defines two lists: 'num\_col' containing numerical column names like 'person\_age', 'loan\_amnt', etc., and 'cat\_col' containing categorical column names like 'person\_home\_ownership', 'loan\_intent', etc. Both lists are combined into a single list at the end of the cell.

```

[18]: 1 num_col = [
2   "person_age",
3   "person_income",
4   "person_emp_length",
5   "loan_amnt",
6   "loan_int_rate",
7   "loan_percent_income",
8   "cb_person_cred_hist_length"
9 ]
10
11
12 cat_col = [
13   "person_home_ownership",
14   "loan_intent",
15   "loan_grade",
16   "cb_person_default_on_file"
17 ]
18
19 num_col, cat_col

```

iii. Panggil fungsi info() dari variabel X\_train

## Lampirkan screenshot

The screenshot shows a Jupyter Notebook interface with a file tree on the left and a code editor on the right. The code editor displays the following output from the `X_train.info()` function:

```
[19]: 1 X_train.info()
<class 'pandas.DataFrame'>
Index: 26064 entries, 15884 to 17068
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   person_age       26064 non-null   int64  
 1   person_income    26064 non-null   int64  
 2   person_home_ownership  26064 non-null   str    
 3   person_emp_length 25326 non-null   float64 
 4   loan_intent      26064 non-null   str    
 5   loan_grade       26064 non-null   str    
 6   loan_amnt        26064 non-null   int64  
 7   loan_int_rate    23563 non-null   float64 
 8   loan_percent_income 26064 non-null   float64 
 9   cb_person_default_on_file 26064 non-null   str    
10  cb_person_cred_hist_length 26064 non-null   int64  
dtypes: float64(3), int64(4), str(4)
memory usage: 2.4 MB
```

- iv. Cocokkan catatan yang telah dibuat dengan keluaran fungsi info()
- iv. Buatlah kesimpulan dari yang telah dilakukan serta tindakan kedepannya pada sel jupyter notebook selanjutnya

## Lampirkan screenshot

The screenshot shows a Jupyter Notebook interface with a file tree on the left and a code editor on the right. The code editor contains a section titled "Conclusion:" followed by observations and bullet points:

```
10 cb_person_cred_hist_length 26064 non-null int64
dtypes: float64(3), int64(4), str(4)
memory usage: 2.4 MB

- Conclusion:

Based on the observations using the head() function:
• Numerical columns:
  person_age, person_income, person_emp_length, loan_amnt, loan_int_rate, loan_percent_income,
  cb_person_cred_hist_length
• Categorical columns:
  person_home_ownership, loan_intent, loan_grade, cb_person_default_on_file

After checking with the info() function, the displayed data types match the initial observations:
• Numerical columns have data types int64 and float64
• Categorical columns have data type object

The next step is to check for duplicate values and missing values before proceeding to the preprocessing stage.
```

e. [10 poin] Lakukan pengecekan terhadap data yang duplikat

i. Lakukan pengecekan data duplikat HANYA pada X\_train

ii. Lakukan pengecekan dengan fungsi duplicated

iii. Berikan nilai False pada parameter keep di fungsi duplicated

iv. Urutkan data berdasarkan data pada kolom person income

### Lampirkan screenshot

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

- File Explorer:** Shows a directory structure with files like 'data', 'models', 'src', 'Chalimatu Sadiah...', 'data\_preparation.ipynb', 'data\_preprocessin...', and 'requirements.txt'.
- Code Cell:** Cell [22] contains Python code:

```
# Check duplicate data on X_train
dup_data = X_train[X_train.duplicated(keep=False)]
# Sort based on person_income
dup_data = dup_data.sort_values(by="person_income")
```
- Data Preview:** Below the code cell, the resulting DataFrame 'dup\_data' is displayed:

	person_age	person_income	person_home_ownership	person_emp_length	loan_intent	loan_grade	loan_amnt
15952	24	7800	RENT	1.0	EDUCATION	B	1000
16821	24	7800	RENT	1.0	EDUCATION	B	1000
2431	21	15600	RENT	0.0	MEDICAL	A	2800
17758	21	15600	RENT	0.0	MEDICAL	A	2800
28295	32	18000	OWN	0.0	VENTURE	A	4750
...	...	...	...	...	...	...	...
27677	35	160000	OWN	10.0	VENTURE	B	24000
- Bottom Status Bar:** Shows 'Mode: Command', 'Ln 1, Col 1', and the file name 'eda.ipynb'.

The screenshot shows a Jupyter Notebook interface with two tabs open: 'eda.ipynb' and 'data\_preparation.ipynb'. The 'eda.ipynb' tab is active, displaying a data frame with 192 rows and 11 columns. The columns include 'CODE', 'AGE', 'RENT', 'REV', 'EDUCATION', and 'MEDICAL'. Below the table, a code cell shows the command used to find duplicates: [25]: `X_train.duplicated(keep=False).sum()`. The output of this command is [25]: `np.int64(192)`.

v. Buatlah kesimpulan dari yang telah dilakukan serta tindakan kedepannya pada sel jupyter notebook selanjutnya

#### Lampirkan screenshot

The screenshot shows a Jupyter Notebook interface with two tabs open: 'eda.ipynb' and 'data\_preparation.ipynb'. The 'eda.ipynb' tab is active, displaying a data frame with 192 rows and 11 columns. Below the table, a section titled 'Conclusion:' contains the following text:

Based on the duplicate checking process using the `duplicated()` function with `keep=False`, a total of 192 rows were identified as duplicate data in `X_train`.

This indicates that there are observations with identical values across all columns. Duplicate data can potentially bias the machine learning model because certain information is repeated multiple times.

Therefore, the next step is to remove duplicate rows from `X_train` to ensure that the dataset is clean and does not introduce bias during the training process.

f. [10 poin] Lakukan pengecekan terhadap null value

- i. Lakukan pengecekan pada kolom yang memiliki baris dengan nilai null pada X\_train
- ii. Gunakan fungsi isnull() dan chain dengan fungsi sum()

Lampirkan screenshot

The screenshot shows a Jupyter Notebook interface with two tabs open: 'eda.ipynb' and 'data\_preparation.ipynb'. The 'eda.ipynb' tab is active, displaying Python code and its execution results.

```
[27]: 1 # Check sum of null value on every column
2 null_count = X_train.isnull().sum()
3
4 null_count
```

Output:

```
person_age          0
person_income        0
person_home_ownership 0
person_emp_length    738
loan_intent          0
loan_grade           0
loan_amnt            0
loan_int_rate        2501
loan_percent_income   0
cb_person_default_on_file 0
cb_person_cred_hist_length 0
dtype: int64
```

```
[28]: 1 null_count=null_count > 0]
```

Output:

```
person_emp_length    738
loan_int_rate        2501
dtype: int64
```

The notebook interface includes a sidebar with file navigation, a code editor with syntax highlighting, and a status bar at the bottom indicating 'Mode: Edit' and the current file 'eda.ipynb'.

- iii. Buatlah kesimpulan dari yang telah dilakukan serta tindakan kedepannya pada sel jupyter notebook selanjutnya

Lampirkan screenshot

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

- File Explorer:** Shows a tree view of files and folders including 'data', 'models', 'src', 'Chalimatu Sadiah...', 'data\_preparation.ipynb', and 'data\_preprocessing.ipynb'. The current file is 'eda.ipynb'.
- Code Cell:** Displays the following Python code and its output:

```
loan_percent_income      0
cb_person_default_on_file 0
cb_person_cred_hist_length 0
dtype: int64
```

```
[28]: [1] null_count=null_count > 0]
```

```
[28]: person_emp_length    738
loan_int_rate            2501
dtype: int64
```
- Text Block:** A section titled "Missing Value Checking Conclusion:" states: "Based on the null value checking process using the `isnull().sum()` function, missing values were identified in two columns of the `X_train` dataset."
  - `person_emp_length` has 738 missing values.
  - `loan_int_rate` has 2501 missing values.
- Bottom Status Bar:** Shows "Mode: Command" and "Ln 1, Col 1 eda.ipynb".

g. [10 poin] Lakukan pengecekan distribusi pada tiap input

i. Buat temporary variabel bernama X\_train\_

ii.

iii. Seleksi hanya kolom numerik pada variabel X\_train dan simpan pada variabel X\_train\_ menggunakan daftar nama kolom yang telah dibuat sebelumnya (variabel num\_col)

Lampirkan screenshot

The screenshot shows a Jupyter Notebook interface running on a web browser at localhost:8888. The left sidebar displays a file tree with several notebooks and files. The main area has two tabs: 'eda.ipynb' (active) and 'data\_preparation.ipynb'. The code cell in 'eda.ipynb' contains the following Python code:

```
1 # Make temporary variable with numeric column
2 X_train_ = X_train[num_col]
3
4 X_train_.head()
```

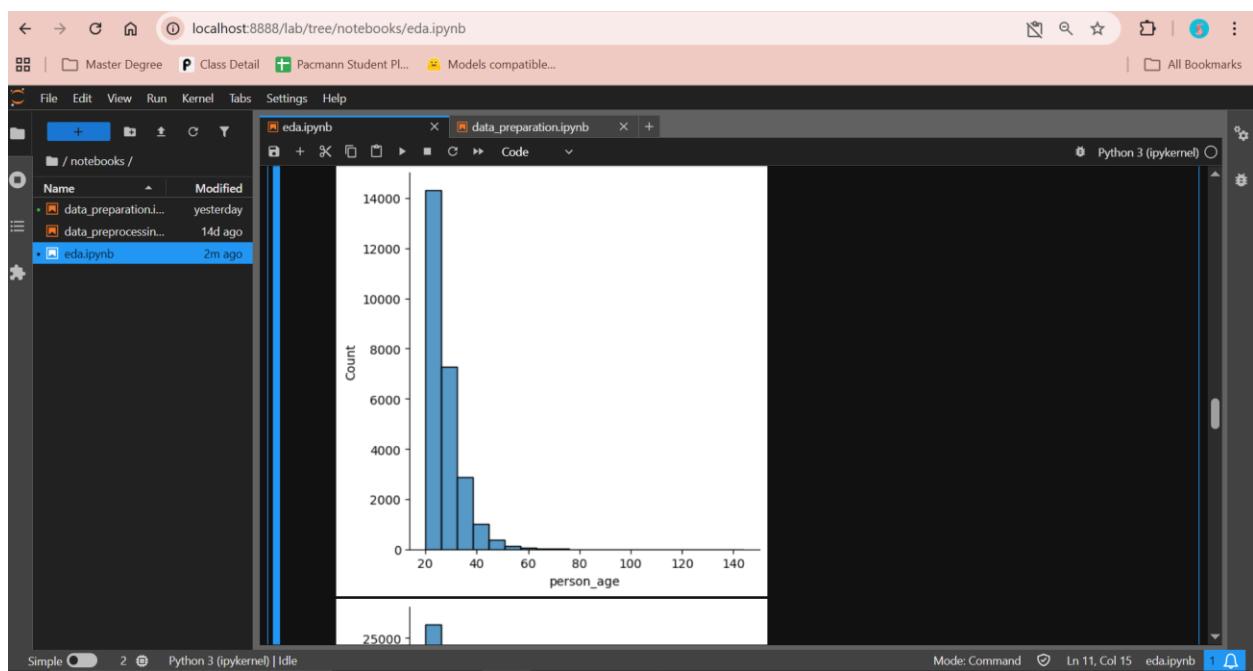
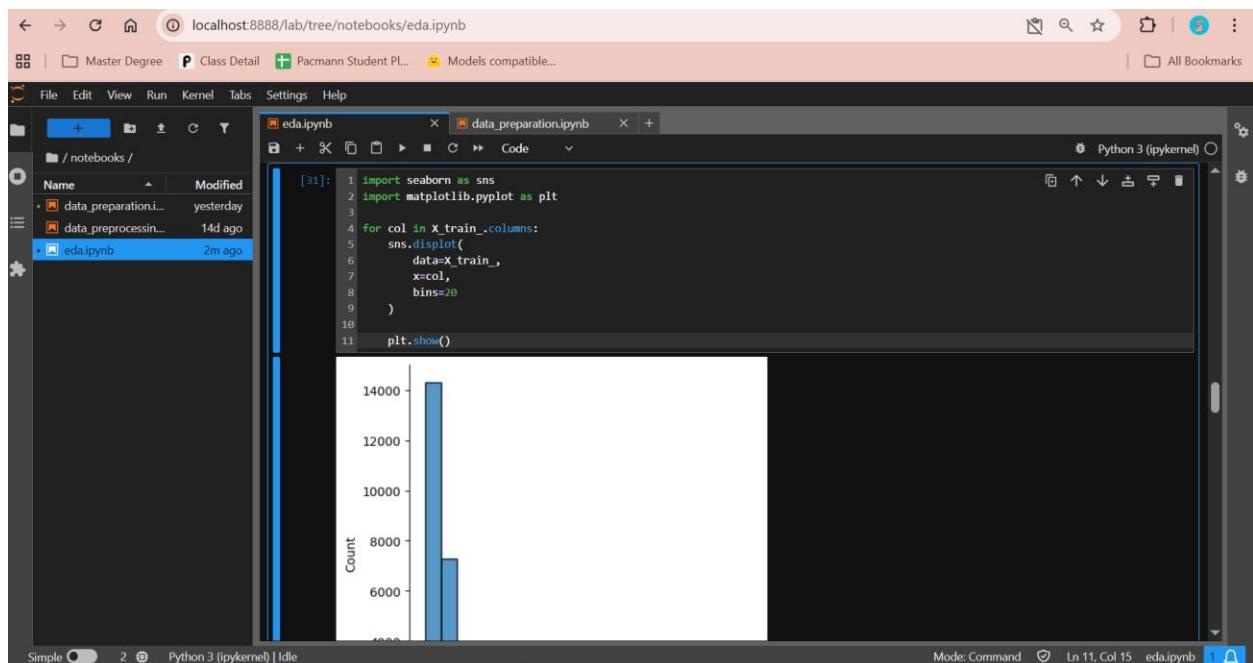
The output cell shows the first five rows of a DataFrame:

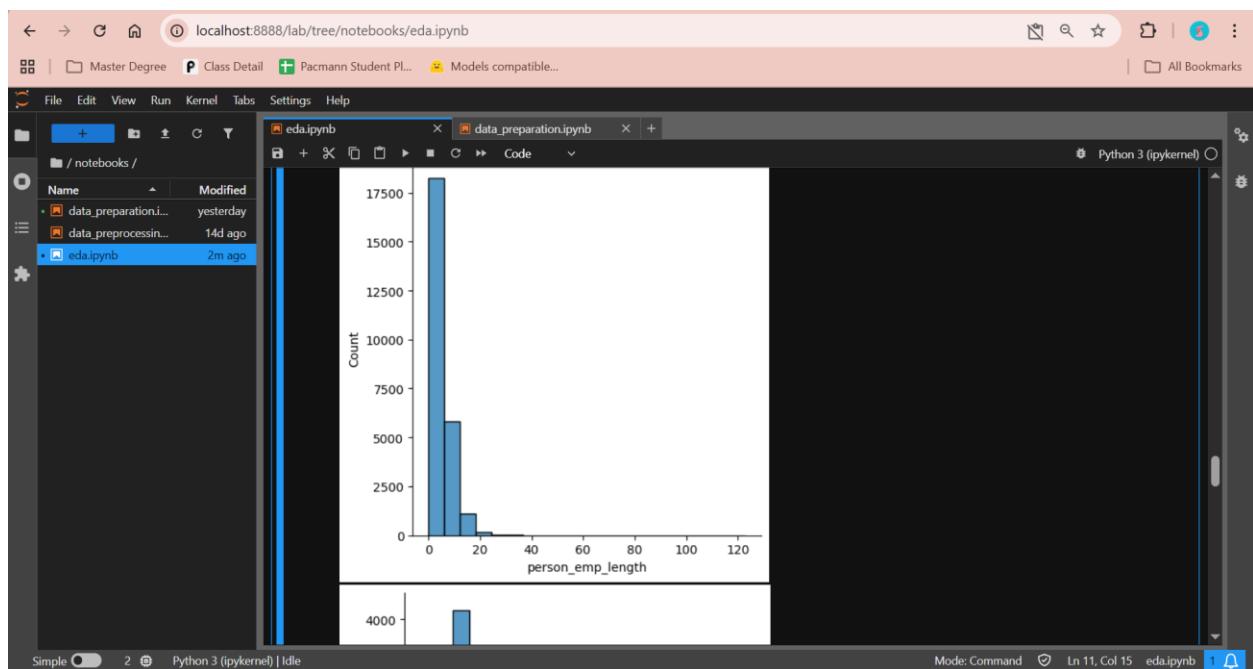
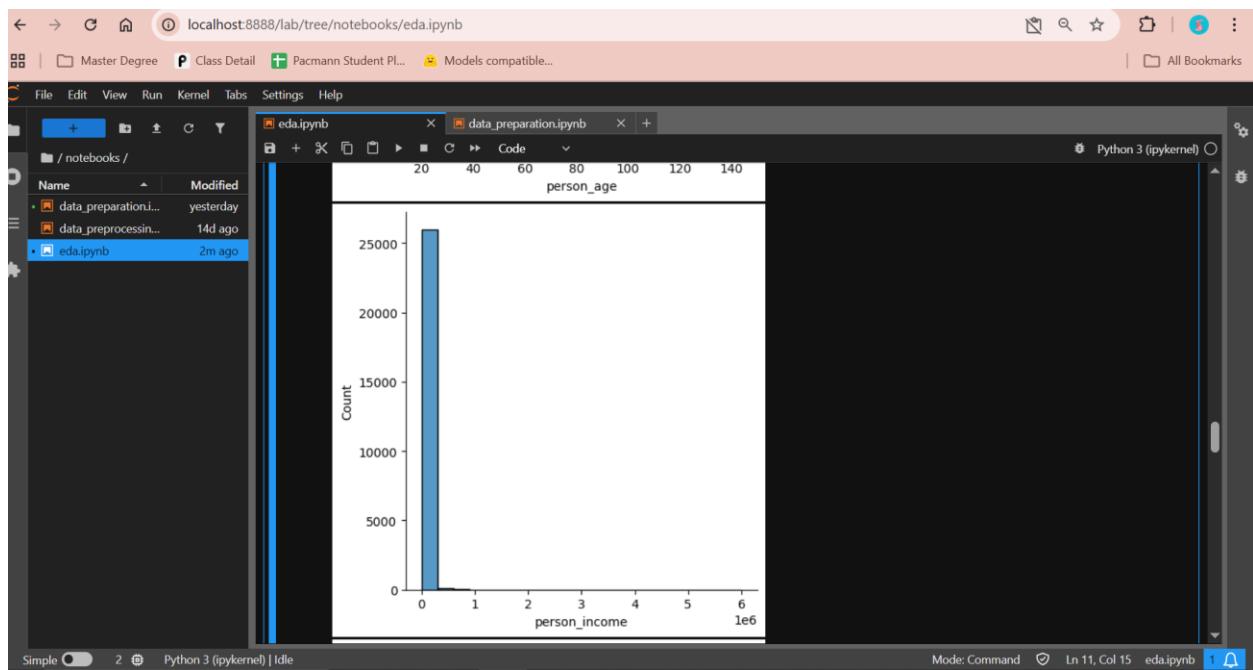
	person_age	person_income	person_emp_length	loan_amnt	loan_int_rate	loan_percent_income	cb_person_cred_hist_len
15884	25	241875	4.0	16000	7.05	0.07	
15138	21	18000	5.0	1500	12.18	0.08	
7474	25	53000	10.0	16000	12.53	0.30	
18212	28	16800	Nan	5000	13.98	0.30	
6493	25	50000	2.0	10000	7.90	0.20	

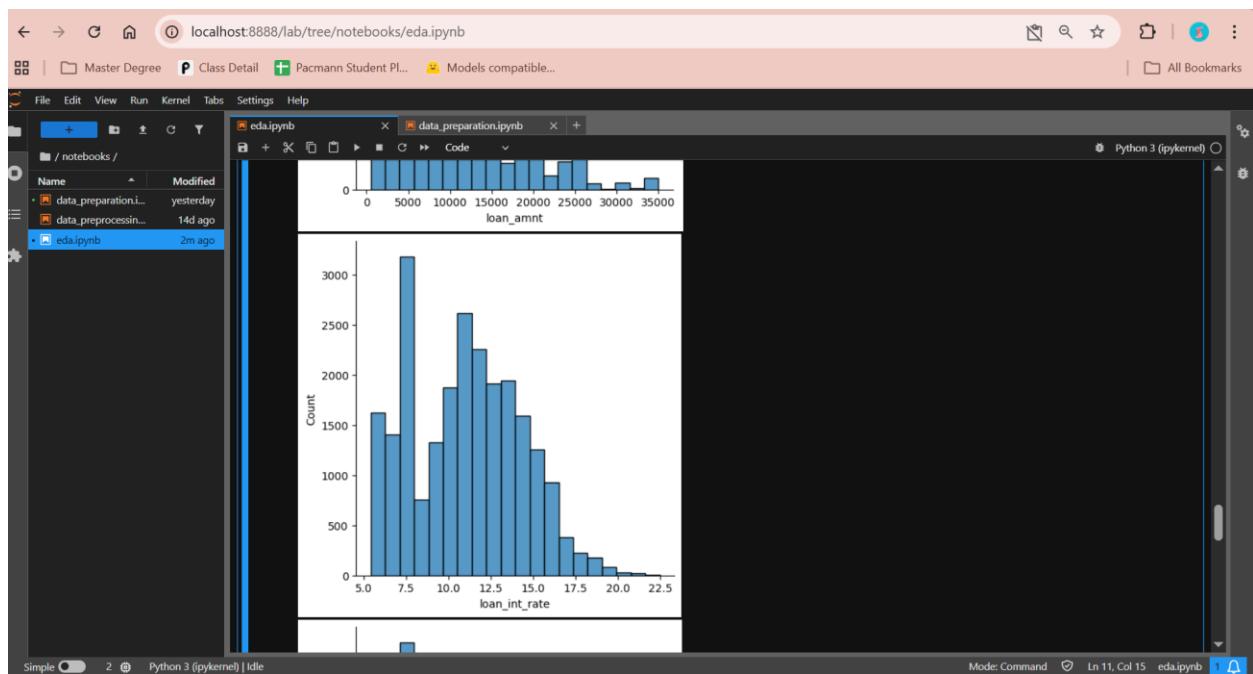
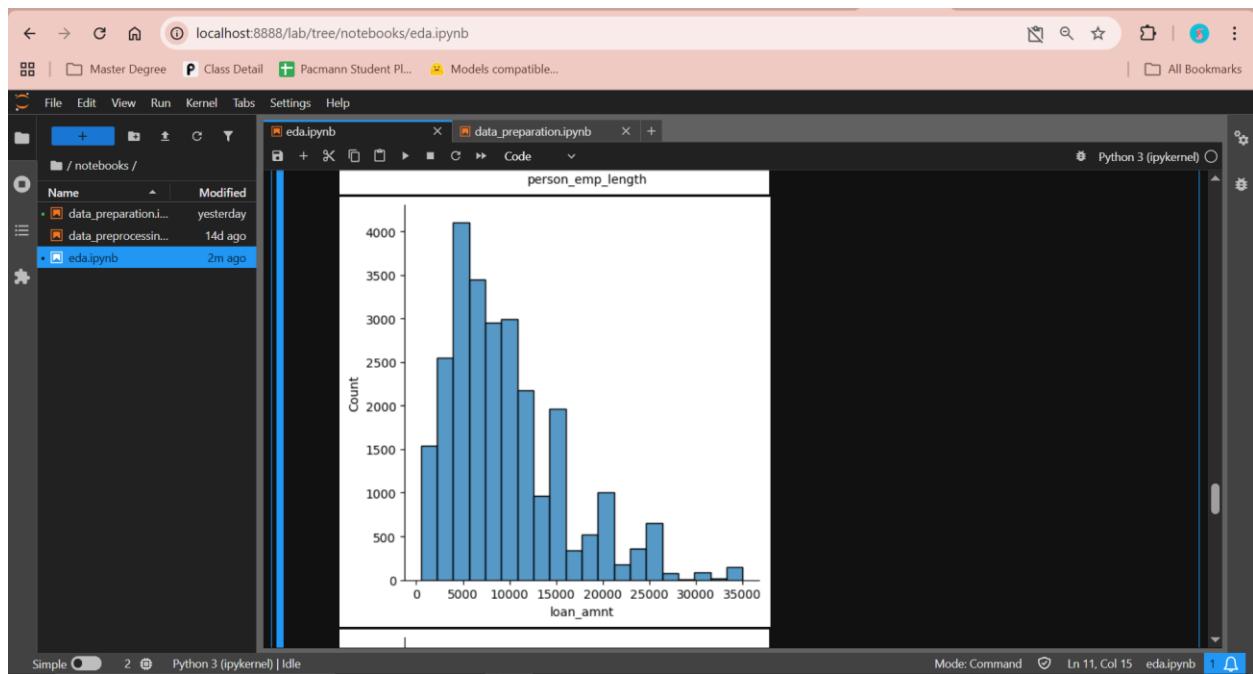
The status bar at the bottom indicates 'Mode: Command' and 'Ln 4, Col 16 eda.ipynb'.

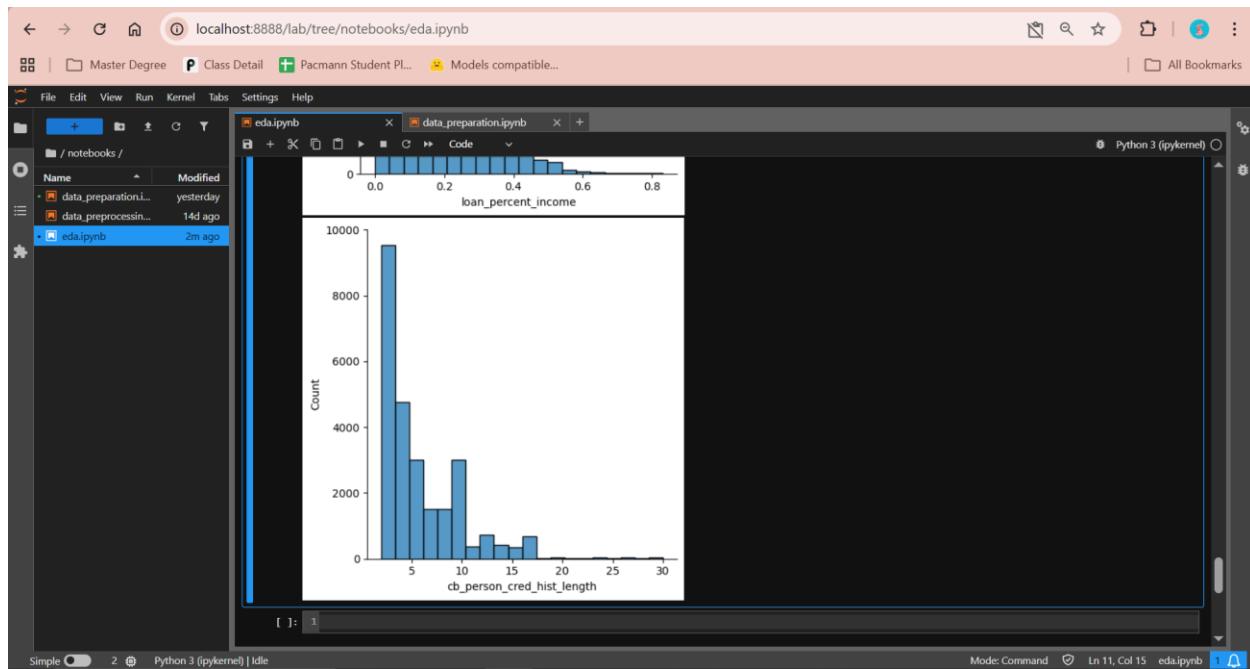
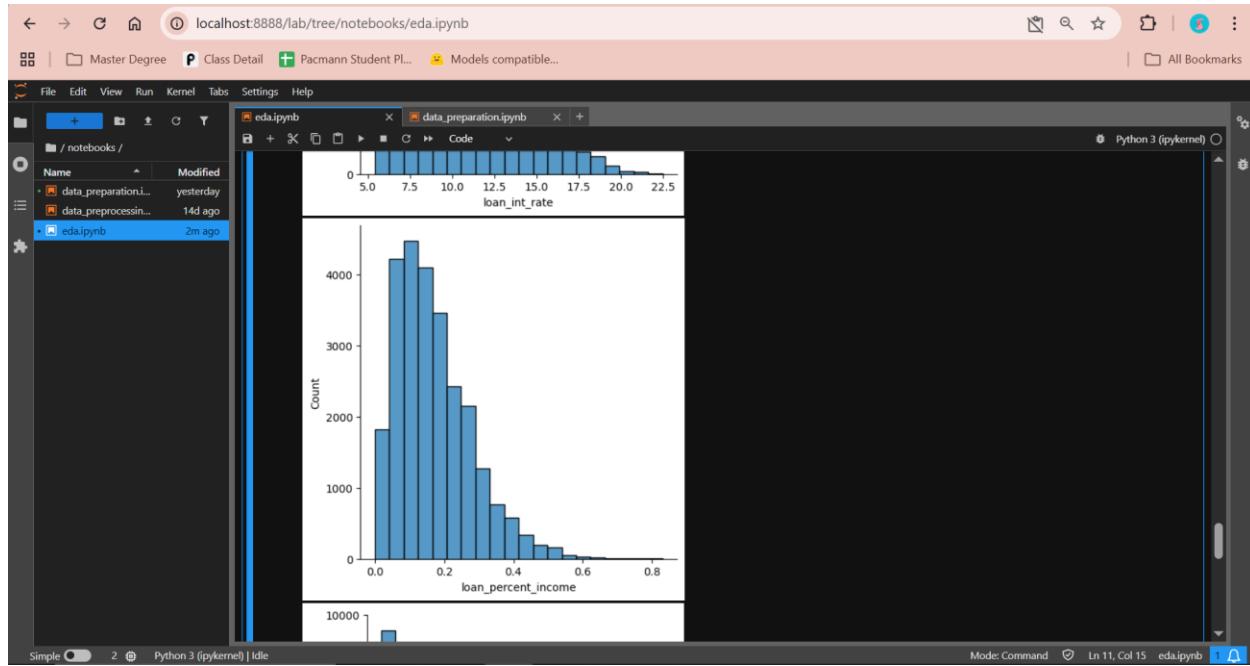
- iv. Buatlah perulangan dari tiap nama kolom pada variabel X\_train\_
- v. Simpan nama kolom pada tiap iterasi perulangan ke variabel bernama col
- vi. Pada tiap iterasi perulangan, panggil fungsi displot() dari library seaborn
  - Data yang digunakan adalah X\_train\_
  - X axis menampilkan data kolom saat ini
  - Gunakan bins 20

Lampirkan screenshot









- vii. Buatlah kesimpulan dari yang telah dilakukan serta tindakan kedepannya pada sel jupyter notebook selanjutnya

Semisal: “dari distribusi ini, saya bisa melakukan xxx untuk tahap preprocessing”

**Lampirkan screenshot**

The screenshot shows a Jupyter Notebook interface with two tabs open: 'eda.ipynb' and 'data\_preparation.ipynb'. The 'eda.ipynb' tab is active, displaying a section titled 'Distribution Analysis Conclusion:' which includes a bulleted list of distribution types for various features.

```
1. person_age  
The distribution is slightly right-skewed, with most values concentrated between 20 and 40 years old. The maximum value appears to be around 70 years, which is still realistic. No extreme outliers are observed.  
2. person_income  
The distribution is heavily right-skewed. Most observations are concentrated at lower income levels, while a small number of observations extend to much higher values. This indicates the presence of high-value outliers.  
3. person_emp_length  
The distribution is right-skewed, with most values under 10 years of employment length. A small number of higher values create a long right tail.  
4. loan_amnt  
The distribution is moderately right-skewed. Most loan amounts are clustered in lower ranges, with fewer high-value loans.  
5. loan_int_rate  
The distribution appears relatively symmetric compared to other features, with no extreme skewness observed.
```

The screenshot shows a Jupyter Notebook interface with two tabs open: 'eda.ipynb' and 'data\_preparation.ipynb'. The 'eda.ipynb' tab is active, displaying a section titled 'Overall Conclusion:' which provides a summary of the dataset's characteristics and preprocessing recommendations.

Most numerical features exhibit right-skewed distributions, particularly income-related variables. This suggests that the dataset contains some high-value observations that may influence model performance.

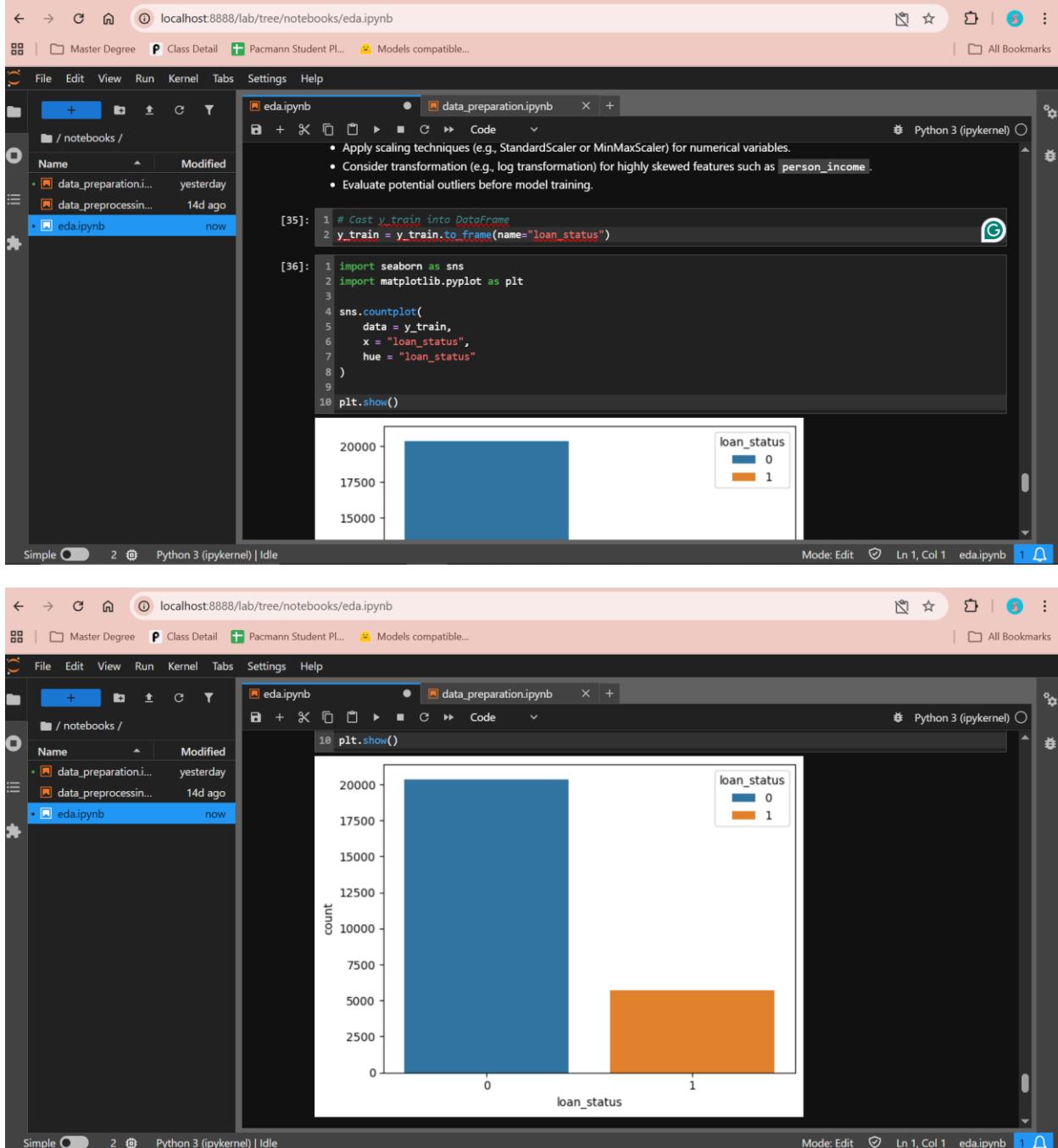
In the preprocessing stage, the following actions may be considered:

- Apply scaling techniques (e.g., StandardScaler or MinMaxScaler) for numerical variables.
- Consider transformation (e.g., log transformation) for highly skewed features such as `person_income`.
- Evaluate potential outliers before model training.

- h. [4 poin] Lakukan pengecekan terhadap target balance
- Panggil fungsi `countplot()` dari library `seaborn`
    - Data yang digunakan adalah `y_train` yang telah diubah ke `dataframe`

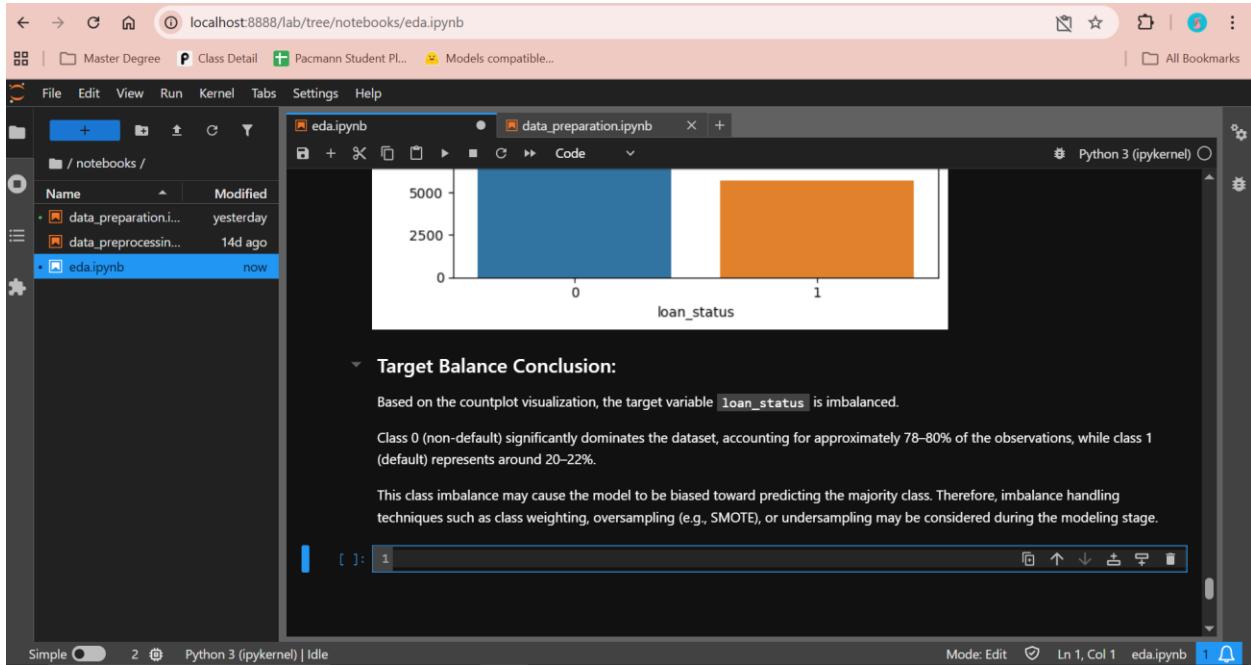
- X axis menampilkan kolom target
- Set hue ke kolom target

### Lampirkan screenshot

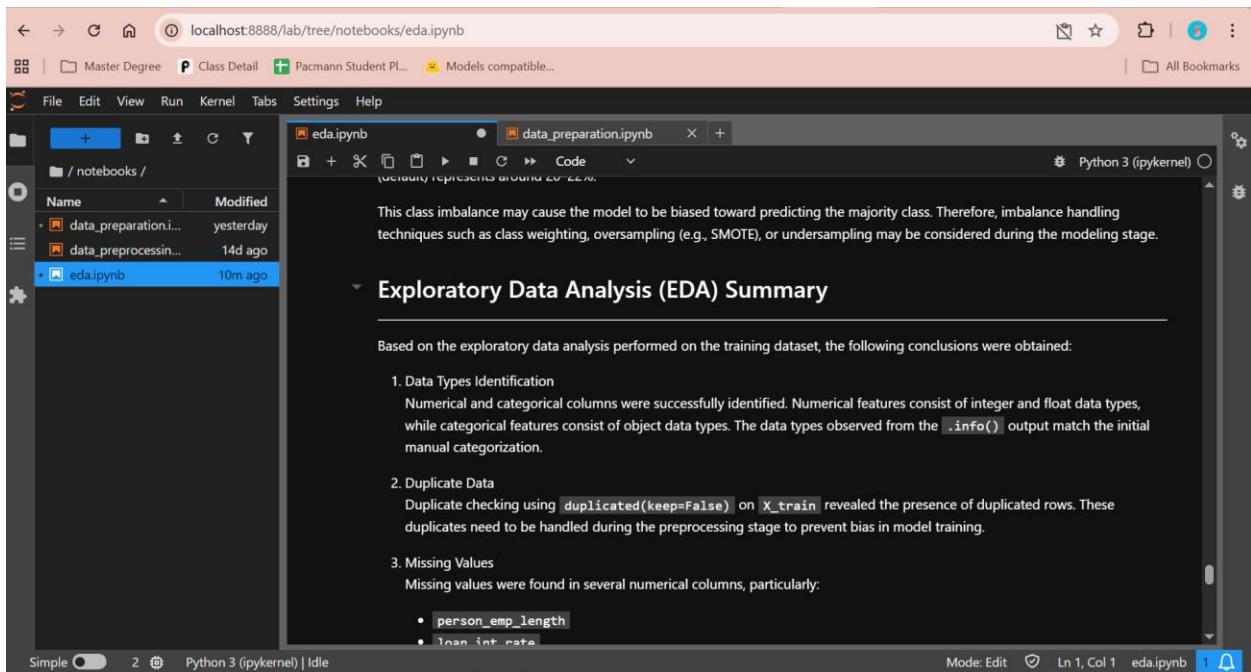


- Buatlah kesimpulan dari yang telah dilakukan serta tindakan kedepannya pada sel jupyter notebook selanjutnya

## Lampirkan screenshot



- i. [6 poin] Rangkum semua kesimpulan yang telah dibuat selama EDA pada cell selanjutnya



localhost:8888/lab/tree/notebooks/eda.ipynb

File Edit View Run Kernel Tabs Settings Help

eda.ipynb data\_preparation.ipynb

Missing values were found in several numerical columns, particularly:

- person\_emp\_length
- loan\_int\_rate

These missing values will require appropriate imputation techniques during preprocessing.

4. Numerical Feature Distribution

Most numerical variables exhibit right-skewed distributions, especially income-related features such as `person_income`. Some variables show moderate skewness but remain within realistic ranges.

Therefore, potential preprocessing steps include:

- Applying scaling techniques (e.g., StandardScaler or MinMaxScaler)
- Considering transformation (e.g., log transformation) for highly skewed variables
- Evaluating and handling outliers if necessary

5. Target Variable Balance

The target variable `loan_status` is imbalanced, with class 0 (non-default) significantly dominating class 1 (default). The distribution is approximately 80:20.

This imbalance may affect model performance, and techniques such as class weighting or resampling may be considered during model development.

### Overall Conclusion

Saving completed

Mode: Edit Python 3 (ipykernel) | Idle

eda.ipynb

localhost:8888/lab/tree/notebooks/eda.ipynb

File Edit View Run Kernel Tabs Settings Help

eda.ipynb data\_preparation.ipynb

This imbalance may affect model performance, and techniques such as class weighting or resampling may be considered during model development.

### Overall Conclusion

The dataset shows common real-world characteristics such as skewed distributions, missing values, duplicate records, and class imbalance. These findings indicate that a structured preprocessing pipeline will be required before model training to ensure optimal and unbiased model performance.

```
[ ]: 1 |
```

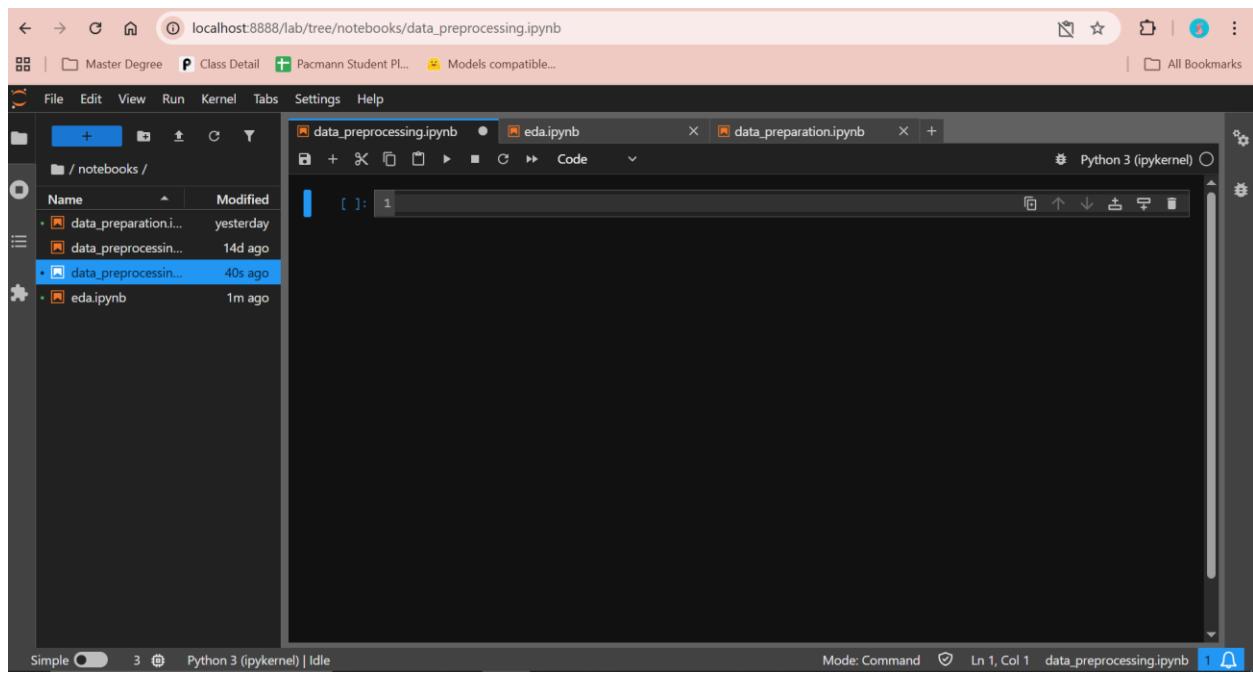
Mode: Edit Python 3 (ipykernel) | Idle

eda.ipynb

## 2. [40 poin] Data Preprocessing

- [1 poin] Buat satu file bernama `data_preprocessing.ipynb` di root project folder Anda

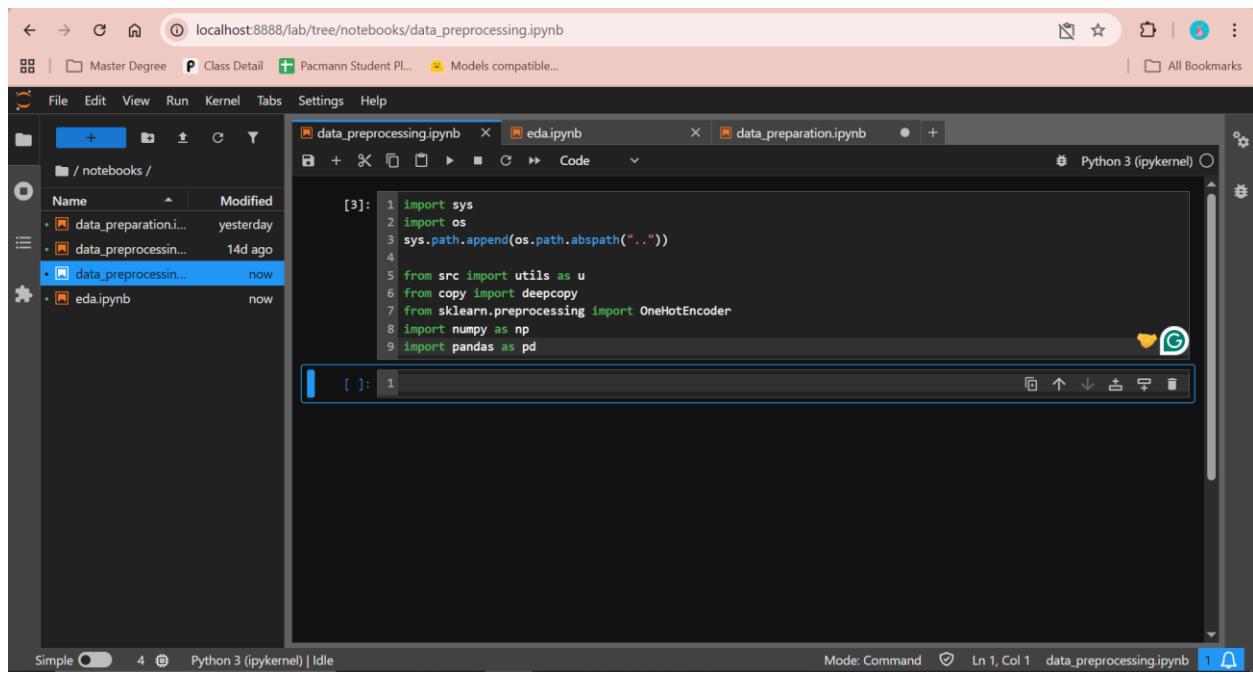
Lampirkan screenshot



b. [1 poin] Import library yang dibutuhkan

Buka file `data_preprocessing.ipynb` dan import `utils` dari folder `src`, `deepcopy` dari library `copy`, `OneHotEncoder` dari `sklearn.preprocessing`, `numpy` dengan alias `np` dan `pandas` dengan alias `pd`

**Lampirkan screenshot**



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

- Header:** localhost:8888/lab/tree/notebooks/data\_preprocessing.ipynb
- Toolbar:** File, Edit, View, Run, Kernel, Tabs, Settings, Help
- File Explorer:** Shows a folder named "notebooks/" containing files: data\_preparation.ipynb (yesterday), data\_preprocessing.ipynb (14d ago), and eda.ipynb (now).
- Code Cell:** Tab [3] contains the following Python code:

```
1 import sys
2 import os
3 sys.path.append(os.path.abspath("../"))
4
5 from src import utils as u
6 from copy import deepcopy
7 from sklearn.preprocessing import OneHotEncoder
8 import numpy as np
9 import pandas as pd
```
- Status Bar:** Mode: Command, Ln 1, Col 1, data\_preprocessing.ipynb

c. [1 poin] Muat splitted dataset

i. Panggil fungsi deserialize\_data() dari utils

- Berikan argumen berupa lokasi dimana data X dan y untuk train, test, dan validation Anda berada
- Simpan keluaran fungsi pada variabel sesuai dengan set datanya

Lampirkan screenshot

```

1 # Define path
2 X_TRAIN_PATH = "../data/interim/X_train.pkl"
3 y_TRAIN_PATH = "../data/interim/y_train.pkl"
4
5 X_VALID_PATH = "../data/interim/X_valid.pkl"
6 y_VALID_PATH = "../data/interim/y_valid.pkl"
7
8 X_TEST_PATH = "../data/interim/X_test.pkl"
9 y_TEST_PATH = "../data/interim/y_test.pkl"
10
11 # Load data
12 X_train = u.deserialize_data(X_TRAIN_PATH)
13 y_train = u.deserialize_data(y_TRAIN_PATH)
14
15 X_valid = u.deserialize_data(X_VALID_PATH)
16 y_valid = u.deserialize_data(y_VALID_PATH)
17
18 X_test = u.deserialize_data(X_TEST_PATH)
19 y_test = u.deserialize_data(y_TEST_PATH)

[5]: 1 print("Train:", X_train.shape)
2 print("Valid:", X_valid.shape)
3 print("Test:", X_test.shape)

Train: (26064, 11)
Valid: (3258, 11)
Test: (3259, 11)

```

d. [13 poin] Buang data duplikat

i. Buat fungsi untuk drop duplicate

- Buat fungsi bernama drop\_duplicate\_data()
- Fungsi tersebut memiliki parameter bernama X dan y
- Parameter X
  - a. Harus bertipe dataframe
  - b. Merupakan set data (train, test, ataupun valid) yang ingin dibuang data duplicatenya
- Parameter y
  - a. Harus bertipe series
  - b. Merupakan data target dari data pada parameter X
- Buat 3 percabangan untuk validasi parameter

Buat 3 percabangan untuk

- a. Percabangan pertama: melakukan komparasi antara parameter X dan dataframe
  - b. Percabangan kedua: melakukan komparasi antara parameter y dan series
  - c. Percabangan ketiga: print pesan “Fungsi drop\_duplicate\_data: parameter telah divalidasi.”
- Lakukan copy pada parameter X dan y dengan menggunakan fungsi copy() dari pandas dan simpan pada variabel yang sama
  - Print pesan tentang shape dari data pada parameter X
    - a. Pesannya adalah “Fungsi drop\_duplicate\_data: shape dataset sebelum dropping duplicate adalah [shape dari data pada parameter X].”
    - b. Gunakan macro format (f) pada string yang akan diprint menggunakan fungsi print
    - c. Akses atribut shape pada data X untuk mendapatkan shape data tersebut
  - Pengecekan duplicate
    - a. Buat satu variabel baru bernama X\_duplicate
    - b. Variabel tersebut diisi oleh dataframe baru hasil seleksi dari data X yang duplicate
      - i. Gunakan teknik seleksi dataframe dengan menggunakan fungsi duplicated()
      - ii. Ambil hanya baris yang memiliki nilai kembalian dari fungsi duplicated() True
  - Print pesan tentang shape dari data yang duplikat
    - a. Pesannya adalah “Fungsi drop\_duplicate\_data: shape dari data yang duplicate adalah [shape dari data yang duplikat].”
    - b. Gunakan macro format (f) pada string yang akan diprint menggunakan fungsi print
    - c. Akses attribute shape dari data X\_duplicate untuk mendapatkan shape data tersebut
  - Kalkulasi shape setelah drop duplicate
    - a. Buat satu variabel bernama X\_clean
    - b. Variabel X\_clean bertipe tuple

- c. Untuk elemen pertama, isi dengan selisih baris antara data X yang belum didrop duplicate dan data X\_duplicate yang merupakan data duplicate dari X
  - d. Untuk elemen kedua, isi dengan nilai kolom dari data X
- Print pesan tentang hasil prediksi kalkulasi shape dari data X setelah didrop duplicate
    - a. Pesannya adalah “Fungsi drop\_duplicate\_data: shape dataset setelah drop duplicate seharusnya adalah [prediksi hasil shape].”
    - b. Gunakan macro format (f) pada string yang akan diprint menggunakan fungsi print
    - c. Gunakan X\_clean sebagai hasil prediksi shape yang akan ditampilkan dari data X
  - Lakukan drop duplicate pada data X dengan menggunakan fungsi drop\_duplicates() dari pandas
    - a. Set parameter inplace menjadi True
  - Lakukan seleksi pada data y dengan menggunakan index dari data X yang telah didrop duplicate
  - Print pesan tentang shape dari data X setelah dilakukan dropping duplicate
    - a. Pesannya adalah “Fungsi drop\_duplicate\_data: shape dataset setelah dropping duplicate adalah [shape dari data X yang telah didrop duplicate].”
    - b. Gunakan macro format (f) pada string yang akan diprint menggunakan fungsi print
    - c. Akses atribut shape pada data X untuk mendapatkan nilai shape
  - Kembalikan X dan y
  - Buat docstring yang menjelaskan tujuan dari fungsi tersebut serta parameternya

Lampirkan screenshot

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

- Title Bar:** localhost:8888/lab/tree/notebooks/data\_preprocessing.ipynb
- Toolbar:** File, Edit, View, Run, Kernel, Tabs, Settings, Help
- File Explorer:** Shows a folder structure under /notebooks/:
  - data\_preparation.ipynb (modified 2m ago)
  - data\_preprocessing.ipynb (modified 14d ago)
  - data\_preprocessing.ipynb (modified now)
  - eda.ipynb (modified 31s ago)
- Code Editor:** Tab [7]:

```
1 def drop_duplicate_data(X, y):
2     """
3         Function to remove duplicate rows from dataset X
4         and align the target variable y accordingly.
5
6     Parameters:
7     -----
8     X : pandas.DataFrame
9         Feature dataset (train, valid, or test) to remove
10        duplicates from.
11
12     y : pandas.Series
13         Target variable corresponding to dataset X.
14
15     Returns:
16     -----
17     X : pandas.DataFrame
18         Cleaned feature dataset after duplicate removal.
19
20     y : pandas.Series
21         Target variable align with cleaned dataset.
22
23     """
24
25     # Parameter validation
26     if not isinstance(X, pd.DataFrame):
27         raise TypeError("Parameter X must be a pandas DataFrame.")
28
29     if not isinstance(y, pd.Series):
30         raise TypeError("Parameter y must be a pandas Series.")
31
32     print("Fungsi drop_duplicate_data: parameter telah divalidasi.")
33
34     # Copy Data
35     X = X.copy()
36
37     y = y.copy()
38
39     # Shape before dropping
40     print(f"Fungsi drop_duplicate_data: shape dataset sebelum dropping duplicate adalah {X.shape}.")
41
42     # Duplicate checking
43     X_duplicate = X[X.duplicated()]
44
45     print(f"Fungsi drop_duplicate_data: shape dari data yang duplicate adalah {X_duplicate.shape}.")
46
47     # Predicted shape after dropping duplicates
```
- Kernel:** Python 3 (ipykernel)
- Status Bar:** Mode: Command, Ln 1, Col 1, data\_preprocessing.ipynb, 1

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

- Title Bar:** localhost:8888/lab/tree/notebooks/data\_preprocessing.ipynb
- Toolbar:** File, Edit, View, Run, Kernel, Tabs, Settings, Help
- File Explorer:** Shows a folder structure under /notebooks/:
  - data\_preparation.ipynb (modified 2m ago)
  - data\_preprocessing.ipynb (modified 14d ago)
  - data\_preprocessing.ipynb (modified now)
  - eda.ipynb (modified 31s ago)
- Code Editor:** Tab [7]:

```
23
24
25     # Parameter validation
26     if not isinstance(X, pd.DataFrame):
27         raise TypeError("Parameter X must be a pandas DataFrame.")
28
29     if not isinstance(y, pd.Series):
30         raise TypeError("Parameter y must be a pandas Series.")
31
32     print("Fungsi drop_duplicate_data: parameter telah divalidasi.")
33
34     # Copy Data
35     X = X.copy()
36
37     y = y.copy()
38
39     # Shape before dropping
40     print(f"Fungsi drop_duplicate_data: shape dataset sebelum dropping duplicate adalah {X.shape}.")
41
42     # Duplicate checking
43     X_duplicate = X[X.duplicated()]
44
45     print(f"Fungsi drop_duplicate_data: shape dari data yang duplicate adalah {X_duplicate.shape}.")
46
47     # Predicted shape after dropping duplicates
```
- Kernel:** Python 3 (ipykernel)
- Status Bar:** Saving completed, Mode: Command, Ln 1, Col 1, data\_preprocessing.ipynb, 1

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

- Header:** localhost:8888/lab/tree/notebooks/data\_preprocessing.ipynb
- Toolbar:** File, Edit, View, Run, Kernel, Tabs, Settings, Help
- File Explorer:** Shows notebooks /, data\_preparation.ipynb (modified 2m ago), data\_preprocessing.ipynb (modified 14d ago), and eda.ipynb (modified now).
- Code Editor:** The current tab is data\_preprocessing.ipynb. The code defines a function drop\_duplicate\_data that prints the shape of the dataset before and after dropping duplicates, checks for duplicates, drops them, aligns y with cleaned X, and returns X and y.
- Output:** In cell [12], the command `X_train, y_train = drop_duplicate_data(X_train, y_train)` is run, followed by the validation message "Fungsi drop\_duplicate\_data: parameter telah divalidasi."
- Status Bar:** Mode: Command, Ln 1, Col 1, data\_preprocessing.ipynb, 1

## ii. Jalankan fungsi

- Panggil fungsi `drop_duplicate_data()`
- Berikan data `X_train` dan `y_train` sebagai argumen untuk parameter `X` dan `y` pada fungsi tersebut
- Simpan keluaran fungsi tersebut ke variabel `X_train` dan `y_train` (direplace)

Lampirkan screenshot

The screenshot shows a Jupyter Notebook interface running on localhost:8888. The left sidebar lists notebooks: 'data\_preparation.ipynb' (modified 2m ago), 'data\_preprocessing.ipynb' (modified 14d ago), 'eda.ipynb' (modified 31s ago), and 'data\_preprocessing.ipynb' (now). The main area has three tabs: 'data\_preprocessing.ipynb', 'eda.ipynb', and 'data\_preparation.ipynb'. The 'data\_preprocessing.ipynb' tab is active, showing a code cell output:

```
[12]: 1 X_train, y_train = drop_duplicate_data(X_train, y_train)
Fungsi drop_duplicate_data: parameter telah divalidasi.
Fungsi drop_duplicate_data: shape dataset sebelum dropping duplicate adalah (26064, 11).
Fungsi drop_duplicate_data: shape dari data yang duplicate adalah (96, 11).
Fungsi drop_duplicate_data: shape dataset setelah drop duplicate seharusnya adalah (25968, 11).
Fungsi drop_duplicate_data: shape dataset setelah dropping duplicate adalah (25968, 11).
```

e. [13 poin] Imputasi data numerik

i. Buat fungsi untuk imputasi data numerik

- Buat fungsi bernama `median_imputation()`
- Fungsi tersebut memiliki parameter bernama `data`, `subset_data`, dan `fit`
- Parameter `data` harus bertipe `dataframe`
- Parameter `data` merupakan set data (train, test, ataupun valid) yang ingin diimputasi datanya
- Parameter `fit` harus bertipe `data boolean`
- Jika parameter `fit` bernilai `True` maka fungsi akan melakukan kalkulasi nilai median dari tiap nama kolom yang telah dispesifikan pada parameter `subset_data`
- Jika parameter `fit` bernilai `False` maka fungsi akan melakukan imputasi pada data berdasarkan kalkulasi sebelumnya (`fit = True`)
- Untuk melakukan imputasi, pengguna diharuskan untuk melakukan `fit` terlebih dahulu

- Parameter subset\_data untuk fit = True bertipe list 1 dimensi
- a. Parameter subset\_data bertipe list merupakan daftar nama kolom dari data yang ingin diimputasi
- Parameter subset\_data untuk fit = False bertipe dict
- a. Parameter subset\_data bertipe dict merupakan daftar nama kolom sebagai key beserta nilai mediannya sebagai value
- Buat percabangan untuk validasi parameter
- a. Percabangan pertama
- i.Melakukan komparasi antara parameter data dan dataframe
  - ii.Gunakan fungsi isinstance() untuk melakukan komparasi
  - iii.Jika parameter data tidak sama dengan dataframe, maka raise runtimeerror dengan pesan “Fungsi median\_imputation: parameter data haruslah bertipe DataFrame!”
  - iv.Jika parameter data sama dengan dataframe, maka eksekusi kode diluar cakupan percabangan ini
- b. Percabangan kedua
- i.Melakukan komparasi antara parameter fit dan True
  - ii.Gunakan operator equal untuk melakukan komparasi ini
  - iii.Jika parameter fit sama dengan True:
    1. Buat anak cabang pertama
    - a. Melakukan komparasi antara parameter subset\_data dengan list
    - b. Gunakan fungsi instance untuk melakukan komparasi
    - c. Jika parameter subset\_data tidak sama dengan list, maka raise runtimeerror dengan pesan “Fungsi median\_imputation: untuk nilai parameter fit = True, subset\_data harus bertipe list dan berisi daftar nama kolom yang ingin dicari nilai mediannya guna menjadi data imputasi pada kolom tersebut.”
    - d. Jika parameter subset\_data sama dengan list, maka eksekusi kode diluar cakupan percabangan ini
  - iv.Jika parameter fit tidak sama dengan True, maka eksekusi percabangan ketiga

c. Percabangan ketiga

i.Melakukan komparasi antara parameter fit dan False

ii.Gunakan operator equal untuk melakukan komparasi ini

iii.Jika parameter fit sama dengan False:

1. Buat anak cabang pertama

- a. Melakukan komparasi antara parameter subset\_data dengan dict
- b. Gunakan fungsi instance untuk melakukan komparasi
- c. Jika parameter subset\_data tidak sama dengan dict, maka raise runtimeerror dengan pesan “Fungsi median\_imputation: untuk nilai parameter fit = False, subset\_data harus bertipe dict dan berisi key yang merupakan nama kolom beserta value yang merupakan nilai median dari kolom tersebut.”
- d. Jika parameter subset\_data sama dengan dict, maka eksekusi kode diluar cakupan percabangan ini

iv.Jika parameter fit tidak sama dengan False, maka eksekusi percabangan keempat

d. Percabangan keempat

i.Raise runtimeerror dengan pesan “Fungsi median\_imputation: parameter fit haruslah bertipe boolean, bernilai True atau False.”

- Print pesan bahwa parameter telah divalidasi

a. Pesannya adalah “Fungsi median\_imputation: parameter telah divalidasi.”

- Lakukan copy pada parameter data

a. Gunakan fungsi copy() dari pandas

b. Simpan pada variabel baru bernama data (direplace)

- Lakukan copy pada parameter subset\_data

a. Gunakan fungsi deep\_copy() dari library copy

b. Simpan pada variabel baru bernama subset\_data (direplace)

- Buat percabangan untuk parameter fit bernilai True, False, dan selain keduanya

a. Pada percabangan dengan parameter fit bernilai True:

i.Buat satu variabel bernama imputation\_data dan bertipe dict

ii.Buat perulangan dari subset\_data yang akan memberikan tiap elemennya

iii.Simpan elemen tersebut pada variabel bernama subset

iv.Dalam perulangan tersebut:

1. Seleksi data berdasarkan nama kolom yang terdapat pada variabel subset
2. Kalkulasi nilai mediannya
3. Simpan data nilai median beserta nama kolomnya dalam variabel imputation\_data

v.Print pesan tentang hasil dari fitting

1. Pesannya adalah “Fungsi median\_imputation: proses fitting telah selesai, berikut hasilnya [hasil dari fitting].”
2. Gunakan macro format (f) pada string yang akan diperintahkan menggunakan fungsi print
3. Gunakan imputation\_data sebagai hasil dari fitting.

vi.Kembalikan variabel imputation\_data dari fungsi

b. Pada percabangan dengan parameter fit bernilai False:

i.Print pesan tentang kondisi dari tiap kolom sebelum dilakukan imputasi

1. Pesan pertama yang diperintahkan adalah “Fungsi median\_imputation: informasi count na sebelum dilakukan imputasi.”
2. Pesan kedua yang diperintahkan adalah keluaran dari chaining fungsi isna() dan sum() pada parameter data
3. Pesan ketiga adalah string kosong, hal ini sebagai new line

ii.Lakukan imputasi dengan memanggil fungsi fillna()

1. Gunakan subset\_data sebagai argumen pada fungsi fillna()

2. Gunakan nilai True pada parameter inplace di fungsi fillna()

iii. Print pesan tentang kondisi dari tiap kolom setelah dilakukan imputasi

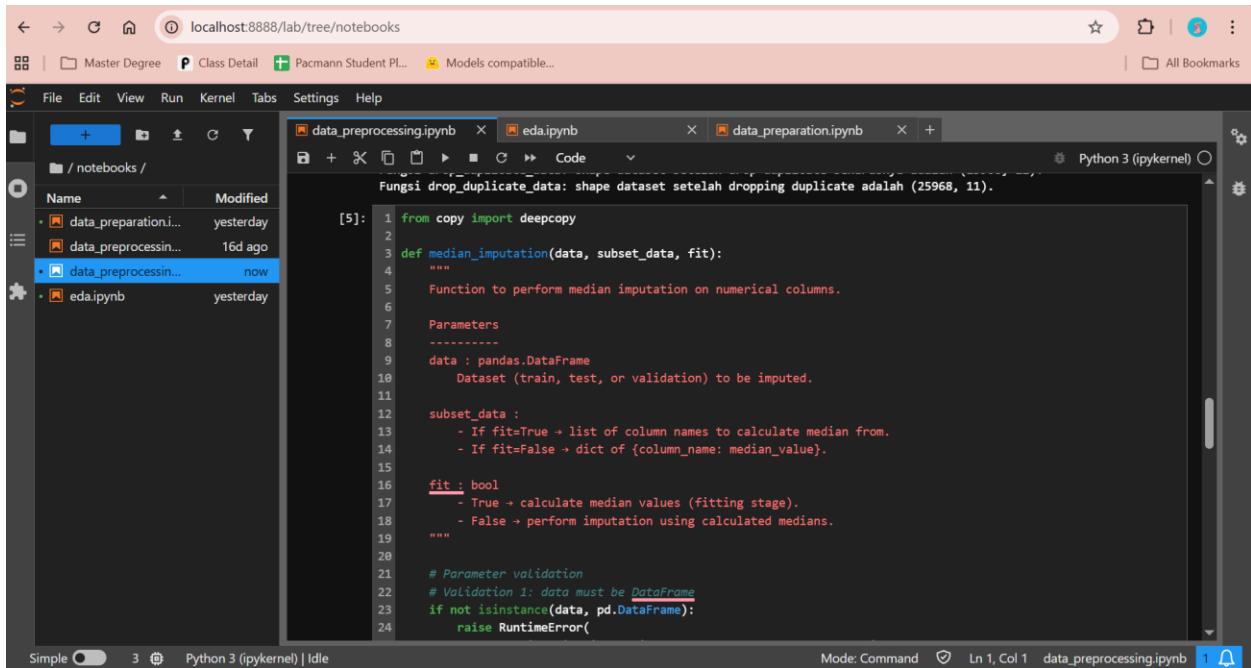
1. Pesan pertama yang diprint adalah “Fungsi median\_imputation: informasi count na setelah dilakukan imputasi:”

2. Pesan kedua yang diprint adalah keluaran dari chaining fungsi isna() dan sum() pada parameter data

3. Pesan ketiga adalah string kosong, hal ini sebagai new line

iv. Kembalikan variabel data dari fungsi

#### Lampirkan screenshot



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

- Header:** localhost:8888/lab/tree/notebooks
- File Menu:** File, Edit, View, Run, Kernel, Tabs, Settings, Help
- Toolbar:** Back, Forward, Stop, Refresh, Home, All Bookmarks
- Left Sidebar:** Shows a file tree with notebooks: data\_preprocessing.ipynb (modified yesterday), data\_preprocessing.ipynb (modified 16d ago), data\_preprocessing.ipynb (modified now), and eda.ipynb (modified yesterday).
- Code Cell:** [5]:

```
from copy import deepcopy
def median_imputation(data, subset_data, fit):
    """
    Function to perform median imputation on numerical columns.

    Parameters
    -----
    data : pandas.DataFrame
        Dataset (train, test, or validation) to be imputed.

    subset_data :
        - If fit=True + list of column names to calculate median from.
        - If fit=False + dict of {column_name: median_value}.

    fit : bool
        - True → calculate median values (fitting stage).
        - False → perform imputation using calculated medians.

    """
    # Parameter validation
    # Validation 1: data must be DataFrame
    if not isinstance(data, pd.DataFrame):
        raise RuntimeError(
```
- Output Cell:** Fungsi drop\_duplicate\_data: shape dataset setelah dropping duplicate adalah (25968, 11).
- Kernel:** Python 3 (ipykernel)
- Bottom Status Bar:** Simple, Python 3 (ipykernel) | Idle, Mode: Command, Ln 1, Col 1, data\_preprocessing.ipynb

The screenshot shows a Jupyter Notebook interface with three tabs open: `data_preprocessing.ipynb`, `eda.ipynb`, and `data_preparation.ipynb`. The `data_preprocessing.ipynb` tab is active, displaying Python code for validating parameters of the `median_imputation` function. The code includes validation for `data` (must be a DataFrame), `fit` (must be True or False), and `fit` (must be boolean). It also handles the case where `fit` is not provided.

```
21 # Parameter validation
22 # Validation 1: data must be DataFrame
23 if not isinstance(data, pd.DataFrame):
24     raise RuntimeError(
25         "Fungsi median_imputation: parameter data haruslah bertipe DataFrame!"
26     )
27
28 # Validation 2: fit == True
29 if fit == True:
30     if not isinstance(subset_data, list):
31         raise RuntimeError(
32             "Fungsi median_imputation: untuk nilai parameter fit = True, subset_data harus bertipe list dan berisi daftar nama kolom yang ingin dicari nilai mediannya guna menjadi data imputasi pada kolom tersebut."
33         )
34
35 # Validation 3: fit == False
36 elif fit == False:
37     if not isinstance(subset_data, dict):
38         raise RuntimeError(
39             "Fungsi median_imputation: untuk nilai parameter fit = False, subset_data harus bertipe dict dan berisi key yang merupakan nama kolom beserta value yang merupakan nilai median dari kolom tersebut."
40         )
41
42 # Validation 4: fit must be boolean
43 else:
44     raise RecursionError()
```

The screenshot shows the continuation of the `median_imputation` function implementation in the `data_preprocessing.ipynb` notebook. The code prints a validation message, copies the data, and enters the fitting stage where it iterates over subsets to calculate median values and store them in a dictionary.

```
41
42     # Validation 4: fit must be boolean
43 else:
44     raise RecursionError(
45         "Fungsi median_imputation: parameter fit haruslah bertipe boolean, bernilai True atau False."
46     )
47
48 print("Fungsi median_imputation: parameter telah divalidasi.")
49
50
51 # Copy data
52 data = data.copy()
53 subset_data = deepcopy(subset_data)
54
55
56 # Fitting stage
57 if fit == True:
58
59     imputation_data = dict()
60
61     for subset in subset_data:
62         median_value = data[subset].median()
63         imputation_data[subset] = median_value
64         print(
65             "Fungsi median_imputation: proses fitting telah selesai, berikut hasilnya {imputation_data}."
66         )
```

```

58     if fit == True:
59         imputation_data = dict()
60
61         for subset in subset_data:
62             median_value = data[subset].median()
63             imputation_data[subset] = median_value
64
65             print(
66                 f"Fungsi median_imputation: proses fitting telah selesai, berikut hasilnya {imputation_data}."
67             )
68
69         return imputation_data
70
71     # Imputation stage
72     elif fit == False:
73         print("Fungsi median_imputation: informasi count na sebelum dilakukan imputasi:")
74         print(data.isna().sum())
75         print("*")
76
77         data.fillna(subset_data, inplace=True)
78
79         print("Fungsi median_imputation: informasi count na setelah dilakukan imputasi:")
80         print(data.isna().sum())
81         print("*")
82
83     return data

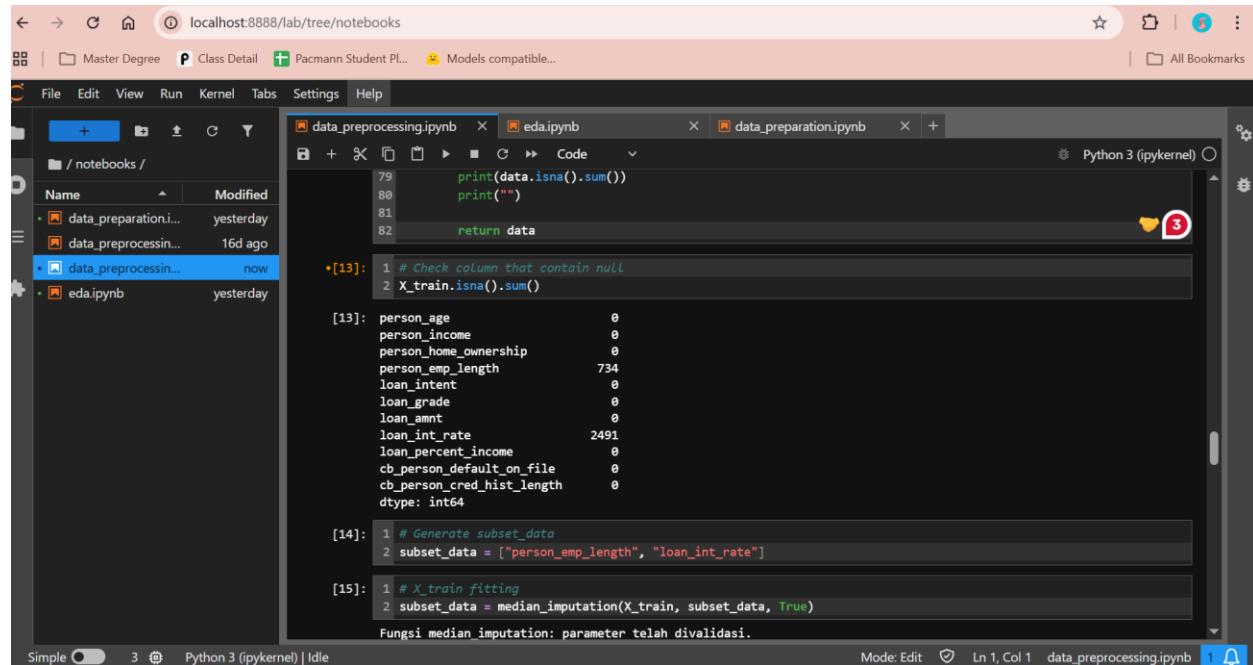
```

## ii. Jalankan fungsi

- Buat variabel bernama `subset_data`
  - Isi variabel tersebut dengan nama kolom yang hendak diimputasi
  - Panggil fungsi `median_imputation()`
- a. Gunakan `X_train`, `subset_data`, dan `True` sebagai argumen untuk parameter `data`, `subset_data`, dan `fit` pada fungsi tersebut secara berurutan
- Simpan keluaran fungsi pada variabel `subset_data` (`replace`)
  - Panggil ulang fungsi `median_imputation()`
- a. Gunakan `X_train`, `subset_data`, dan `False` sebagai argumen untuk parameter `data`, `subset_data`, dan `fit` pada fungsi tersebut secara berurutan
- Simpan keluaran fungsi pada variabel `X_train`
  - Panggil ulang fungsi `median_imputation`
- a. Gunakan `X_test`, `subset_data`, dan `False` sebagai argumen untuk parameter `data`, `subset_data`, dan `fit` pada fungsi tersebut secara berurutan
- Simpan keluaran fungsi pada variabel `X_test`

- Panggil ulang fungsi median\_imputation
- a. Gunakan X\_valid, subset\_data, dan False sebagai argumen untuk parameter parameter data, subset\_data, dan fit pada fungsi tersebut secara berurutan
- Simpan keluaran fungsi pada variabel X\_valid

Lampirkan screenshot



The screenshot shows a Jupyter Notebook interface with three tabs: data\_preprocessing.ipynb, eda.ipynb, and data\_preparation.ipynb. The data\_preprocessing.ipynb tab is active. The code in cell [13] is:

```

1 # Check column that contain null
2 X_train.isna().sum()

```

The output shows the sum of null values for each column:

```

person_age          0
person_income        0
person_home_ownership 0
person_emp_length    734
loan_intent          0
loan_grade            0
loan_amnt             0
loan_int_rate         2491
loan_percent_income   0
cb_person_default_on_file 0
cb_person_cred_hist_length 0
dtype: int64

```

Cell [14] contains:

```

1 # Generate subset_data
2 subset_data = ["person_emp_length", "loan_int_rate"]

```

Cell [15] contains:

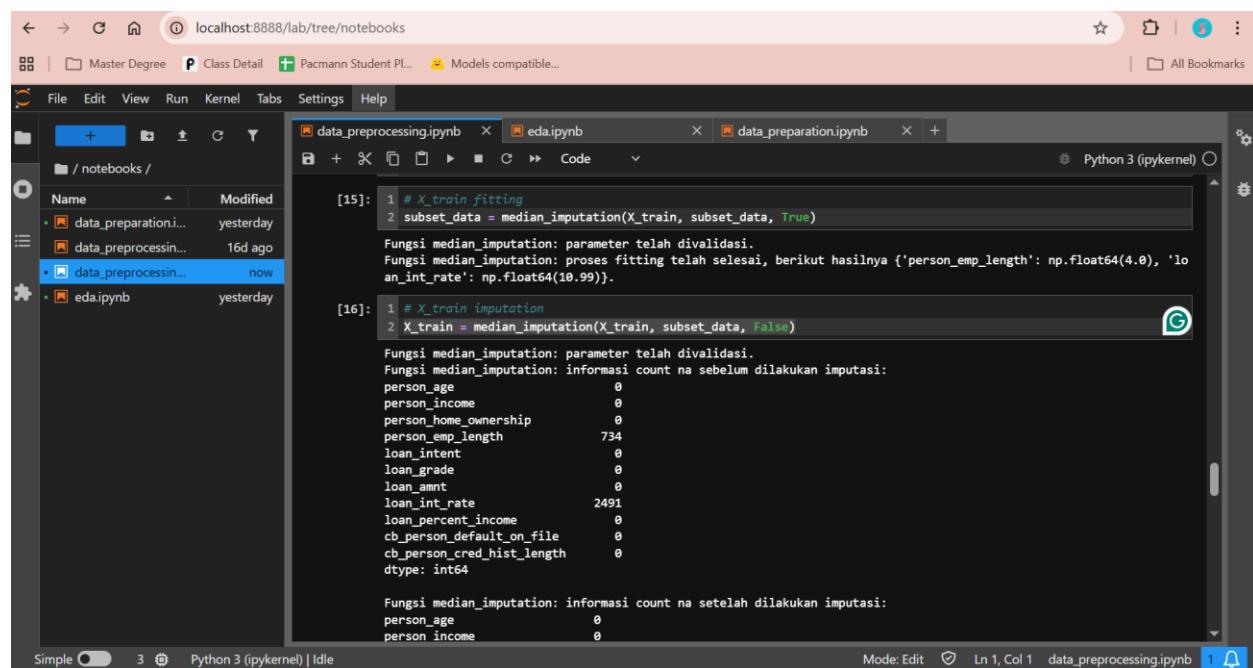
```

1 # X_train fitting
2 subset_data = median_imputation(X_train, subset_data, True)

```

The output for cell [15] is:

Fungsi median\_imputation: parameter telah divalidasi.



The screenshot shows the continuation of the Jupyter Notebook session. Cell [15] has already been run, so its output is shown again:

Fungsi median\_imputation: parameter telah divalidasi.

Cell [16] contains:

```

1 # X_train imputation
2 X_train = median_imputation(X_train, subset_data, False)

```

The output for cell [16] is:

Fungsi median\_imputation: parameter telah divalidasi.  
Fungsi median\_imputation: informasi count na sebelum dilakukan imputasi:  
person\_age 0
person\_income 0
person\_home\_ownership 0
person\_emp\_length 734
loan\_intent 0
loan\_grade 0
loan\_amnt 0
loan\_int\_rate 2491
loan\_percent\_income 0
cb\_person\_default\_on\_file 0
cb\_person\_cred\_hist\_length 0
dtype: int64  
  
Fungsi median\_imputation: informasi count na setelah dilakukan imputasi:  
person\_age 0
person\_income 0

localhost:8888/lab/tree/notebooks

Master Degree Class Detail Pacmann Student Pl... Models compatible... All Bookmarks

File Edit View Run Kernel Tabs Settings Help

data\_preprocessing.ipynb eda.ipynb data\_preparation.ipynb Python 3 (ipykernel)

Name / notebooks / Modified

- data\_preprocessing.ipynb yesterday
- data\_preprocessing.ipynb 1d ago
- data\_preprocessing.ipynb now
- eda.ipynb yesterday

```
Fungsi median_imputation: informasi count na setelah dilakukan imputasi:  
person_age 0  
person_income 0  
person_home_ownership 0  
person_emp_length 0  
loan_intent 0  
loan_grade 0  
loan_amnt 0  
loan_int_rate 0  
loan_percent_income 0  
cb_person_default_on_file 0  
cb_person_cred_hist_length 0  
dtype: int64
```

```
[17]: 1 # X_test_imputation  
2 X_test = median_imputation(X_test, subset_data, False)
```

Fungsi median\_imputation: parameter telah divalidasi.  
Fungsi median\_imputation: informasi count na sebelum dilakukan imputasi:  
person\_age 0  
person\_income 0  
person\_home\_ownership 0  
person\_emp\_length 77  
loan\_intent 0  
loan\_grade 0  
loan\_amnt 0

Mode: Edit Ln 1, Col 1 data\_preprocessing.ipynb

localhost:8888/lab/tree/notebooks

Master Degree Class Detail Pacmann Student Pl... Models compatible... All Bookmarks

File Edit View Run Kernel Tabs Settings Help

data\_preprocessing.ipynb eda.ipynb data\_preparation.ipynb Python 3 (ipykernel)

Name / notebooks / Modified

- data\_preprocessing.ipynb yesterday
- data\_preprocessing.ipynb 1d ago
- data\_preprocessing.ipynb now
- eda.ipynb yesterday

```
cb_person_default_on_file 0  
cb_person_cred_hist_length 0  
dtype: int64
```

```
[17]: 1 # X_test_imputation  
2 X_test = median_imputation(X_test, subset_data, False)
```

Fungsi median\_imputation: parameter telah divalidasi.  
Fungsi median\_imputation: informasi count na sebelum dilakukan imputasi:  
person\_age 0  
person\_income 0  
person\_home\_ownership 0  
person\_emp\_length 77  
loan\_intent 0  
loan\_grade 0  
loan\_amnt 0  
loan\_int\_rate 303  
loan\_percent\_income 0  
cb\_person\_default\_on\_file 0  
cb\_person\_cred\_hist\_length 0  
dtype: int64

```
Fungsi median_imputation: informasi count na setelah dilakukan imputasi:  
person_age 0  
person_income 0  
person_home_ownership 0  
person_emp_length 0  
loan_intent 0
```

Mode: Edit Ln 1, Col 1 data\_preprocessing.ipynb

localhost:8888/lab/tree/notebooks

Master Degree Class Detail Pacmann Student Pl... Models compatible... All Bookmarks

File Edit View Run Kernel Tabs Settings Help

data\_preprocessing.ipynb eda.ipynb data\_preparation.ipynb Python 3 (ipykernel)

Name Modified

- data\_preparation.ipynb yesterday
- data\_preprocessing.ipynb 16d ago
- eda.ipynb yesterday

Code

```
loan_percent_income 0
cb_person_default_on_file 0
cb_person_cred_hist_length 0
dtype: int64

Fungsi median_imputation: informasi count na setelah dilakukan imputasi:
person_age 0
person_income 0
person_home_ownership 0
person_emp_length 0
loan_intent 0
loan_grade 0
loan_amnt 0
loan_int_rate 0
loan_percent_income 0
cb_person_default_on_file 0
cb_person_cred_hist_length 0
dtype: int64

[18]: 1 # X_valid_imputation
2 X_Valid = median_imputation(X_valid, subset_data, False)

Fungsi median_imputation: parameter telah divalidasi.
Fungsi median_imputation: informasi count na sebelum dilakukan imputasi:
person_age 0
person_income 0
person_home_ownership 0
person_emp_length 80
loan_intent 0
loan_grade 0
loan_amnt 0
loan_int_rate 312
loan_percent_income 0
cb_person_default_on_file 0
cb_person_cred_hist_length 0
dtype: int64
```

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

localhost:8888/lab/tree/notebooks

Master Degree Class Detail Pacmann Student Pl... Models compatible... All Bookmarks

File Edit View Run Kernel Tabs Settings Help

data\_preprocessing.ipynb eda.ipynb data\_preparation.ipynb Python 3 (ipykernel)

Name Modified

- data\_preparation.ipynb yesterday
- data\_preprocessing.ipynb 16d ago
- eda.ipynb yesterday

Code

```
loan_percent_income 0
cb_person_default_on_file 0
cb_person_cred_hist_length 0
dtype: int64

Fungsi median_imputation: informasi count na setelah dilakukan imputasi:
person_age 0
person_income 0
person_home_ownership 0
person_emp_length 0
loan_intent 0
loan_grade 0
loan_amnt 0
loan_int_rate 0
loan_percent_income 0
cb_person_default_on_file 0
cb_person_cred_hist_length 0
dtype: int64

[18]: 1 # X_valid_imputation
2 X_Valid = median_imputation(X_valid, subset_data, False)

Fungsi median_imputation: parameter telah divalidasi.
Fungsi median_imputation: informasi count na sebelum dilakukan imputasi:
person_age 0
person_income 0
person_home_ownership 0
person_emp_length 80
loan_intent 0
loan_grade 0
loan_amnt 0
loan_int_rate 312
loan_percent_income 0
cb_person_default_on_file 0
cb_person_cred_hist_length 0
dtype: int64

Fungsi median_imputation: informasi count na setelah dilakukan imputasi:
person_age 0
person_income 0
person_home_ownership 0
person_emp_length 0
loan_intent 0
loan_grade 0
loan_amnt 0
```

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

The screenshot shows a Jupyter Notebook interface with three tabs open: 'data\_preprocessing.ipynb', 'eda.ipynb', and 'data\_preparation.ipynb'. The 'data\_preprocessing.ipynb' tab is active. In the code cell, there is a function definition for 'median\_imputation' and some sample data. The output cell shows the execution results for the 'loan\_grade' column.

```
loan_grade      0
loan_amnt       0
loan_int_rate   312
loan_percent_inc 0
cb_person_default_on_file 0
cb_person_cred_hist_length 0
dtype: int64

Fungsi median_imputation: informasi count na setelah dilakukan imputasi:
person_age      0
person_income    0
person_home_ownership 0
person_emp_length 0
loan_intent      0
loan_grade       0
loan_amnt        0
loan_int_rate    0
loan_percent_inc 0
cb_person_default_on_file 0
cb_person_cred_hist_length 0
dtype: int64
```

f. [10 poin] Encoding data kategorik

i.Buat fungsi untuk membuat encoder data kategorik

- Buat fungsi bernama `create_onehot_encoder()`
- Fungsi tersebut memiliki parameter bernama `categories` dan `path`
- Parameter `categories` harus bertipe list 1 dimensi
- Parameter `categories` berisi daftar kategorik yang akan dibuat encodernya
- Parameter `path` harus bertipe string
- Parameter `path` berisi lokasi pada disk komputer dimana encoder yang dibuat akan disimpan
- Buat percabangan untuk validasi parameter

a. Percabangan pertama

i.Melakukan komparasi antara parameter `categories` dan list

ii.Gunakan fungsi `isinstance()` untuk melakukan komparasi

- iii.Jika parameter categories tidak sama dengan list, maka raise runtimeerror dengan pesan “Fungsi create\_onehot\_encoder: parameter categories haruslah bertipe list, berisi kategori yang akan dibuat encodernya.”
- iv.Jika parameter categories sama dengan list, maka eksekusi kode diluar cakupan percabangan ini
- b. Percabangan kedua
- i.Melakukan komparasi antara parameter path dan str
  - ii.Gunakan fungsi isinstance() untuk melakukan komparasi
- iii.Jika parameter path tidak sama dengan str, maka raise runtimeerror dengan pesan “Fungsi create\_onehot\_encoder: parameter path haruslah bertipe string, berisi lokasi pada disk komputer dimana encoder akan disimpan.”
- iv.Jika parameter path sama dengan str, maka eksekusi kode diluar cakupan percabangan ini
- Buat instance one hot encoder dengan cara memanggil fungsi OneHotEncoder()
- a. Simpan instance tersebut pada variabel bernama ohe
- Melakukan fitting encoder
- a. Ubah tipe data pada parameter categories yang awalnya list 1 dimensi menjadi numpy array 2 dimensi dengan menggunakan fungsi array() dari numpy yang dichain dengan fungsi reshape()
- i.Gunakan data pada parameter categories sebagai argumen pada fungsi array()
  - ii.Berikan argument -1, 1 pada fungsi reshape()
- b. Gunakan hasil dari ubahan tipe data parameter categories sebagai argumen pada fungsi fit dari instance ohe
- Panggil fungsi serialize\_data() untuk menyimpan encoder yang baru saja difitting karena akan digunakan di masa depan
- a. Gunakan ohe dan data pada parameter path sebagai argumen pada fungsi serialize\_data()
- Lakukan print dengan pesan “Kategori yang telah dipelajari adalah “ diikuti dengan daftar kategori yang telah dipelajari oleh ohe
- a. Gunakan macro format (f) pada string yang akan diperintah pada fungsi print

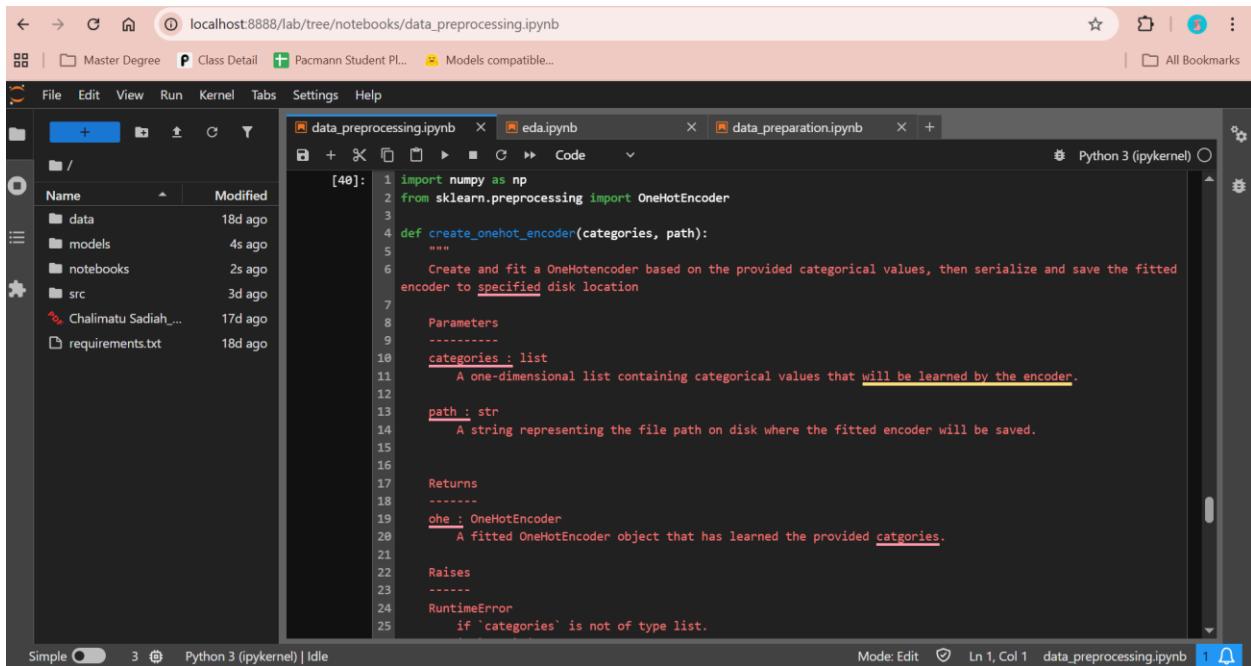
b. Dapatkan daftar kategori yang telah dipelajari dengan cara mengakses atribut categories\_index ke 0

i.Ubah ke list dengan menggunakan fungsi tolist() setelah mengakses index ke 0 dari data categories\_

ii.Gunakan hasil perubahan tersebut sebagai argumen pada fungsi print untuk ditampilkan beserta pesan sebelumnya

- Kembalikan ohe dari fungsi

### Lampirkan screenshot



The screenshot shows a Jupyter Notebook interface with three tabs: 'data\_preprocessing.ipynb' (active), 'eda.ipynb', and 'data\_preparation.ipynb'. The code cell contains the following Python code:

```
1 import numpy as np
2 from sklearn.preprocessing import OneHotEncoder
3
4 def create_onehot_encoder(categories, path):
5     """
6         Create and fit a OneHotEncoder based on the provided categorical values, then serialize and save the fitted
7         encoder to specified disk location
8
9     Parameters
10    -----
11        categories : list
12            A one-dimensional list containing categorical values that will be learned by the encoder.
13
14        path : str
15            A string representing the file path on disk where the fitted encoder will be saved.
16
17    Returns
18    -----
19        ohe : OneHotEncoder
20            A fitted OneHotEncoder object that has learned the provided categories.
21
22    Raises
23    -----
24        RuntimeError
25            if 'categories' is not of type list.
```

The screenshot shows a Jupyter Notebook interface with three tabs open: 'data\_preprocessing.ipynb', 'eda.ipynb', and 'data\_preparation.ipynb'. The 'data\_preprocessing.ipynb' tab is active, displaying Python code for creating a OneHotEncoder. The code includes validation for 'categories' (must be a list) and 'path' (must be a string). It creates a OneHotEncoder object with 'sparse\_output=False' and 'handle\_unknown='ignore''. The 'categories\_array' is reshaped to (-1, 1).

```
22     Raises
23     -----
24     RuntimeError
25         if 'categories' is not of type list.
26         if 'path' is not of type str.
27     """
28
29     # Parameter Validation
30     if not isinstance(categories, list):
31         raise RuntimeError(
32             "Fungsi create_onehot_encoder: parameter categories haruslah bertipe list, berisi kategori yang akan dibuat encodenya."
33         )
34
35     if not isinstance(path, str):
36         raise RuntimeError(
37             "Fungsi create_onehot_encoder: parameter path haruslah bertipe string, berisi lokasi pada disk komputer dimana encoder akan disimpan."
38         )
39
40
41     # Create Encoder
42     ohe = OneHotEncoder(sparse_output=False, handle_unknown='ignore')
43
44     categories_array = np.array(categories).reshape(-1, 1)
45
```

The screenshot shows the same Jupyter Notebook interface. The code has been completed with additional logic to fit the encoder to the data and serialize it to a file. A yellow circular badge in the bottom right corner indicates there are 5 notifications.

```
33     akan dibuat encodenya."
34
35     if not isinstance(path, str):
36         raise RuntimeError(
37             "Fungsi create_onehot_encoder: parameter path haruslah bertipe string, berisi lokasi pada disk komputer dimana encoder akan disimpan."
38         )
39
40
41     # Create Encoder
42     ohe = OneHotEncoder(sparse_output=False, handle_unknown='ignore')
43
44     categories_array = np.array(categories).reshape(-1, 1)
45
46     ohe.fit(categories_array)
47
48     u.serialize_data(ohe, path)
49
50     learned_categories = ohe.categories_[0].tolist()
51
52     print(f"Kategori yang telah dipelajari adalah {learned_categories}")
53
54     return ohe
55
```

ii. Jalankan fungsi pembuatan encoder

- Buat empat konstan variabel bertipe list dan bernama person\_home\_ownership, loan\_intent, loan\_grade, cb\_person\_default\_on\_file
  - Isi variabel tersebut dengan kategorik dari kolom masing-masing
  - Panggil fungsi create\_onehot\_encoder()
- a. Berikan variabel person\_home\_ownership dan path models/ohe\_home\_ownership.pkl sebagai argumen untuk parameter categories dan path secara berurutan
  - b. Simpan keluaran fungsi pada variabel bernama ohe\_home\_ownership
    - Panggil ulang fungsi create\_onehot\_encoder()
- a. Berikan variabel loan\_intent dan path models/ohe\_loan\_intent.pkl sebagai argumen untuk parameter categories dan path secara berurutan
  - b. Simpan keluaran fungsi pada variabel bernama ohe\_loan\_intent
    - Panggil ulang fungsi create\_onehot\_encoder()
- a. Berikan variabel loan\_grade dan path models/ohe\_loan\_grade.pkl sebagai argumen untuk parameter categories dan path secara berurutan
  - b. Simpan keluaran fungsi pada variabel bernama ohe\_loan\_grade
    - Panggil ulang fungsi create\_onehot\_encoder()
- a. Berikan variabel cb\_person\_default\_on\_file dan path models/ohe\_default\_on\_file.pkl sebagai argumen untuk parameter categories dan path secara berurutan
  - b. Simpan keluaran fungsi pada variabel bernama ohe\_default\_on\_file

**Lampirkan screenshot**

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

Master Degree Class Detail Pacmann Student Pl... Models compatible... All Bookmarks

File Edit View Run Kernel Tabs Settings Help

[41]:

```
1 # Create category List
2 person_home_ownership = X_train["person_home_ownership"].unique().tolist()
3 loan_intent = X_train["loan_intent"].unique().tolist()
4 loan_grade = X_train["loan_grade"].unique().tolist()
5 cb_person_default_on_file = X_train["cb_person_default_on_file"].unique().tolist()
```

[42]:

```
1 # Make encoder
2 ohe_home_ownership = create_onehot_encoder(
3     person_home_ownership,
4     "../models/ohe_home_ownership.pkl"
5 )
6
7 ohe_loan_intent = create_onehot_encoder(
8     loan_intent,
9     "../models/ohe_loan_intent.pkl"
10 )
11
12 ohe_loan_grade = create_onehot_encoder(
13     loan_grade,
14     "../models/ohe_loan_grade.pkl"
15 )
16
17 ohe_default_on_file = create_onehot_encoder(
18     cb_person_default_on_file,
19     "../models/ohe_default_on_file.pkl"
20 )
```

Python 3 (ipykernel)

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

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

Master Degree Class Detail Pacmann Student Pl... Models compatible... All Bookmarks

File Edit View Run Kernel Tabs Settings Help

[42]:

```
1 # Make encoder
2 ohe_home_ownership = create_onehot_encoder(
3     person_home_ownership,
4     "../models/ohe_home_ownership.pkl"
5 )
6
7 ohe_loan_intent = create_onehot_encoder(
8     loan_intent,
9     "../models/ohe_loan_intent.pkl"
10 )
11
12 ohe_loan_grade = create_onehot_encoder(
13     loan_grade,
14     "../models/ohe_loan_grade.pkl"
15 )
16
17 ohe_default_on_file = create_onehot_encoder(
18     cb_person_default_on_file,
19     "../models/ohe_default_on_file.pkl"
20 )
```

Kategori yang telah dipelajari adalah ['MORTGAGE', 'OTHER', 'OWN', 'RENT']  
Kategori yang telah dipelajari adalah ['DEBTCONSOLIDATION', 'EDUCATION', 'HOMEIMPROVEMENT', 'MEDICAL', 'PERSONAL', 'VENTURE']  
Kategori yang telah dipelajari adalah ['A', 'B', 'C', 'D', 'E', 'F', 'G']  
Kategori yang telah dipelajari adalah ['N', 'Y']

Python 3 (ipykernel)

Simple Python 3 (ipykernel) | Idle Mode: Command Ln 20, Col 2 data\_preprocessing.ipynb

iii. Buat fungsi untuk melakukan encoding data kategorik

- Buat fungsi bernama ohe\_transform()
- Fungsi tersebut memiliki parameter bernama dataset, subset, prefix dan ohe
- Parameter dataset harus bertipe dataframe
- Parameter dataset merupakan set data yang ingin dilakukan pengkodean
- Parameter subset harus bertipe string
- Parameter subset merupakan nama kolom yang terdapat pada data di parameter dataset
- Parameter prefix harus bertipe string
- Parameter prefix merupakan nama awalan yang akan disematkan pada kolom hasil pengkodean
- Parameter ohe harus bertipe OneHotEncoder dari sklearn.preprocessing
- Parameter ohe merupakan encoder yang sebelumnya telah dilatih oleh data kategorik khusus
- Buat percabangan guna validasi paramater

a. Percabangan pertama

i.Melakukan komparasi antara parameter dataset dan dataframe

ii.Gunakan fungsi isinstance() untuk melakukan komparasi

iii.Jika parameter dataset tidak sama dengan dataset, maka raise runtimeerror dengan pesan  
“Fungsi ohe\_transform: parameter dataset harus bertipe DataFrame!”

iv.Jika parameter dataset sama dengan dataframe, maka eksekusi kode diluar cakupan  
percabangan ini

b. Percabangan kedua

i.Melakukan komparasi antara parameter ohe dan OneHotEncoder

ii.Gunakan fungsi isinstance() untuk melakukan komparasi

- iii.Jika parameter ohe tidak sama dengan OneHotEncoder, maka raise runtimeerror dengan pesan “Fungsi ohe\_transform: parameter ohe harus bertipe OneHotEncoder!”
- iv.Jika parameter ohe sama dengan OneHotEncoder, maka eksekusi kode diluar cakupan percabangan ini
- b. Percabangan ketiga
- v.Melakukan komparasi antara parameter prefix dan str
- vi.Gunakan fungsi isinstance() untuk melakukan komparasi
- vii.Jika parameter prefix tidak sama dengan str, maka raise runtimeerror dengan pesan “Fungsi ohe\_transform: parameter prefix harus bertipe str!”
- viii.Jika parameter prefix sama dengan str, maka eksekusi kode diluar cakupan percabangan ini
- b. Percabangan keempat
- ix.Melakukan komparasi antara parameter subset dan str
- x.Gunakan fungsi isinstance() untuk melakukan komparasi
- xi.Jika parameter subset tidak sama dengan str, maka raise runtimeerror dengan pesan “Fungsi ohe\_transform: parameter subset harus bertipe str!”
- xii.Jika parameter subset sama dengan str, maka eksekusi kode diluar cakupan percabangan ini
- b. Percabangan kelima
- xiii.Melakukan pengecekan data pada parameter subset di daftar nama kolom pada parameter dataset
- xiv.Buat blok pengetesan kode program untuk melakukan hal tersebut
1. Pada bagian try, buat list dari semua nama kolom yang ada pada parameter dataset
    - a. Gunakan fungsi index untuk mendapatkan nama kolom yang diinginkan
    - b. Berikan parameter subset sebagai argumen pada fungsi index
  2. Pada bagian except
    - a. Raise runtimeerror dengan pesan “Fungsi ohe\_transform: parameter subset string namun data tidak ditemukan dalam daftar kolom yang terdapat pada parameter dataset.”
      - Print pesan “Fungsi ohe\_transform: parameter telah divalidasi.”

- Buat duplikat dari data pada parameter dataset dan simpan duplikatnya pada variabel bernama dataset (direplace)
  - Print pesan yang menampilkan daftar nama kolom sebelum dilakukan pengkodean
    - a. Gunakan macro format (f) pada string yang akan diprint pada fungsi print
    - b. Pesannya adalah “Fungsi ohe\_transform: daftar nama kolom sebelum dilakukan pengkodean adalah [daftar nama kolom].”
    - c. Akses data nama kolom pada dataset dengan cara mengakses atribut columns
    - d. Ubah daftar nama kolom ke tipe data list dengan fungsi list
    - e. Gunakan daftar nama kolom tersebut sebagai argumen pada fungsi print
  - Buat satu variabel bernama col\_names untuk menyimpan nama kolom yang telah dikodekan
    - a. Nama kolom dibuat dengan menggunakan list comprehension
    - b. Buat perulangan dalam list comprehension dari daftar kategori yang ada pada parameter ohe
- i.Data daftar kategori dapat diakses melalui atribut categories\_index ke 0
- ii.Ubah tipe data daftar kategori tersebut ke list dengan menggunakan fungsi tolist()
- iii.Simpan nama kategori diiap iterasi perulangan ke variabel bernama col\_name
- iv.Dalam tiap iterasi perulangan, gabungkan data pada parameter prefix, karakter garis bawah, serta data pada variabel col\_name
- Proses pengkodean:
    - a. Buat satu variabel bernama encoded
    - b. Variabel tersebut diisi oleh sebuah dataframe
    - c. Data pada dataframe tersebut didapatkan dari hasil transformasi ohe
    - d. Proses transformasi ohe:
- i.Panggil fungsi transform dari variabel ohe
- ii.Ubah subset ke tipe data list dengan cara menambahkan kurung siku pada kedua sisinya
- iii.Gunakan list subset tersebut sebagai selektor dari dataset

iv. Hasil seleksi dataset dijadikan argumen dari fungsi transform ohe

v. Ubah tipe data hasil transform ohe ke array dengan cara chaining fungsi toarray()

vi. Gunakan array hasil transformasi tersebut sebagai argumen pertama pada pembuatan dataframe di poin c

e. Gunakan col\_names sebagai argumen dari parameter columns pada pembuatan dataframe di poin c

f. Set index dari dataframe yang dibuat di poin c dengan cara

i. Ambil index dari dataset dengan mengakses atribut indexnya

ii. Gunakan data tersebut sebagai argumen dari parameter index pada pembuatan dataframe di poin c

- Proses penyatuan hasil pengkodean dengan data sebelum pengkodean

a. Panggil fungsi concat() dari pandas

b. Untuk argumen pertama, buat list yang berisi data sebelum pengkodean dan data setelah pengkodean

c. Set parameter axis menjadi satu (1)

d. Simpan hasil dari fungsi concat() ke variabel bernama dataset (direplace)

- Proses penghapusan kolom yang tidak diperlukan

a. Menghapus kolom dari dataframe yang telah dikodekan

b. Panggil fungsi drop dari dataframe

c. Untuk parameter columns pada fungsi drop, berikan list dari subset sebagai argumen

i. Ubah tipe data subset ke list dapat dilakukan dengan cara memberikan kurung siku pada kedua sisinya

d. Untuk parameter inplace pada fungsi drop, berikan nilai True

- Print pesan yang menandakan bahwa proses pengkodean telah berhasil

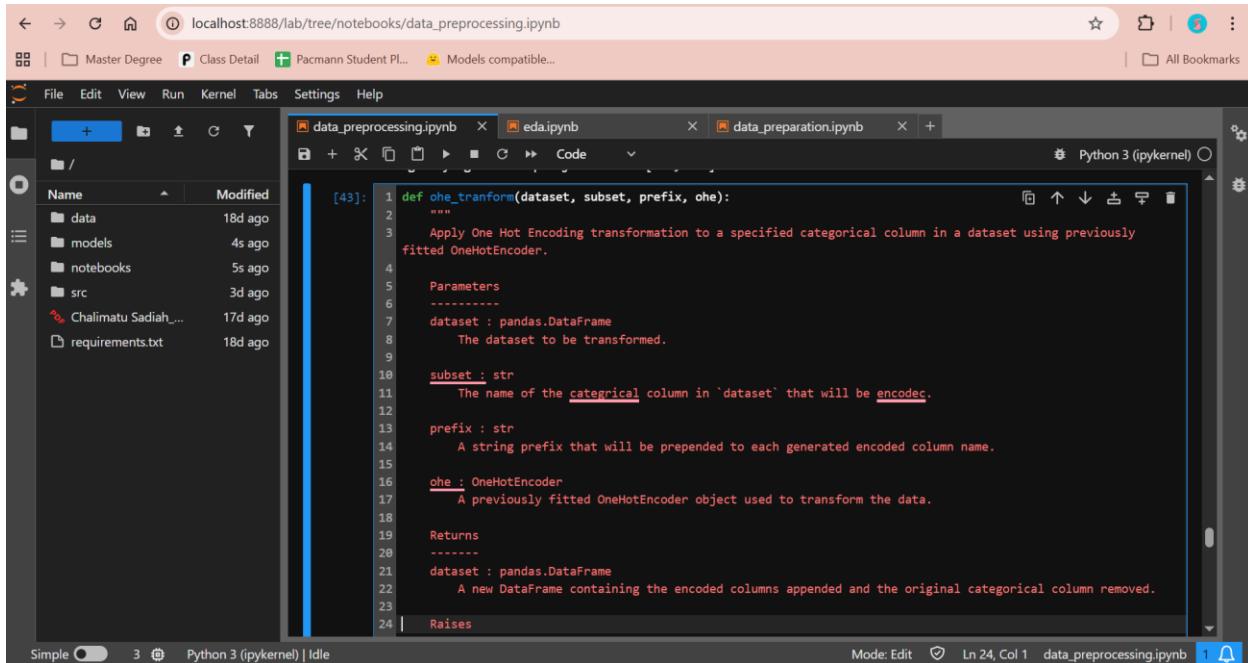
a. Gunakan macro format (f) pada string yang akan diprint pada fungsi print

b. Pesannya adalah "Fungsi ohe\_transform: daftar nama kolom setelah dilakukan pengkodean adalah [daftar nama kolom]."

c. Akses data nama kolom pada dataset dengan cara mengakses atribut columns

- d. Ubah daftar nama kolom ke tipe data list dengan fungsi list
- e. Gunakan daftar nama kolom tersebut sebagai argumen pada fungsi print
- f. Kembalikan dataset dari fungsi

### Lampirkan screenshot

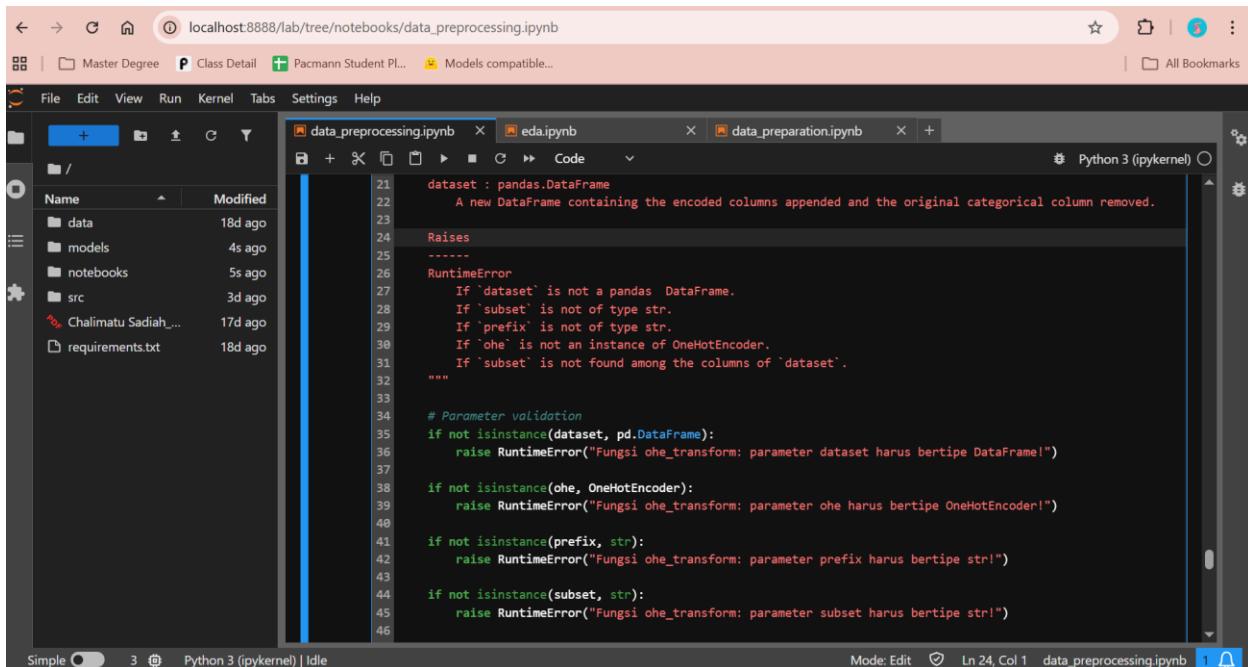


The screenshot shows a Jupyter Notebook interface with three tabs: 'data\_preprocessing.ipynb' (active), 'eda.ipynb', and 'data\_preparation.ipynb'. The code editor displays the following Python function:

```

1 def ohe_transform(dataset, subset, prefix, ohe):
2     """
3         Apply One Hot Encoding transformation to a specified categorical column in a dataset using previously
4         fitted OneHotEncoder.
5
6     Parameters
7     -----
8         dataset : pandas.DataFrame
9             The dataset to be transformed.
10
11        subset : str
12            The name of the categorical column in `dataset` that will be encoded.
13
14        prefix : str
15            A string prefix that will be prepended to each generated encoded column name.
16
17        ohe : OneHotEncoder
18            A previously fitted OneHotEncoder object used to transform the data.
19
20    Returns
21    -----
22        dataset : pandas.DataFrame
23            A new DataFrame containing the encoded columns appended and the original categorical column removed.
24
25    Raises
26    -----
27        RuntimeError
28            If `dataset` is not a pandas DataFrame.
29            If `subset` is not of type str.
30            If `prefix` is not of type str.
31            If `ohe` is not an instance of OneHotEncoder.
32            If `subset` is not found among the columns of `dataset`.
33
34    # Parameter validation
35    if not isinstance(dataset, pd.DataFrame):
36        raise RuntimeError("Fungsi ohe_transform: parameter dataset harus bertipe DataFrame!")
37
38    if not isinstance(ohe, OneHotEncoder):
39        raise RuntimeError("Fungsi ohe_transform: parameter ohe harus bertipe OneHotEncoder!")
40
41    if not isinstance(prefix, str):
42        raise RuntimeError("Fungsi ohe_transform: parameter prefix harus bertipe str!")
43
44    if not isinstance(subset, str):
45        raise RuntimeError("Fungsi ohe_transform: parameter subset harus bertipe str!")

```



The screenshot continues the Jupyter Notebook interface from the previous one. The code editor now shows the continuation of the function definition, specifically the parameter validation and runtime error handling logic.

```

43     if not isinstance(subset, str):
44         raise RuntimeError("Fungsi ohe_transform: parameter subset harus bertipe str!")
45
46     try:
47         list(dataset.columns).index(subset)
48     except:
49         raise RuntimeError(
50             "Fungsi ohe_transform: parameter subset string namun data tidak ditemukan dalam daftar kolom yang
51             terdapat pada parameter dataset."
52         )
53
54     print("Fungsi ohe_transform: parameter telah divalidasi.")
55
56     # Copy Data
57     dataset = dataset.copy()
58
59     print(f"Fungsi ohe_transform: daftar nama kolom sebelum dilakukan pengkodean adalah
60 {list(dataset.columns)}")
61
62     # Create New Column Names
63     col_names = [
64         f"{prefix}_{col_name}"
65         for col_name in ohe.categories_[0].tolist()
66     ]

```

```

67
68     # Transform
69     encoded_array = ohe.transform(dataset[[subset]])
70
71     encoded = pd.DataFrame(
72         encoded_array,
73         columns=col_names,
74         index=dataset.index
75     )
76
77     # Concatenate
78     dataset = pd.concat([dataset, encoded], axis=1)
79
80     # Drop original column
81     dataset.drop(columns=[subset], inplace=True)
82
83     print(f"Fungsi ohe_transform: daftar nama kolom setelah dilakukan pengkodean adalah
84 {list(dataset.columns)}")
85
86     return dataset

```

iv. Jalankan fungsi untuk melakukan encoding data kategorik

- Untuk X\_train

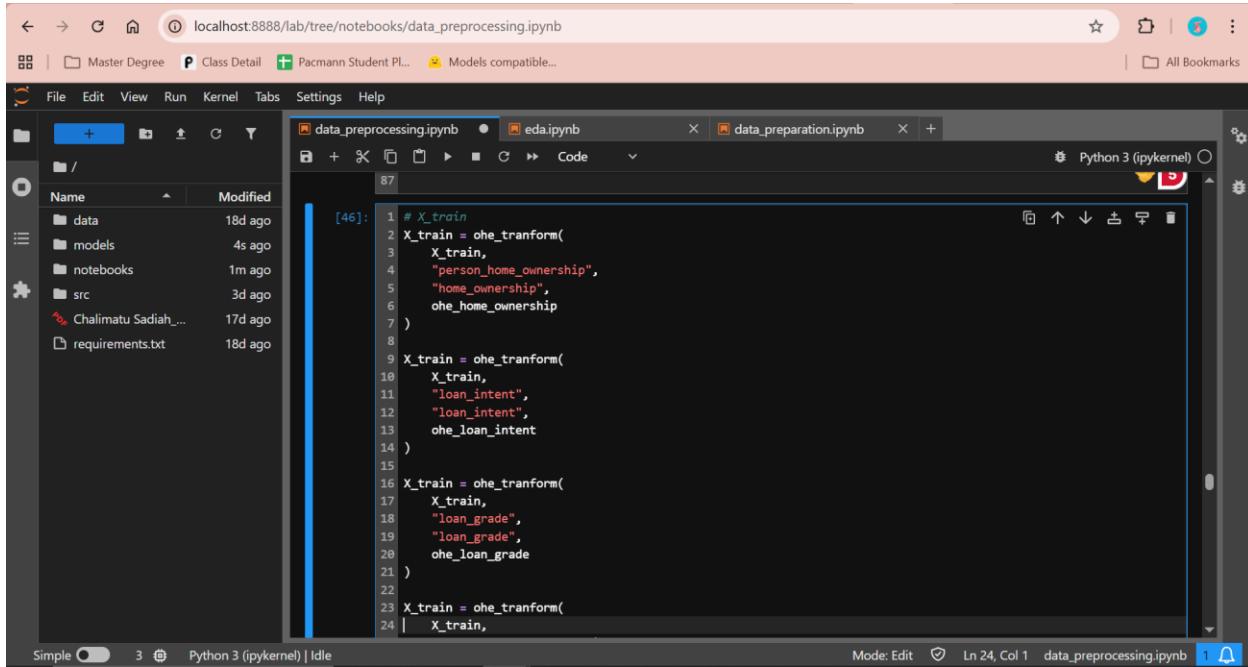
a. Untuk kolom person\_home\_ownership

i. Panggil fungsi ohe\_transform()

- ii.Berikan X\_train sebagai argumen pertama
  - iii.Berikan string “person\_home\_ownership” sebagai argumen kedua
  - iv.Berikan string “home\_ownership” sebagai argumen ketiga
  - v.Berikan OneHotEncoder ohe\_home\_ownership sebagai argumen keempat
  - vi.Simpan hasil transform ke variabel bernama X\_train
- b. Untuk kolom loan\_intent
    - i.Panggil fungsi ohe\_transform()
    - ii.Berikan X\_train sebagai argumen pertama
    - iii.Berikan string “loan\_intent” sebagai argumen kedua
    - iv.Berikan string “loan\_intent” sebagai argumen ketiga
    - v.Berikan OneHotEncoder ohe\_loan\_intent sebagai argumen keempat
    - vi.Simpan hasil transform ke variabel bernama X\_train
  - c. Untuk kolom loan\_grade
    - i.Panggil fungsi ohe\_transform()
    - ii.Berikan X\_train sebagai argumen pertama
    - iii.Berikan string “loan\_grade” sebagai argumen kedua
    - iv.Berikan string “loan\_grade” sebagai argumen ketiga
    - v.Berikan OneHotEncoder ohe\_loan\_grade sebagai argumen keempat
    - vi.Simpan hasil transform ke variabel bernama X\_train
  - d. Untuk kolom cb\_person\_default\_on\_file
    - i.Panggil fungsi ohe\_transform()
    - ii.Berikan X\_train sebagai argumen pertama
    - iii.Berikan string “cb\_person\_default\_on\_file” sebagai argumen kedua
    - iv.Berikan string “default\_onfile” sebagai argumen ketiga
    - v.Berikan OneHotEncoder ohe\_defaul\_on\_file sebagai argumen keempat
    - vi.Simpan hasil transform ke variabel bernama X\_train

- Lakukan hal yang sama dengan poin “Untuk X\_train” pada X\_test dan juga X\_valid, ubah argumen pertama dari X\_train ke X\_test dan X\_valid

### Lampirkan screenshot



The screenshot shows a Jupyter Notebook interface with three tabs: 'data\_preprocessing.ipynb' (active), 'eda.ipynb', and 'data\_preparation.ipynb'. The code in the active tab is as follows:

```
[46]: 
1 # X_train
2 X_train = ohe_transform(
3     X_train,
4     "person_home_ownership",
5     "home_ownership",
6     ohe_home_ownership
7 )
8
9 X_train = ohe_transform(
10    X_train,
11    "loan_intent",
12    "loan_intent",
13    ohe_loan_intent
14 )
15
16 X_train = ohe_transform(
17    X_train,
18    "loan_grade",
19    "loan_grade",
20    ohe_loan_grade
21 )
22
23 X_train = ohe_transform(
24     X_train,
```

Below the code, there is a warning message from the 'ohe\_transform' function:

Fungsi ohe\_transform: parameter telah divalidasi.  
Fungsi ohe\_transform: daftar nama kolom sebelum dilakukan pengkodean adalah ['person\_age', 'person\_income', 'person\_home\_ownership', 'person\_emp\_length', 'loan\_intent', 'loan\_grade', 'loan\_amnt', 'loan\_int\_rate', 'loan\_percent\_inc\_ome', 'cb\_person\_default\_on\_file', 'cb\_person\_cred\_hist\_length'].  
/home/chalimasadiah/CHALIMA\_MLPROCESS/.CHALIMA\_VENV/lib/python3.12/site-packages/sklearn/utils/validation.py:2684:  
UserWarning: X has feature names, but OneHotEncoder was fitted without feature names  
warnings.warn(  
/home/chalimasadiah/CHALIMA\_MLPROCESS/.CHALIMA\_VENV/lib/python3.12/site-packages/sklearn/utils/validation.py:2684:  
UserWarning: X has feature names, but OneHotEncoder was fitted without feature names  
warnings.warn(  
Fungsi ohe\_transform: daftar nama kolom setelah dilakukan pengkodean adalah ['person\_age', 'person\_income', 'person\_emp\_length', 'loan\_intent', 'loan\_grade', 'loan\_amnt', 'loan\_int\_rate', 'loan\_percent\_income', 'cb\_person\_default\_on\_file', 'cb\_person\_cred\_hist\_length', 'home\_ownership\_MORTGAGE', 'home\_ownership\_OTHER', 'home\_ownership\_OWN', 'home\_ownership\_RENT'].  
Fungsi ohe\_transform: parameter telah divalidasi.  
Fungsi ohe\_transform: daftar nama kolom sebelum dilakukan pengkodean adalah ['person\_age', 'person\_income', 'person\_emp\_length', 'loan\_intent', 'loan\_grade', 'loan\_amnt', 'loan\_int\_rate', 'loan\_percent\_income', 'cb\_person\_default\_on\_file', 'cb\_person\_cred\_hist\_length', 'home\_ownership\_MORTGAGE', 'home\_ownership\_OTHER', 'home\_ownership\_OWN', 'home\_ownership\_RENT']

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

Master Degree Class Detail Pacmann Student Pl... Models compatible... All Bookmarks

File Edit View Run Kernel Tabs Settings Help

data\_preprocessing.ipynb eda.ipynb data\_preparation.ipynb Python 3 (ipykernel)

```
[47]: 1 # X_test
2 X_test = ohe_transform(
3     X_test,
4     "person_home_ownership",
5     "home_ownership",
6     ohe_home_ownership
7 )
8
9 X_test = ohe_transform(
10    X_test,
11    "loan_intent",
12    "loan_intent",
13    ohe_loan_intent
14 )
15
16 X_test = ohe_transform(
17    X_test,
18    "loan_grade",
19    "loan_grade",
20    ohe_loan_grade
21 )
22
23 X_test = ohe_transform(
24    X_test,
25    "cb_person_default_on_file",
26    "cb_person_default_on_file"
27 )
28
```

Simple Python 3 (ipykernel) | Idle Mode: Edit Ln 24, Col 1 data\_preprocessing.ipynb

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

Master Degree Class Detail Pacmann Student Pl... Models compatible... All Bookmarks

File Edit View Run Kernel Tabs Settings Help

data\_preprocessing.ipynb eda.ipynb data\_preparation.ipynb Python 3 (ipykernel)

```
22
23 X_test = ohe_transform(
24     X_test,
25     "cb_person_default_on_file",
26     "default_onfile",
27     ohe_default_onfile
28 )
Fungsi ohe_transform: parameter telah divalidasi.
Fungsi ohe_transform: daftar nama kolom sebelum dilakukan pengkodean adalah ['person_age', 'person_income', 'person_home_ownership', 'person_emp_length', 'loan_intent', 'loan_grade', 'loan_amnt', 'loan_int_rate', 'loan_percent_income', 'cb_person_default_on_file', 'cb_person_cred_hist_length'].
Fungsi ohe_transform: daftar nama kolom setelah dilakukan pengkodean adalah ['person_age', 'person_income', 'person_emp_length', 'loan_intent', 'loan_grade', 'loan_amnt', 'loan_int_rate', 'loan_percent_income', 'cb_person_default_on_file', 'cb_person_cred_hist_length', 'home_ownership_MORTGAGE', 'home_ownership_OTHER', 'home_ownership_OWN', 'home_ownership_RENT'].
Fungsi ohe_transform: parameter telah divalidasi.
Fungsi ohe_transform: daftar nama kolom sebelum dilakukan pengkodean adalah ['person_age', 'person_income', 'person_emp_length', 'loan_intent', 'loan_grade', 'loan_amnt', 'loan_int_rate', 'loan_percent_income', 'cb_person_default_on_file', 'cb_person_cred_hist_length', 'home_ownership_MORTGAGE', 'home_ownership_OTHER', 'home_ownership_OWN', 'home_ownership_RENT'].
Fungsi ohe_transform: daftar nama kolom setelah dilakukan pengkodean adalah ['person_age', 'person_income', 'person_emp_length', 'loan_grade', 'loan_amnt', 'loan_int_rate', 'loan_percent_income', 'cb_person_default_on_file', 'cb_person_cred_hist_length', 'home_ownership_MORTGAGE', 'home_ownership_OTHER', 'home_ownership_OWN', 'home_ownership_RENT', 'loan_intent_DEBTCONSOLIDATION', 'loan_intent_EDUCATION', 'loan_intent_HOMEIMPROVEMENT', 'loan_intent_MEDICAL', 'loan_intent_PERSONAL', 'loan_intent_VENTURE'].
Fungsi ohe_transform: parameter telah divalidasi.
Fungsi ohe_transform: daftar nama kolom sebelum dilakukan pengkodean adalah ['person_age', 'person_income', 'person
```

Simple Python 3 (ipykernel) | Idle Mode: Edit Ln 24, Col 1 data\_preprocessing.ipynb

The screenshot shows a Jupyter Notebook interface with three tabs: 'data\_preprocessing.ipynb' (active), 'eda.ipynb', and 'data\_preparation.ipynb'. The code cell contains the following Python code:

```
# [48]:  
1 # X_valid  
2 X_valid = ohe_transform(  
3     X_valid,  
4     "person_home_ownership",  
5     "home_ownership",  
6     ohe_home_ownership  
7 )  
8  
9 X_valid = ohe_transform(  
10    X_valid,  
11    "loan_intent",  
12    "loan_intent",  
13    ohe_loan_intent  
14 )  
15  
16 X_valid = ohe_transform(  
17    X_valid,  
18    "loan_grade",  
19    "loan_grade",  
20    ohe_loan_grade  
21 )  
22  
23 X_valid = ohe_transform(  
24    X_valid,  
25    "cb_person_default_on_file",  
26 )
```

The screenshot shows a Jupyter Notebook interface with three tabs: 'data\_preprocessing.ipynb' (active), 'eda.ipynb', and 'data\_preparation.ipynb'. The code cell contains the following Python code:

```
22  
23 X_valid = ohe_transform(  
24     X_valid,  
25     "cb_person_default_on_file",  
26     "default_onfile",  
27     ohe_default_on_file  
28 )
```

Below the code, several error messages are displayed:

- Fungsi ohe\_transform: parameter telah divalidasi.
- Fungsi ohe\_transform: daftar nama kolom sebelum dilakukan pengkodean adalah ['person\_age', 'person\_income', 'person\_home\_ownership', 'person\_emp\_length', 'loan\_intent', 'loan\_grade', 'loan\_amnt', 'loan\_int\_rate', 'loan\_percent\_inc\_ome', 'cb\_person\_default\_on\_file', 'cb\_person\_cred\_hist\_length'].
- Fungsi ohe\_transform: daftar nama kolom setelah dilakukan pengkodean adalah ['person\_age', 'person\_income', 'person\_emp\_length', 'loan\_intent', 'loan\_grade', 'loan\_amnt', 'loan\_int\_rate', 'loan\_percent\_income', 'cb\_person\_default\_on\_file', 'cb\_person\_cred\_hist\_length', 'home\_ownership\_MORTGAGE', 'home\_ownership\_OTHER', 'home\_ownership\_OWN', 'home\_ownership\_RENT'].
- Fungsi ohe\_transform: parameter telah divalidasi.
- Fungsi ohe\_transform: daftar nama kolom sebelum dilakukan pengkodean adalah ['person\_age', 'person\_income', 'person\_emp\_length', 'loan\_intent', 'loan\_grade', 'loan\_amnt', 'loan\_int\_rate', 'loan\_percent\_income', 'cb\_person\_default\_on\_file', 'cb\_person\_cred\_hist\_length', 'home\_ownership\_MORTGAGE', 'home\_ownership\_OTHER', 'home\_ownership\_OWN', 'home\_ownership\_RENT'].
- Fungsi ohe\_transform: daftar nama kolom setelah dilakukan pengkodean adalah ['person\_age', 'person\_income', 'person\_emp\_length', 'loan\_grade', 'loan\_amnt', 'loan\_int\_rate', 'loan\_percent\_income', 'cb\_person\_default\_on\_file', 'cb\_person\_cred\_hist\_length', 'home\_ownership\_MORTGAGE', 'home\_ownership\_OTHER', 'home\_ownership\_OWN', 'home\_ownership\_RENT', 'loan\_intent\_DEBTCONSOLIDATION', 'loan\_intent\_EDUCATION', 'loan\_intent\_HOMEIMPROVEMENT', 'loan\_intent\_MEDICAL', 'loan\_intent\_PERSONAL', 'loan\_intent\_VENTURE'].
- Fungsi ohe\_transform: parameter telah divalidasi.
- Fungsi ohe\_transform: daftar nama kolom sebelum dilakukan pengkodean adalah ['person\_age', 'person\_income', 'person

g. [1 poin] Serialize set data yang telah diencode

i.Serialize X data

- Panggil fungsi serialze\_data

- Berikan dataframe X\_train sebagai argumen pertama
  - Berikan string data/processed/X\_train\_prep.pkl sebagai argumen kedua
  - Ulangi 3 poin diatas untuk X\_test dan X\_valid
- a. Ubah argumen pertama ke X\_test dan X\_valid
  - b. Ubah nama file dalam argumen kedua dengan X\_test\_prep.pkl dan X\_valid\_prep.pkl
  - c. Lokasi folder dalam string untuk argumen kedua tetap sama

### Lampirkan screenshot

```

# Serialize X_Train
u.serialize_data(
    X_train,
    "../data/processed/X_train_prep.pkl"
)

# Serialize X_test
u.serialize_data(
    X_test,
    "../data/processed/X_test_prep.pkl"
)

# Serialize X_valid
u.serialize_data(
    X_valid,
    "../data/processed/X_valid_prep.pkl"
)

# Serialize y_train
u.serialize_data(
    y_train,
    "../data/processed/y_train_prep.pkl"
)

```

### ii. Serialize y data

- Panggil fungsi serialize\_data
- Berikan dataframe y\_train sebagai argumen pertama
- Berikan string data/processed/y\_train\_prep.pkl sebagai argumen kedua

### Lampirkan screenshot

The screenshot shows a Jupyter Notebook interface with the URL [localhost:8888/lab/tree/notebooks/data\\_preprocessing.ipynb](http://localhost:8888/lab/tree/notebooks/data_preprocessing.ipynb). The left sidebar shows a file tree with a folder named 'processed' containing four files: X\_test\_prep.pkl, X\_train\_prep.pkl, X\_valid\_prep.pkl, and y\_train\_prep.pkl, all modified 6s ago. The main notebook area displays the following Python code:

```
[49]: 1 # Serialize X_Train
2 u.serialize_data(
3     X_train,
4     "../data/processed/X_train_prep.pkl"
5 )
6
7 # Serialize X_test
8 u.serialize_data(
9     X_test,
10    "../data/processed/X_test_prep.pkl"
11 )
12
13 # Serialize X_valid
14 u.serialize_data(
15     X_valid,
16     "../data/processed/X_valid_prep.pkl"
17 )
18
19 # Serialize y_train
20 u.serialize_data(
21     y_train,
22     "../data/processed/y_train_prep.pkl"
23 )
```

### 3. [10 poin] Feature Engineering

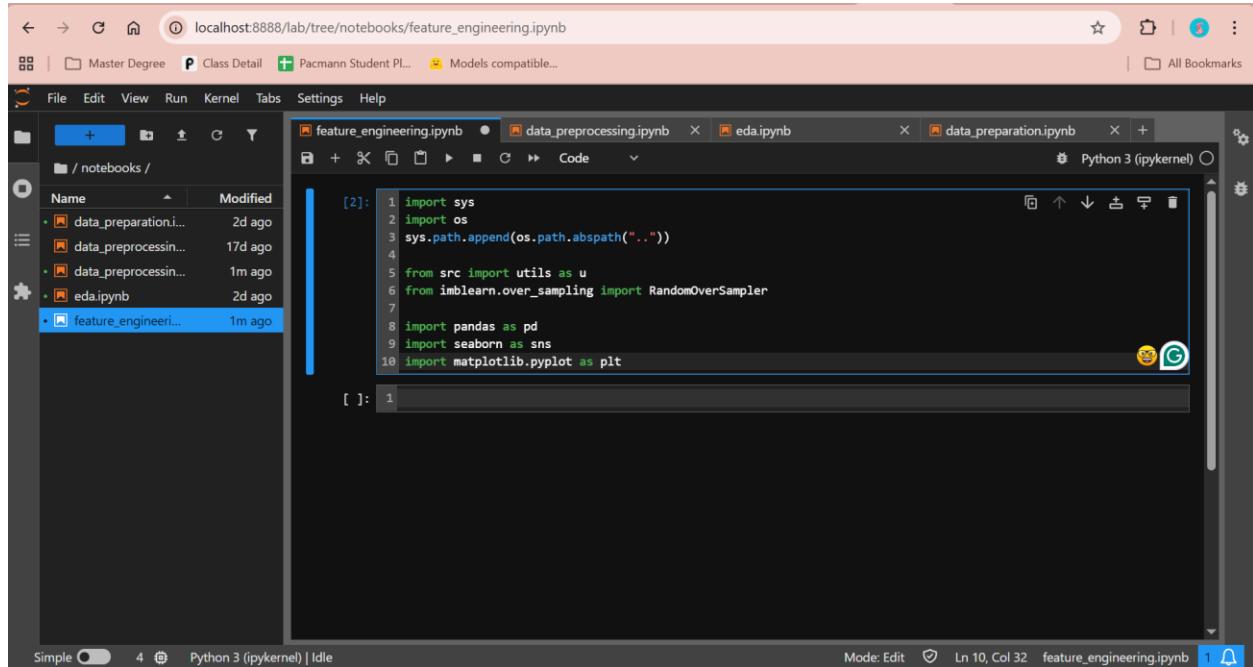
- [1 poin] Buat satu file bernama feature\_engineering.ipynb

Lampirkan screenshot

The screenshot shows a Jupyter Notebook interface with the URL [localhost:8888/lab/tree/notebooks/feature\\_engineering.ipynb](http://localhost:8888/lab/tree/notebooks/feature_engineering.ipynb). The left sidebar shows a file tree with a folder named 'notebooks' containing five files: data\_preparation.ipynb (2d ago), data\_preprocessing.ipynb (17d ago), eda.ipynb (2d ago), and feature\_engineering.ipynb (now). The main notebook area is currently empty, showing a single cell with the number 1.

b. [1 poin] Import utils dari src dan import RandomOverSampler dari imblearn.over\_sampling

Lampirkan screenshot



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

- URL:** localhost:8888/lab/tree/notebooks/feature\_engineering.ipynb
- File Explorer:** Shows a list of notebooks in the current directory:
  - data\_preparation.ipynb (modified 2d ago)
  - data\_preprocessing.ipynb (modified 17d ago)
  - data\_preprocessing.ipynb (modified 1m ago)
  - eda.ipynb (modified 2d ago)
  - feature\_engineering.ipynb (modified 1m ago)
- Code Cell (Cell 2):**

```
1 import sys
2 import os
3 sys.path.append(os.path.abspath(".."))
4
5 from src import utils as u
6 from imblearn.over_sampling import RandomOverSampler
7
8 import pandas as pd
9 import seaborn as sns
10 import matplotlib.pyplot as plt
```
- Code Cell (Cell 1):**

```
[ ]: 1
```
- Kernel:** Python 3 (ipykernel)
- Status Bar:** Mode: Edit, Line 10, Col 32, feature\_engineering.ipynb

c. [1 poin] Muat data train set yang telah dilakukan preprocessing

- Panggil fungsi deserialize\_data() dari utils
- Berikan argumen berupa lokasi berkas X\_train\_prep.pkl
- Simpan keluaran fungsi ke X\_train\_prep
- Panggil ulang fungsi deserialize\_data() dari utils
- Berikan argumen berupa lokasi berkas y\_train\_prep.pkl
- Simpan keluaran fungsi ke y\_train\_prep

Lampirkan screenshot

A screenshot of a Jupyter Notebook interface. The top navigation bar shows the URL as `localhost:8888/lab/tree/notebooks/feature_engineering.ipynb`. The left sidebar displays a file tree under the path `/notebooks/`, listing several notebooks: `data_preparation.ipynb`, `data_preprocessing.ipynb`, `eda.ipynb`, and `feature_engineering.ipynb`. The main workspace contains a code cell (cell 3) with the following Python code:

```
# Load X_train
X_train_prep = u.deserialize_data(
    "../data/processed/X_train_prep.pkl"
)
# Load y_train
y_train_prep = u.deserialize_data(
    "../data/processed/y_train_prep.pkl"
)
```

d. [1 poin] Buat instance RandomOverSampler()

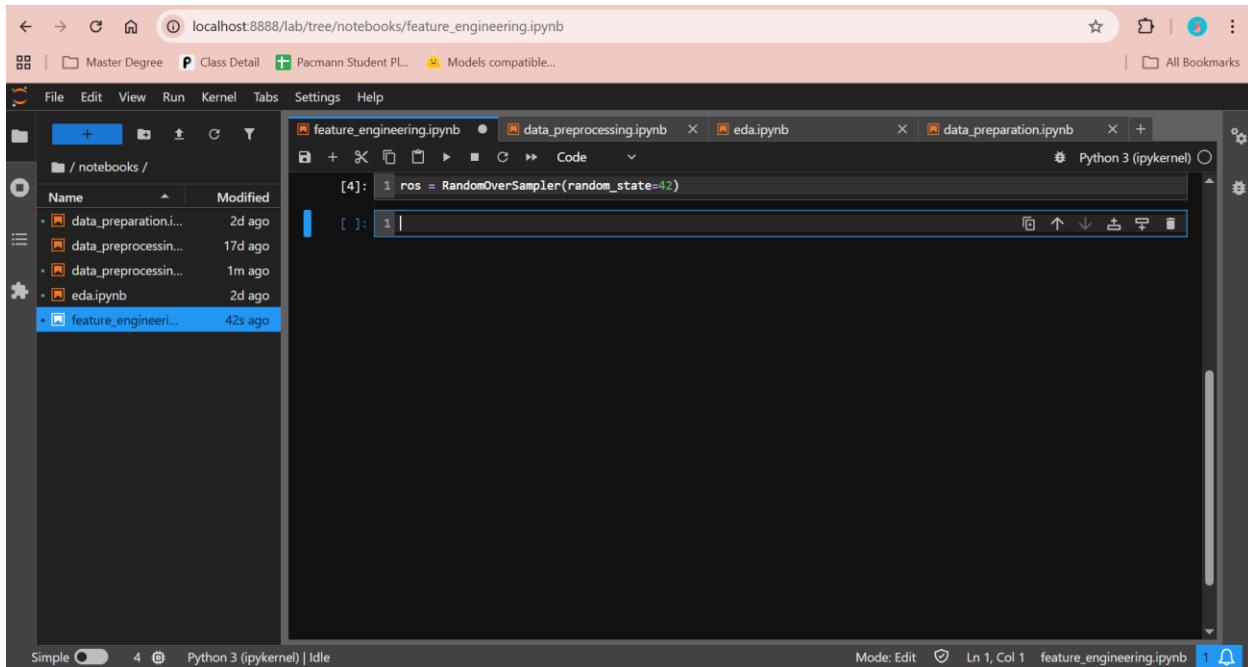
#### Lampirkan screenshot

A screenshot of a Jupyter Notebook interface. The top navigation bar shows the URL as `localhost:8888/lab/tree/notebooks/feature_engineering.ipynb`. The left sidebar displays a file tree under the path `/notebooks/`, listing several notebooks: `data_preparation.ipynb`, `data_preprocessing.ipynb`, `eda.ipynb`, and `feature_engineering.ipynb`. The main workspace contains a code cell (cell 4) with the following Python code:

```
ros = RandomOverSampler(random_state=42)
```

e. [1 poin] Simpan instance tersebut dalam variabel bernama ros

## Lampirkan screenshot



The screenshot shows a Jupyter Notebook interface running on localhost:8888. The left sidebar displays a file tree under 'notebooks/' with several notebooks listed: 'data\_preparation.ipynb' (modified 2d ago), 'data\_preprocessing.ipynb' (modified 17d ago), 'data\_preprocessing.ipynb' (modified 1m ago), 'eda.ipynb' (modified 2d ago), and 'feature\_engineering.ipynb' (modified 42s ago). The main area contains a code cell [4]:

```
[4]: 1 ros = RandomOverSampler(random_state=42)
```

The status bar at the bottom indicates 'Mode: Edit' and 'Ln 1, Col 1 feature\_engineering.ipynb'. There are also icons for Python 3 (ipykernel) and Idle.

f. [1 poin] Buat grafik tentang kondisi label sebelum dilakukan oversampling

- i. Gunakan fungsi countplot dari seaborn
- ii. Berikan `y_train_prep` yang telah diubah ke dataframe sebagai argumen untuk parameter data
- iii. Berikan string `loan_status` sebagai argumen untuk parameter x dan hue

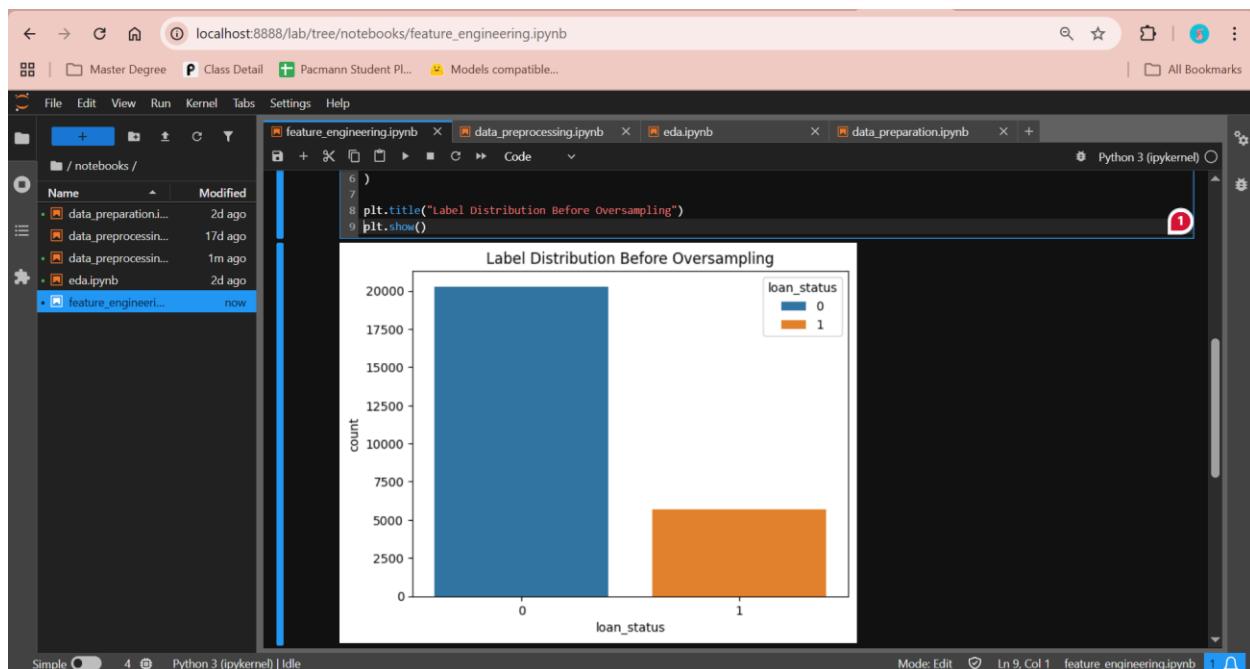
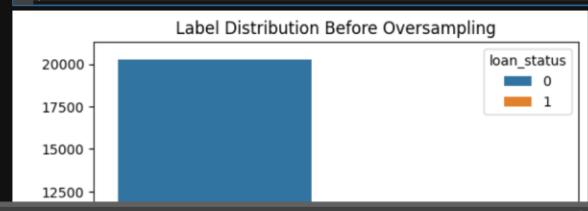
## Lampirkan screenshot

Screenshot of Jupyter Notebook showing code execution and a histogram.

Code in cell [6]:

```
1 # Plot
2 sns.countplot(
3     data=y_train_df,
4     x="loan_status",
5     hue="loan_status"
6 )
7 plt.title("Label Distribution Before Oversampling")
8 plt.show()
```

The histogram displays the count of 'loan\_status' values (0 and 1) before oversampling. The x-axis is labeled 'loan\_status' with categories 0 and 1. The y-axis is labeled 'count' ranging from 0 to 20,000. Category 0 has a count of approximately 20,000, and category 1 has a count of approximately 5,000.



g. [2 poin] Lakukan oversampling data tersebut

- i. Panggil fungsi fit\_resample dari variabel ros
- ii. Berikan X\_train\_prep sebagai argumen pertama

iii. Berikan `y_train_prep` sebagai argumen kedua

iv. Simpan keluaran dari fungsi `fit_resample()` ke dua variabel bernama `X_train_ros` dan `y_train_ros`

Lampirkan screenshot

The screenshot shows a Jupyter Notebook interface. On the left, there's a file tree with several notebooks listed. In the center, a code cell at index [7] contains Python code for oversampling:

```
# Oversampling
x_train_ros, y_train_ros = ros.fit_resample(
    x_train_prep,
    y_train_prep
)
print(x_train_prep.shape)
print(x_train_ros.shape)
```

The output of the code shows the shapes of the datasets:

```
(25968, 26)
(40582, 26)
```

At the top of the notebook, there's a countplot titled "loan\_status" with two bars: one for value 0 (white) and one for value 1 (orange). The x-axis is labeled "loan\_status" and has ticks at 0 and 1.

h. [1 poin] Buat grafik tentang kondisi label setelah dilakukan oversampling

i. Gunakan fungsi `countplot` dari `seaborn`

ii. Berikan `y_train_ros` yang telah diubah ke dataframe sebagai argumen untuk parameter `data`

iii. Berikan string `loan_status` sebagai argumen untuk parameter `x` dan `hue`

Lampirkan screenshot

localhost:8888/lab/tree/notebooks/feature\_engineering.ipynb

File Edit View Run Kernel Tabs Settings Help

feature\_engineering.ipynb data\_preprocessing.ipynb eda.ipynb data\_preparation.ipynb Python 3 (ipykernel)

```
[8]: # Graph after oversampling
y_train_ros_df = pd.DataFrame(
    y_train_ros,
    columns=["loan_status"]
)
sns.countplot(
    data=y_train_ros_df,
    x="loan_status",
    hue="loan_status"
)
plt.title("Label Distribution After Oversampling")
plt.show()
```

Mode: Edit Ln 1, Col 1 feature\_engineering.ipynb 1

localhost:8888/lab/tree/notebooks/feature\_engineering.ipynb

File Edit View Run Kernel Tabs Settings Help

feature\_engineering.ipynb data\_preprocessing.ipynb eda.ipynb data\_preparation.ipynb Python 3 (ipykernel)

```
[8]: # Graph after oversampling
y_train_ros_df = pd.DataFrame(
    y_train_ros,
    columns=["loan_status"]
)
sns.countplot(
    data=y_train_ros_df,
    x="loan_status",
    hue="loan_status"
)
plt.title("Label Distribution After Oversampling")
plt.show()
```

Mode: Edit Ln 1, Col 1 feature\_engineering.ipynb 1

i. [1 poin] Serialize train set setelah resample

i.Panggil fungsi `serialize_data()` dari `utils`

ii.Berikan `X_train_ros` sebagai argumen pertama

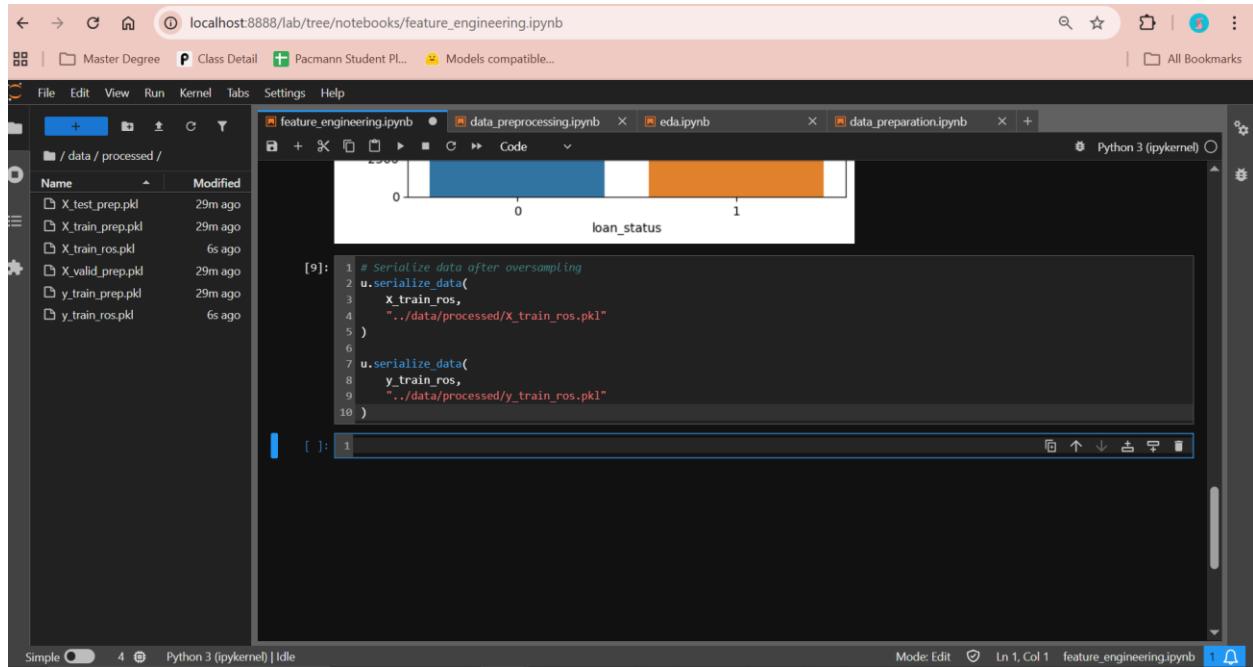
iii.Berikan string `data/processed/X_train_ros.pkl` sebagai argumen kedua

iv. Panggil ulang fungsi `serialize_data()` dari `utils`

v. Berikan `y_train_ros` sebagai argumen pertama

vi. Berikan string data/processed/`y_train_ros.pkl` sebagai argumen kedua

### Lampirkan screenshot



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

- Header:** localhost:8888/lab/tree/notebooks/feature\_engineering.ipynb
- File Explorer:** Shows a directory structure under /data/processed/ containing files: X\_test\_prep.pkl, X\_train\_prep.pkl, X.train\_ros.pkl, X.valid\_prep.pkl, y\_train\_prep.pkl, and y.train\_ros.pkl. The X.train\_ros.pkl file was modified 6s ago.
- Code Editor:** The current cell (cell 9) contains Python code for serializing data after oversampling:

```
# Serialize data after oversampling
u.serialize_data(
    X_train_ros,
    "../data/processed/X_train_ros.pkl"
)
u.serialize_data(
    y_train_ros,
    "../data/processed/y_train_ros.pkl"
)
```

- Output:** A histogram titled "loan\_status" is displayed above the code cell. The x-axis ranges from 0 to 1, and the distribution is highly skewed towards 1, with a small peak at 0.
- Bottom Status Bar:** Shows "Simple" mode, 4 notebooks open, Python 3 (ipykernel) | Idle, Mode: Edit, Ln 1, Col 1, feature\_engineering.ipynb, and a blue status bar indicator.