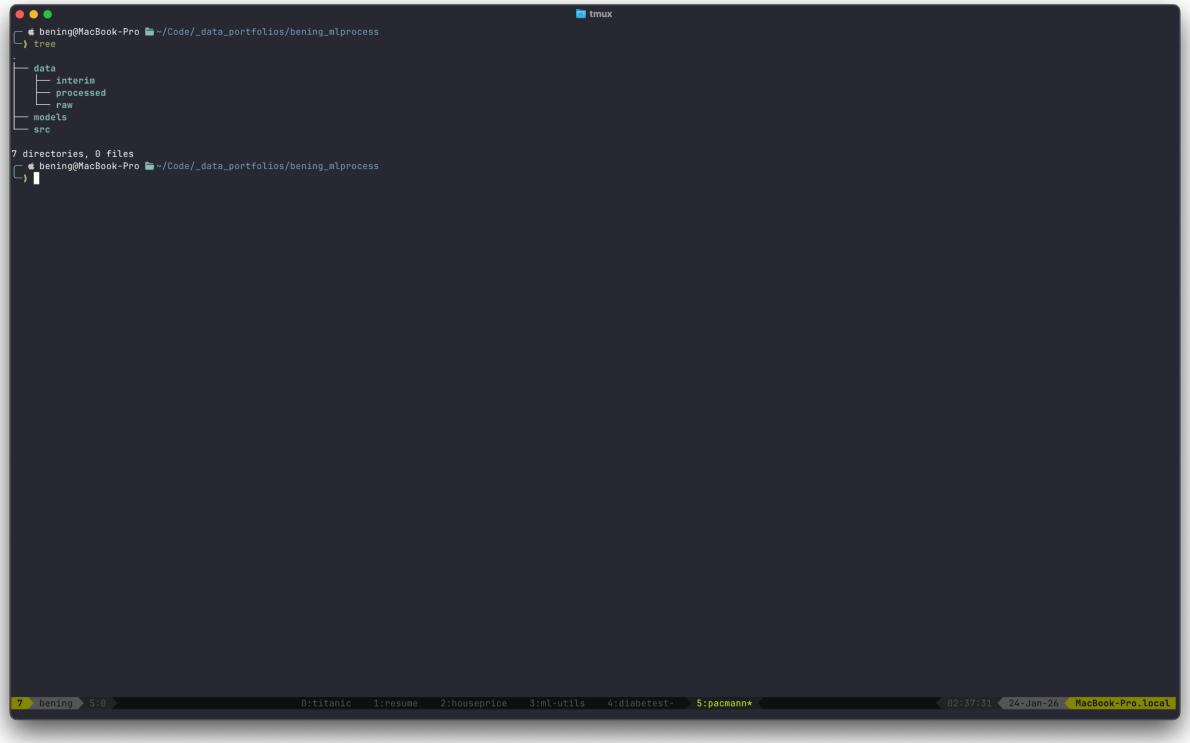
Soal

1. [25 points] Project Environment Setup
    a. [5 points] Buat folder project

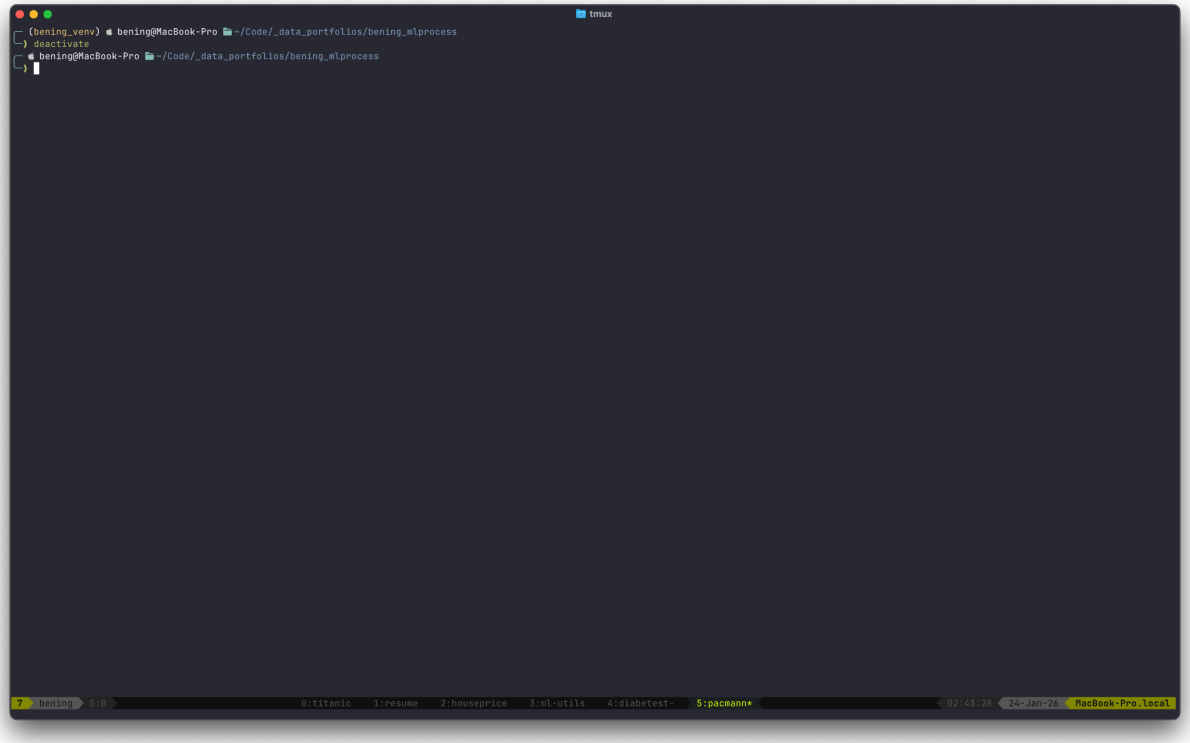


b. [5 points] Buat python virtual environment (venv)

c. [5 points] Update PIP

d. [5 points] Install dependencies

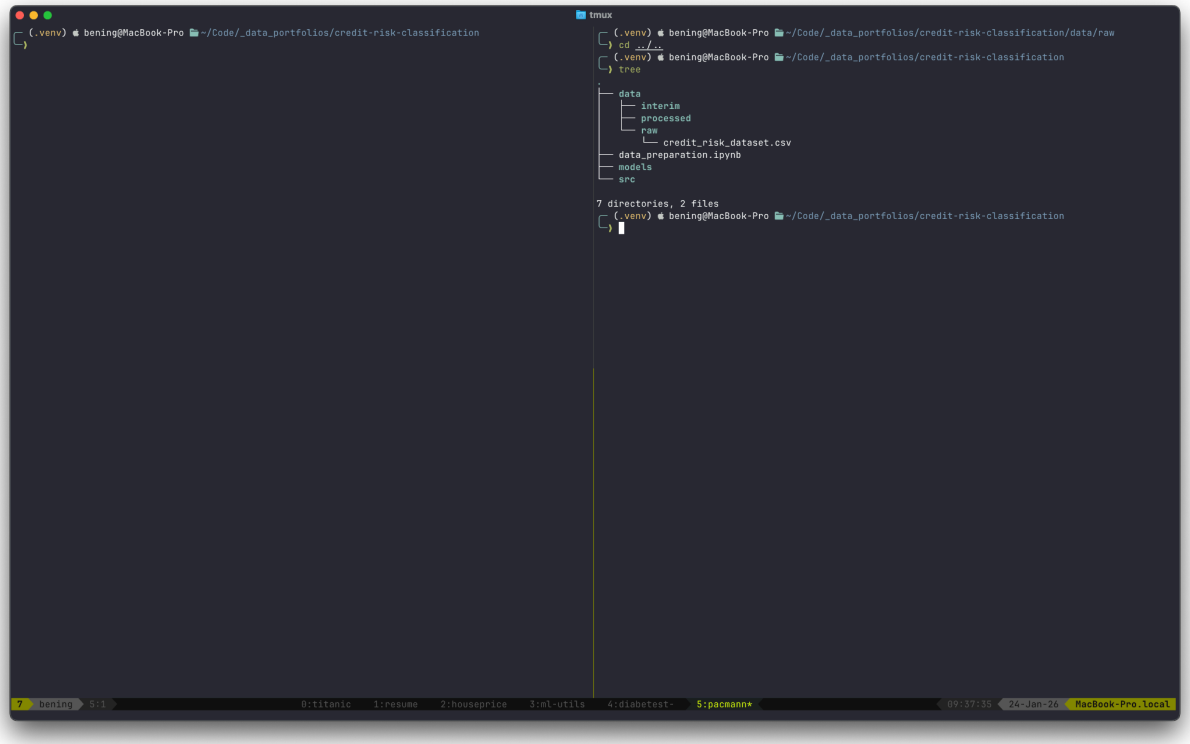
e. [5 points] Non aktifkan venv Anda

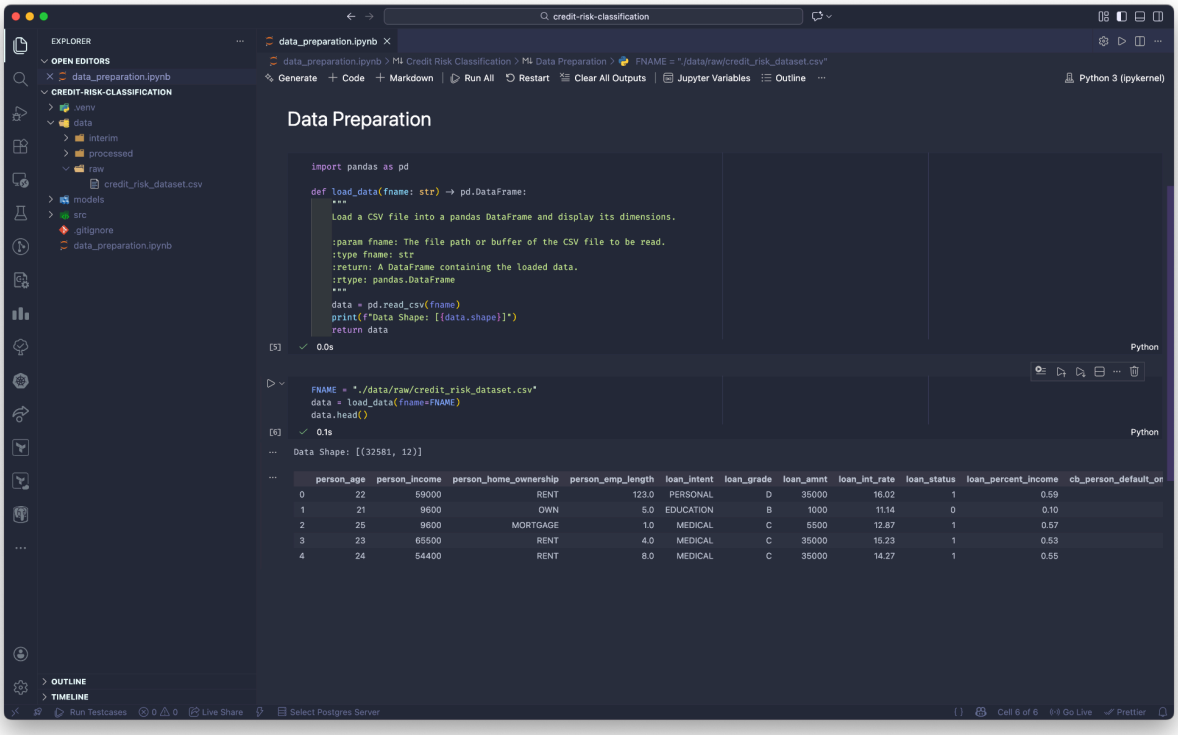


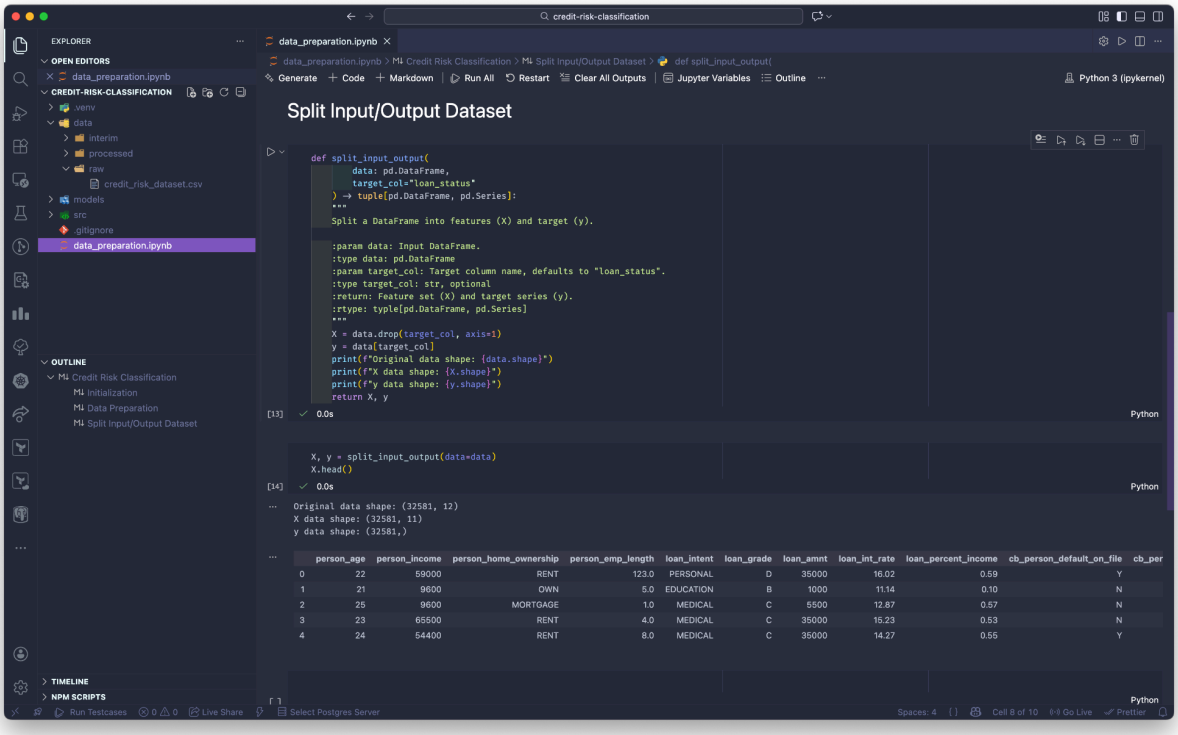
2. [25 points] Rangkum permasalahan bisnis
    - Rangkum mockup interview antara user dan DS di bagian akhir dari dokumen ini
    - **Rangkuman Anda harus bisa menjawab pertanyaan berikut:**

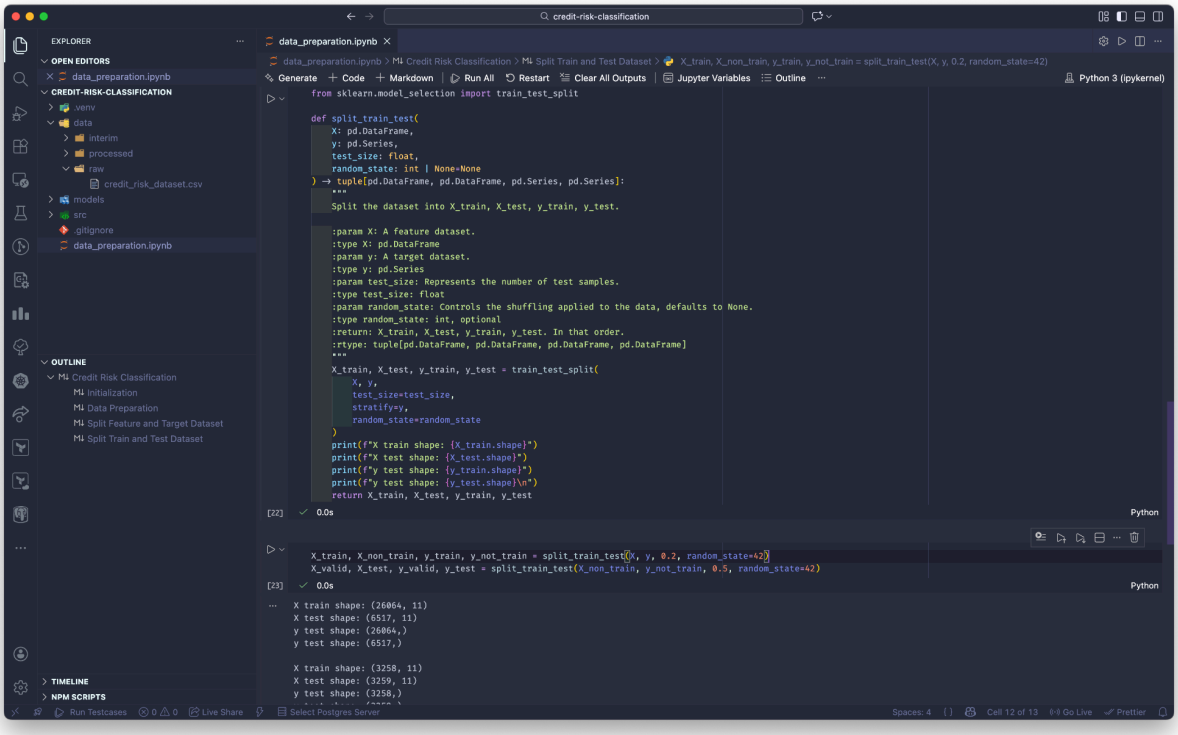
PERTANYAAN	JAWABAN
Latar belakang permasalahan bisnis	*Bank memiliki masalah dengan sistem risiko kredit yang saat ini masih menerima pinjaman-pinjaman yang berpotensi sebagai masalah.*
Objektif bisnis yang ingin dicapai	*Menurunkan angka nasabah-nasabah yang bermasalah*
Metrik pengukuran bisnis yang akan digunakan	*Total Non-Performing Loan*
Kandidat solusi machine learning yang akan dibangun	*Kasus klasifikasi yang berfokus pada ensemble models (Random Forest, XGBoost, Catboost) dengan menggunakan linear models sebagai baseline.*
Metrik pengukuran machine learning yang akan digunakan	*Recall*

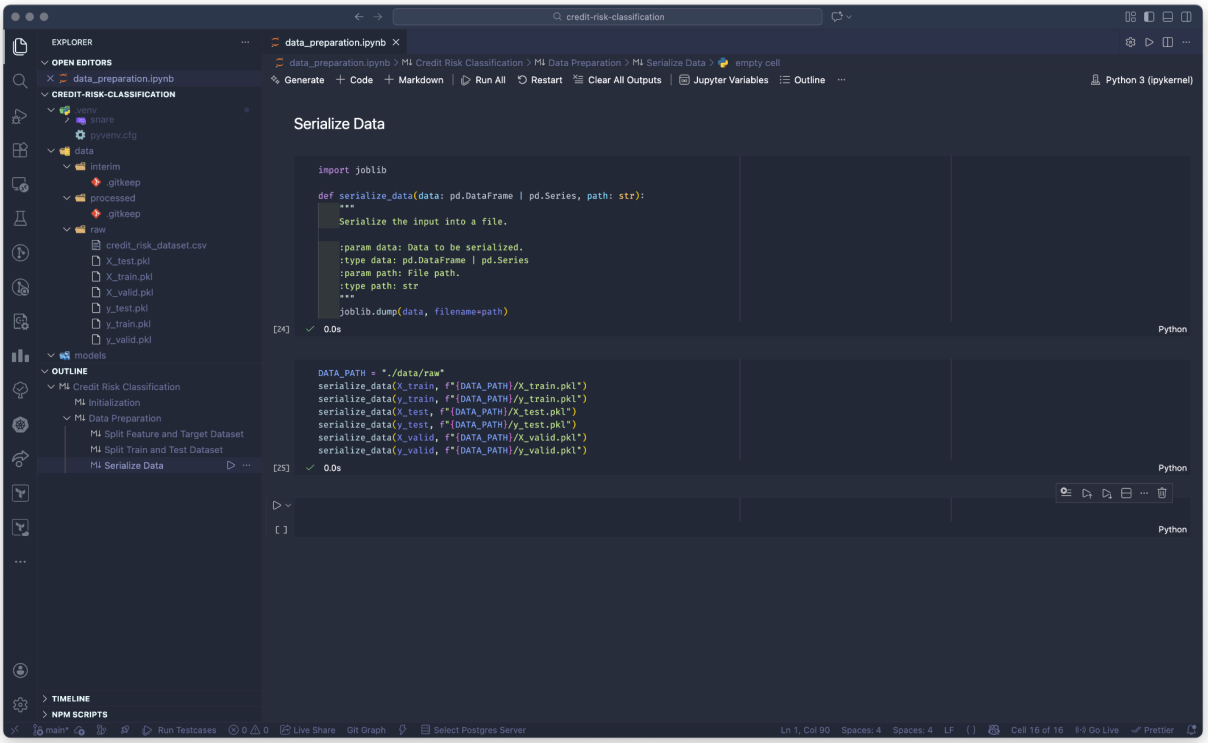
3. [25 points] Persiapan Dataset



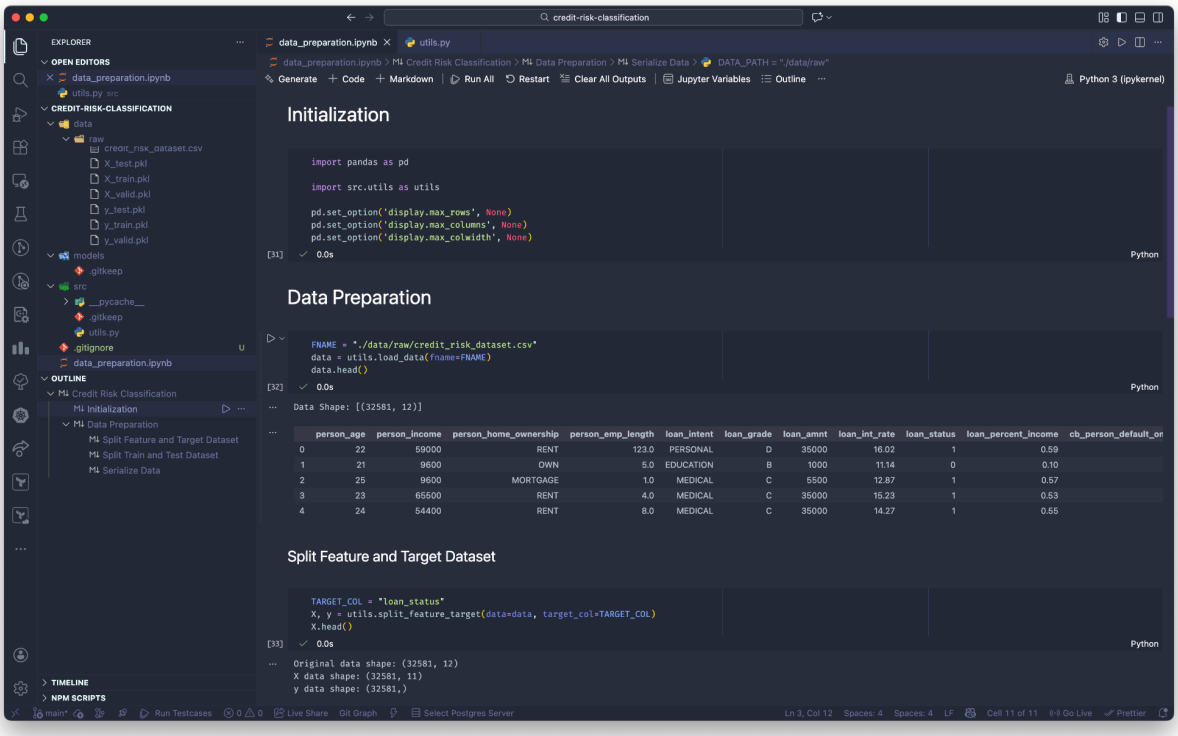


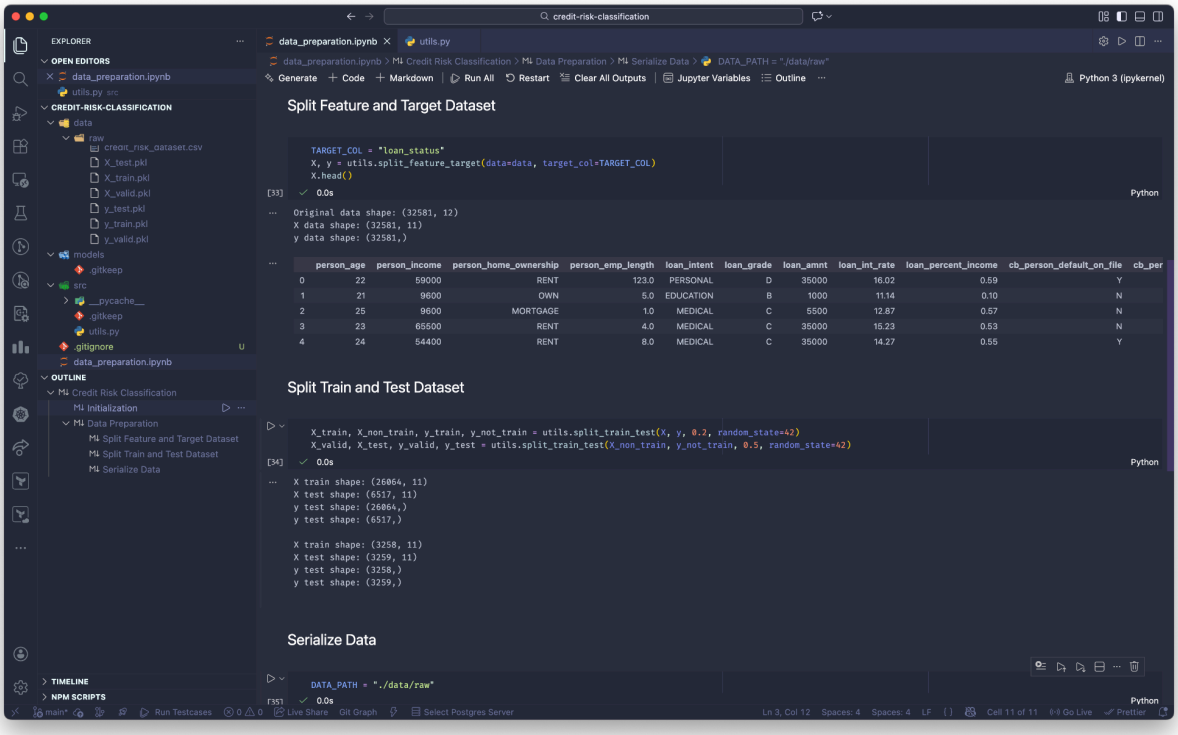

credit-risk-classification
EXPLORER
OPEN EDITORS
data_preparation.ipynb
CREDIT-RISK-CLASSIFICATION
venv
data
interim
processed
raw
credit_risk_dataset.csv
models
src
.gitignore
data_preparation.ipynb
OUTLINE
Credit Risk Classification
Initialization
Data Preparation
Split Input/Output Dataset
TIMELINE
NPM SCRIPTS
data_preparation.ipynb
Generate
Code
Markdown
Run All
Restart
Clear All Outputs
Jupyter Variables
Outline
Python 3 (ipykernel)
Split Input/Output Dataset
def split_input_output(
data: pd.DataFrame,
target_col="loan_status"
) -> tuple[pd.DataFrame, pd.Series]:
"""
Split a DataFrame into features (X) and target (y).
:param data: Input DataFrame.
:type data: pd.DataFrame
:param target_col: Target column name, defaults to "loan_status".
:type target_col: str, optional
:return: Feature set (X) and target series (y).
:rtype: typle[pd.DataFrame, pd.Series]
"""
X = data.drop(target_col, axis=1)
y = data[target_col]
print(f"Original data shape: {data.shape}")
print(f"X data shape: {X.shape}")
print(f"y data shape: {y.shape}")
return X, y
[13] 0.0s
Python
X, y = split_input_output(data=data)
X.head()
[14] 0.0s
Python
Original data shape: (32581, 12)
X data shape: (32581, 11)
y data shape: (32581,)
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_per
0 22 59000 RENT 123.0 PERSONAL D 35000 16.02 0.59 Y
1 21 9600 OWN 5.0 EDUCATION B 1000 11.14 0.10 N
2 25 9600 MORTGAGE 1.0 MEDICAL C 5500 12.87 0.57 N
3 23 65500 RENT 4.0 MEDICAL C 35000 15.23 0.53 N
4 24 54400 RENT 8.0 MEDICAL C 35000 14.27 0.55 Y
Python
Run Testcases
Live Share
Select Postgres Server
Spaces: 4
Cell 8 of 10
Go Live
Prettier


credit-risk-classification
EXPLORER
OPEN EDITORS
data_preparation.ipynb
CREDIT-RISK-CLASSIFICATION
venv
data
interim
processed
raw
credit_risk_dataset.csv
models
src
.gitignore
data_preparation.ipynb
OUTLINE
Credit Risk Classification
Initialization
Data Preparation
Split Feature and Target Dataset
Split Train and Test Dataset
TIMELINE
NPM SCRIPTS
data_preparation.ipynb
Generate
Code
Markdown
Run All
Restart
Clear All Outputs
Jupyter Variables
Outline
Python 3 (ipykernel)
from sklearn.model_selection import train_test_split
def split_train_test(
X: pd.DataFrame,
y: pd.Series,
test_size: float,
random_state: int | None=None
) -> tuple[pd.DataFrame, pd.DataFrame, pd.Series, pd.Series]:
"""
Split the dataset into X_train, X_test, y_train, y_test.
:param X: A feature dataset.
:type X: pd.DataFrame
:param y: A target dataset.
:type y: pd.Series
:param test_size: Represents the number of test samples.
:type test_size: float
:param random_state: Controls the shuffling applied to the data, defaults to None.
:type random_state: int, optional
:return: X_train, X_test, y_train, y_test. In that order.
:rtype: tuple[pd.DataFrame, pd.DataFrame, pd.DataFrame, pd.DataFrame]
"""
X_train, X_test, y_train, y_test = train_test_split(
X, y,
test_size=test_size,
stratify=y,
random_state=random_state
)
print(f"X train shape: {X_train.shape}")
print(f"X test shape: {X_test.shape}")
print(f"y test shape: {y_train.shape}")
print(f"y test shape: {y_test.shape}\n")
return X_train, X_test, y_train, y_test
[22] 0.0s
Python
X_train, X_non_train, y_train, y_not_train = split_train_test(X, y, 0.2, random_state=42)
X_valid, X_test, y_valid, y_test = split_train_test(X_non_train, y_not_train, 0.5, random_state=42)
[23] 0.0s
Python
X train shape: (26064, 11)
X test shape: (6517, 11)
y test shape: (26064,)
y test shape: (6517,)
X train shape: (3258, 11)
X test shape: (3259, 11)
y test shape: (3258,)
Python
Run Testcases
Live Share
Select Postgres Server
Spaces: 4
Cell 12 of 13
Go Live
Prettier


credit-risk-classification
EXPLORER
OPEN EDITORS
data_preparation.ipynb
CREDIT-RISK-CLASSIFICATION
venv
.gitkeep
pyvenv.cfg
data
interim
.gitkeep
processed
.gitkeep
raw
credit_risk_dataset.csv
X_test.pkl
X_train.pkl
X_valid.pkl
y_test.pkl
y_train.pkl
y_valid.pkl
models
OUTLINE
Credit Risk Classification
Initialization
Data Preparation
Split Feature and Target Dataset
Split Train and Test Dataset
Serialize Data
TIMELINE
NPM SCRIPTS
data_preparation.ipynb
data_preparation.ipynb > Credit Risk Classification > Data Preparation > Serialize Data > empty cell
Generate
Code
Markdown
Run All
Restart
Clear All Outputs
Jupyter Variables
Outline
Python 3 (ipykernel)
Serialize Data
import joblib
def serialize_data(data: pd.DataFrame | pd.Series, path: str):
"""
Serialize the input into a file.
:param data: Data to be serialized.
:type data: pd.DataFrame | pd.Series
:param path: File path.
:type path: str
"""
joblib.dump(data, filename=path)
[24]
0.0s
Python
DATA_PATH = "./data/raw"
serialize_data(X_train, f"{DATA_PATH}/X_train.pkl")
serialize_data(y_train, f"{DATA_PATH}/y_train.pkl")
serialize_data(X_test, f"{DATA_PATH}/X_test.pkl")
serialize_data(y_test, f"{DATA_PATH}/y_test.pkl")
serialize_data(X_valid, f"{DATA_PATH}/X_valid.pkl")
serialize_data(y_valid, f"{DATA_PATH}/y_valid.pkl")
[25]
0.0s
Python
[ ]
Python
main*
Run Testcases
Live Share
Git Graph
Select Postgres Server
Ln 1, Col 90
Spaces: 4
Spaces: 4
LF
Cell 16 of 16
Go Live
Prettier

4. [25 points] Buat python utility script





Mockup Percakapan

Pengguna	:	Hai, Pak David! Terima kasih sudah menyempatkan waktu untuk bertemu.
David	:	Hai! Tentu saja, tidak masalah. Saya senang bisa membantu. Ada yang bisa saya bantu?
Pengguna		Jadi begini, Pak David, kami di bank sedang memiliki masalah dengan risiko kredit, nih. Kami ingin bikin sistem prediksi yang bisa bantu kita lebih cepet nangkep pinjaman yang berpotensi jadi masalah, gitu.
David		Ah, mengerti. Jadi tujuannya adalah untuk lebih akurat dalam memprediksi pinjaman bermasalah. Keren, tujuan bisnisnya apa nih?
Pengguna		Ya, tentu aja. Kami pengen turunin jumlah NPL (Non-Performing Loan) dan lebih cepet deteksi pinjaman yang berisiko biar bisa langkah preventif lebih awal.
David		Oke, paham. Jadi pengukuran keberhasilannya nanti gimana?
Pengguna		Gampang, nih. Yang jelas pengen liat jumlah NPL yang beneran turun.
David		Nah, kalo soal solusi machine learning, ada preferensi khusus gak?
Pengguna		Hmm, nggak terlalu sih. Yang penting bisa handle data dalam jumlah besar dan kita bisa coba beberapa model yang beda-beda.
David		Bagus. Terus kalo pengen evaluasi kinerjanya gimana?
Pengguna		Ya, kita memperbolehkan adanya false alarm. Ini karena nantinya kita akan saring lagi yang diprediksi sebagai NPL, apakah emang benar akan NPL? Dan kita juga akan beri treatment agar meminimalisir nasabah yang akan NPL.
David		Terakhir, ada insight apa aja dari eksplorasi data awalnya?
Pengguna		Udah ada beberapa pola menarik terkait perilaku pembayaran pelanggan dan faktor risiko lainnya. Mudah-mudahan nanti model bisa nangkep pola-pola itu dengan baik.
David		Keren, terima kasih udah sharing. Dengan info yang udah kamu kasih, saya bakal buat solusi yang sesuai dengan kebutuhan kamu.
Pengguna		Mantap, Pak David! Kami bener-bener berharap bisa kerja sama dengan Anda.
David		Sama-sama, semoga bisa ngasih hasil yang memuaskan. Kalo ada pertanyaan atau hal lain yang perlu dibahas, jangan ragu buat hubungi gue lagi ya.
Pengguna		Deal! Makasih banyak, Pak David. Sampai ketemu lagi!

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