MLOps Exam 2025 - EN

Summary	The exam assignment consists of two major parts. The first part focus heavily on the Azure Machine Learning story. The goal is to build a pipeline and automate data preprocessing and AI training. In the second part, you will work with HuggingFace to deploy your AI model.
Deadline	You submit no later than 12:30 pm.
Submit	Place the answers to the questions in a .docx or Notion document. You do not need to pay attention to formatting. Take screenshots when prompted. Feel free to add additional notes and screenshots. All your code and this document will be placed in a ZIP file and uploaded via Leho **in a timely manner**.

Objectives

- 1D.2.5 The student responsibly selects and implements the appropriate data service at the level of input, coding, output or maintenance
- 1D.3.1 The student integrates data tailored to the application
- 1D.3.4 The student implements the most efficient algorithm to build an AI application
- 1E.3.1 The student develops an interactive application using a framework or combination of frameworks.
- 2.2.2 The student handles version control in a simple context

Agreements

Accepted

- Internet access, lab solutions, GitHub Copilot, ChatGPT, … and if available your brains.

Not allowed

- Communication with others in any way.

🚧 **Before you start, read the entire document carefully!**

General tips

- If you get stuck, try to rejoin at a later point in the assignment. That way, you will leave fewer points unresolved.
- Ask a question when you are completely stuck. That way you won't lose unnecessary time. We may give you an interim solution if necessary. This may cost one or more points, but you can continue working after that.
- Add one screenshot too many rather than too few. Make sure all relevant information is visible so we can correctly assess your work.
- Use GitHub to keep your code safe during the exam.

Assignment

You will have about 3 hours for this assignment, if you don't finish in time, submit as far as you got, you will also earn a large portion of points for how you tackle your project.

Game of Throne House Predictor

The AI model in this exam is not of great importance, the process is mainly about AI training and the components in a pipeline. Therefore, it is important that you focus on that.

Submit

Create a `.docx` file or a Notion document (export as PDF) with the requested screenshots and a brief explanation of your approach. Note which parts succeeded and which did not.

You should then submit the following via **GitHub Classroom** and **Leho**:

- All the source code
- The `.docx` file or the PDF export from Notion with all screenshots and explanations
- The prompts used (see instructions below)

Submit prompts

We want to track your usage of AI, so we can also give you points on how you got to a certain solution if the answer isn't 100% correct. To track everything correctly, you'll need to submit your prompts and chat history. Below are some instructions for popular tools that we have used. Perhaps your tool works a little different

ChatGPT / Gemini

1. Close the sidebar.
2. Press `Ctrl` or `Cmd` + `S` to save the entire conversation as `.html`.
3. Place this file in a `prompts` folder within your project.

Claude

1. Copy the full conversation.
2. Paste it into Word.
3. Save as a `.pdf`.
4. Place this file in the `prompts` folder.

Copilot Chat (in VSCode)

1. Open the Control Panel of VSCode.
2. Choose **Export Chat**.
3. This generates a `.json` file.
4. Place this file in the `prompts` folder.

Cursor / Antigravity / Perplexity

- Search how to do it yourself, in a way that we can find most of your history.

Azure

Step 0 - Setup of your Azure Machine Learning Workspace.

- For this assignment, you will use your Azure Machine Learning workspace as you used for your final assignment. Either your own environment or use MCT's existing MLOps resource group on Azure.

  Request access to this environment if needed!

- Start up a Compute machine while you take a moment to review the rest of the project. Take a machine that you will later run your Data Processing pipeline on. This one will suffice: **Standard_DS11_v2**

Step 1 - Data Uploading

Points: 3

The data you have been given comes from an AI-generated dataset where there is an attempt to classify which `House` a Game of Thrones character belongs to based on some features.

The task is for you to pour the data preprocessing, as done in some of the cells below, into an Azure pipeline. You've been given some code blocks below that can help with the processing.

Download the data here!!

got_persona_dataset_100.xlsx

Input = `got_persona_dataset_100.xlsx`

Code:

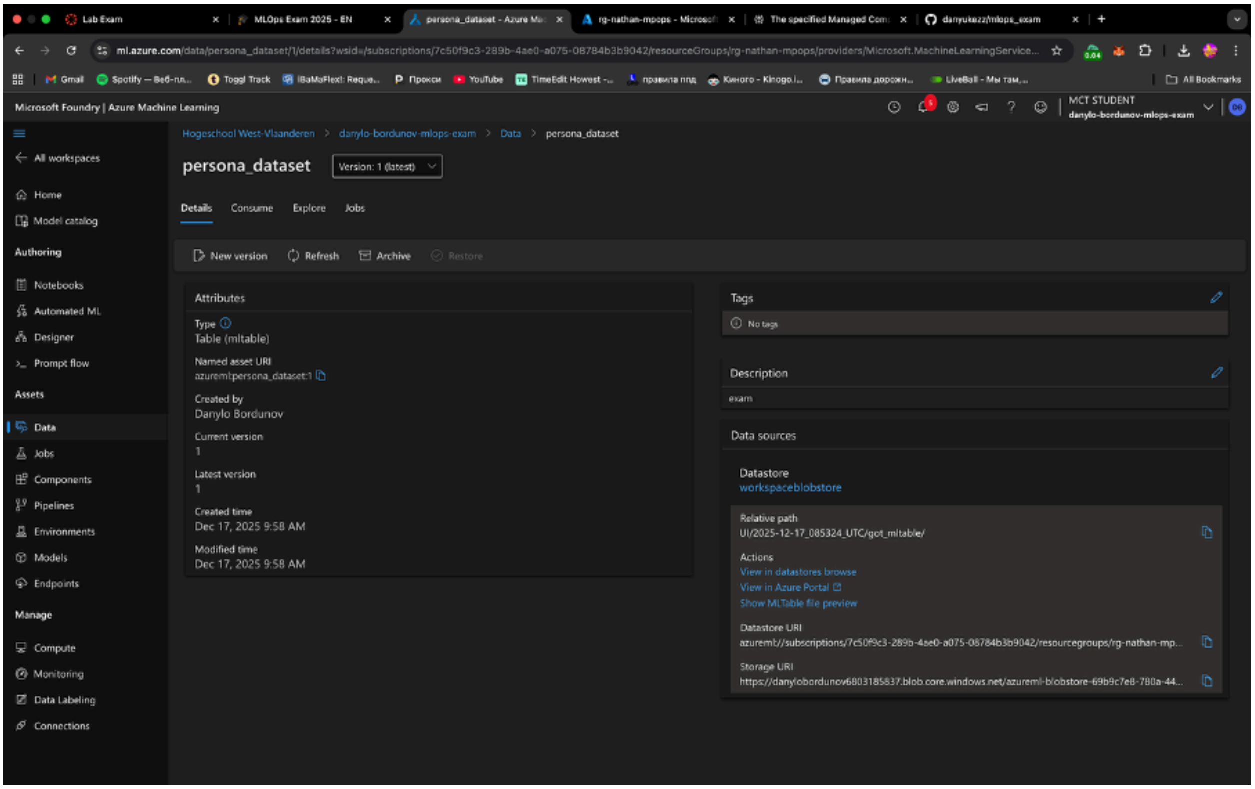
- Make sure the Excel can be loaded. Use the `openpyxl` package:

```
!/anaconda/envs/azureml_py310_sdkv2/bin/python -m pip install openpyxl
```

- Make sure the dataset is uploaded on Azure as a **Tabular** dataset.
- For this, use code to convert the Excel to the correct format.

You may create this via SDK, CLI+YAML or via the GUI Editor.

Screenshot (3 points)
Data Asset with the Tabular data



persona_dataset
Version: 1 (latest)

Attributes

Type
Table (mltable)

Named asset URI
azureml:persona_dataset:1

Created by
Danylo Bordunov

Current version
1

Latest version
1

Created time
Dec 17, 2025 9:58 AM

Modified time
Dec 17, 2025 9:58 AM

Tags
No tags

Description
exam

Data sources

Datastore
workspaceblobstore

Relative path
UI/2025-12-17_085324_UTC/got_mltable/

Actions
View in datastores browse
View in Azure Portal
Show MLTable file preview

Datastore URI
azureml://subscriptions/7c50f9c3-289b-4ae0-a075-08784b3b9042/resourcegroups/rg-nathan-mp...

Storage URI
https://danylobordunov6803185837.blob.core.windows.net/azureml-blobstore-69b9c7e8-780a-44...

Step 1 - AI Data Preparation

Points: 5

You can, using the code below, transform your Data Asset from the raw data, to a trainable version in Training and Testing Features and Targets.

Use a component and a Pipeline to do this.

`conda.yaml` environment file.

```
name: got-prepare-env
channels:
  - conda-forge
dependencies:
  - python=3.10
  - pip
  - pip:
      - pandas
      - scikit-learn
```

`prepare_component.py` for the splitting into Train and Testing data.

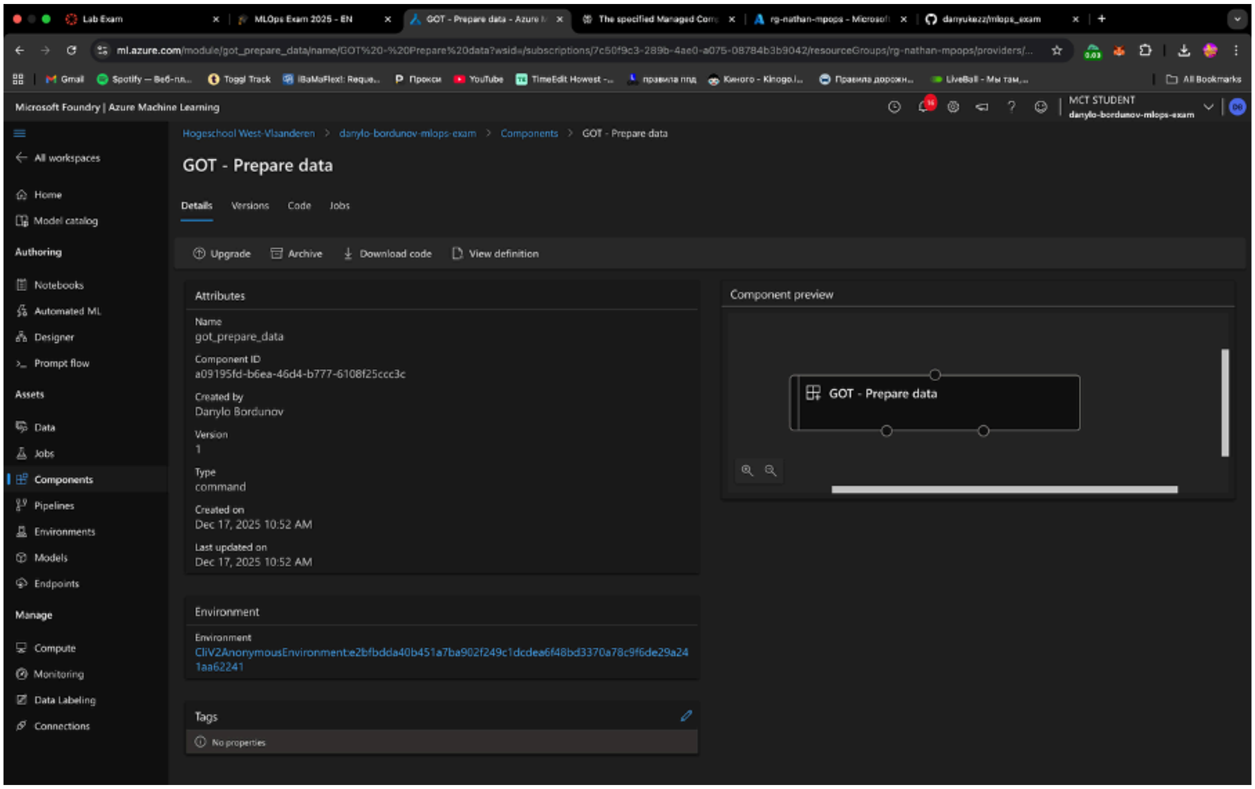
```
import argparse
import json
import os
from glob import glob

import pandas as pd
from sklearn.model_selection import train_test_split


def write_mltable(folder: str, filename: str):
    # MLTable that points to one file
    with open(os.path.join(folder, "MLTable"), "w", encoding="utf-8") as f:
        f.write(
f"""paths:
  - file: ./{filename}
transformations:
  - read_delimited:
      delimiter: ","
      encoding: "utf8"
      header: all_files_same_headers
"""
        )


def main():
    parser = argparse.ArgumentParser()
    parser.add_argument("--input_folder", type=str, required=True)

    parser.add_argument("--target_col", type=str, default="house_affiliation")

    parser.add_argument("--test_size", type=float, default=0.2)
    parser.add_argument("--seed", type=int, default=42)

    parser.add_argument("--stratify", type=int, default=1)  # 1=true, 0=false

    parser.add_argument("--out_train", type=str, required=True)
    parser.add_argument("--out_test", type=str, required=True)

    args = parser.parse_args()

    # locate csv
    csv_path = os.path.join(args.input_folder, "data.csv")
    if not os.path.exists(csv_path):
        csvs = glob(os.path.join(args.input_folder, "*.csv"))
        if not csvs:
            raise FileNotFoundError(f"No CSV found in input folder: {args.input_folder}")
        csv_path = csvs[0]

    df = pd.read_csv(csv_path)
    if args.target_col not in df.columns:
        raise ValueError(f"Target column '{args.target_col}' not found. Columns: {list(df.columns)}")

    # ✅ Drop columns that should never be features (prevent leakage)
    # - character_name is basically an ID / label proxy
    # - character_id is also an ID
    leakage_cols = ["character_id", "character_name"]
    drop_cols = [c for c in leakage_cols if c in df.columns]

    y = df[args.target_col].astype(str)
    X = df.drop(columns=[args.target_col] + drop_cols, errors="ignore")

    strat = None
    if args.stratify == 1:
        vc = y.value_counts()
        too_small = vc[vc < 2]
        if len(too_small) == 0:
            strat = y
            print("✅ Using stratified split.")
        else:
            strat = None
            print("⚠️ Stratify disabled: at least one class has < 2 samples.")
            print("Classes with too few samples:", too_small.to_dict())

    X_train_raw, X_test_raw, y_train, y_test = train_test_split(
        X,
        y,
        test_size=args.test_size,
        random_state=args.seed,
        stratify=strat
    )

    # One-hot encode categoricals on TRAIN, then align TEST
    X_train = pd.get_dummies(X_train_raw, dummy_na=True)
    X_test = pd.get_dummies(X_test_raw, dummy_na=True)

    # Align columns so train/test match exactly
    X_train, X_test = X_train.align(X_test, join="left", axis=1, fill_value=0)

    # Save outputs
    os.makedirs(args.out_train, exist_ok=True)
```

```
    os.makedirs(args.out_test, exist_ok=True)

    X_train_path = os.path.join(args.out_train, "X_train.csv")
    y_train_path = os.path.join(args.out_train, "y_train.csv")
    X_test_path = os.path.join(args.out_test, "X_test.csv")
    y_test_path = os.path.join(args.out_test, "y_test.csv")

    X_train.to_csv(X_train_path, index=False, encoding="utf-8")
    y_train.to_frame(name=args.target_col).to_csv(y_train_path, index=False, encoding="utf-8")

    X_test.to_csv(X_test_path, index=False, encoding="utf-8")
    y_test.to_frame(name=args.target_col).to_csv(y_test_path, index=False, encoding="utf-8")

    # Save metadata for reproducibility & inference compatibility
    with open(os.path.join(args.out_train, "feature_columns.json"), "w", encoding="utf-8") as f:
        json.dump({"columns": list(X_train.columns)}, f, ensure_ascii=False, indent=2)

    classes = sorted(y.unique().tolist())
    with open(os.path.join(args.out_train, "label_classes.json"), "w", encoding="utf-8") as f:
        json.dump({"classes": classes}, f, ensure_ascii=False, indent=2)

    # Add MLTable definitions (point to the main files)
    write_mltable(args.out_train, "X_train.csv")
    write_mltable(args.out_test, "X_test.csv")

    print("✅ Done")
    print("Target:", args.target_col)
    print("Train shape:", X_train.shape, "Test shape:", X_test.shape)
    print("Num classes:", len(classes))
```

```
if __name__ == "__main__":
    main()
```

You may create this via SDK, CLI+YAML or via the GUI Editor.

Screenshot (2 points)
Your component for AI Data Preparation in the Component tab of Azure.



GOT - Prepare data

Attributes

Name
got_prepare_data

Component ID
a09195fd-b6ea-46d4-b777-6108f25ccc3c

Created by
Danylo Bordunov

Version
1

Type
command

Created on
Dec 17, 2025 10:52 AM

Last updated on
Dec 17, 2025 10:52 AM

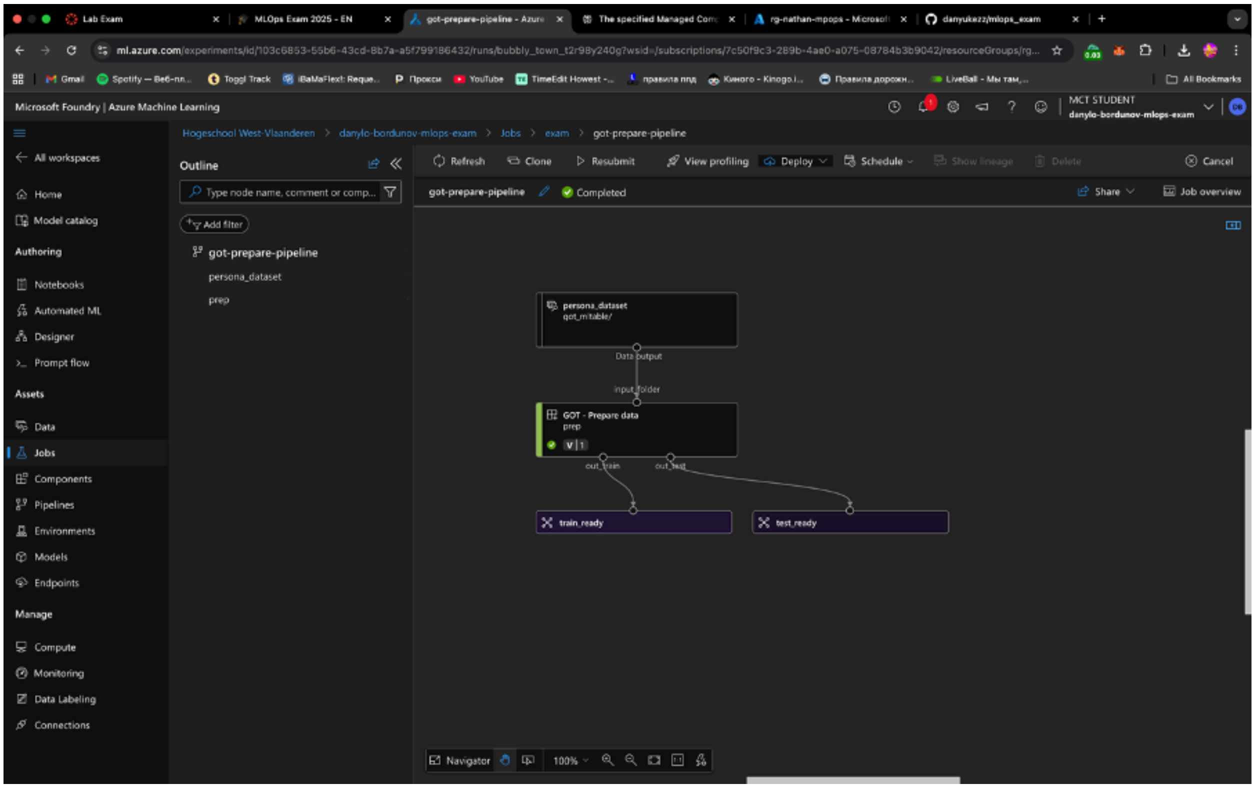
Environment

Environment
CliV2AnonymousEnvironment:e2bfbdda40b451a7ba902f249c1dcdea6f48bd3370a78c9f6de29a241aa62241

Tags

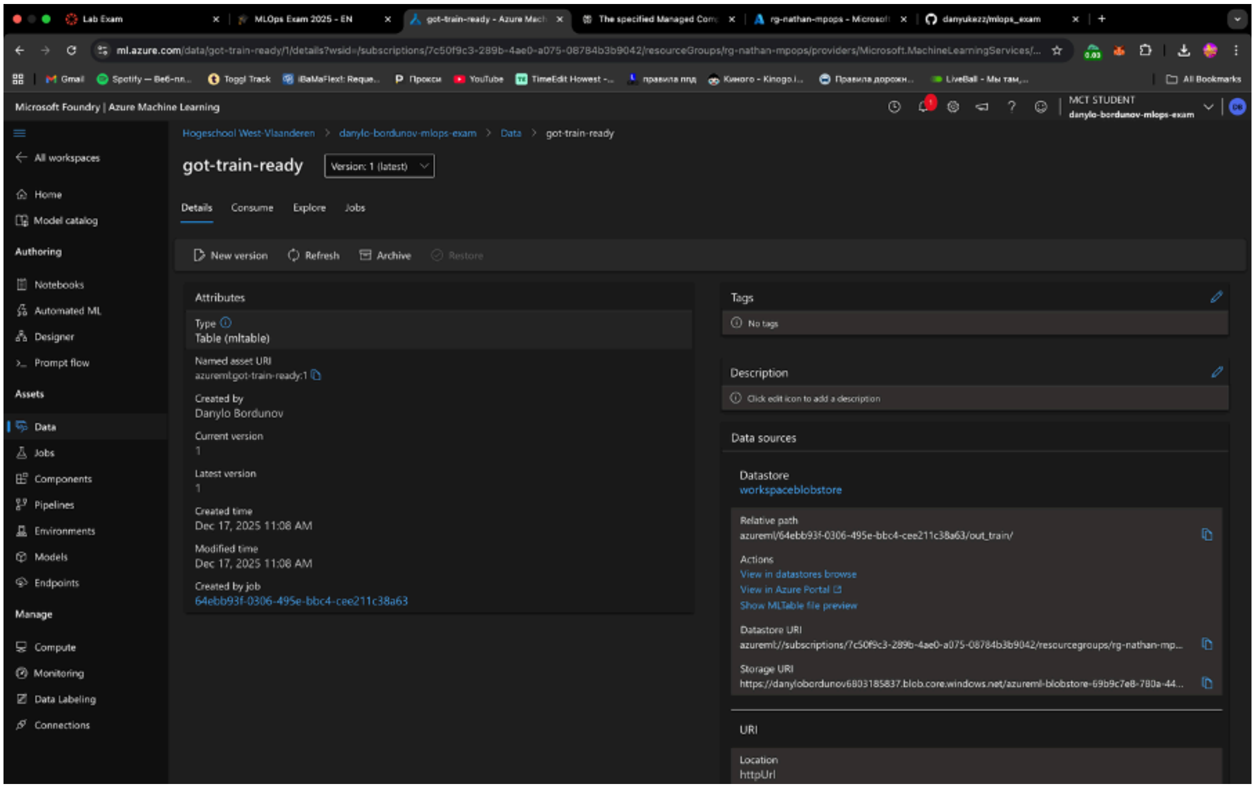
No properties

Screenshot (2 points)
Your **successful** pipeline with **Input** and **Output** visible.



Screenshot (1 point)

Your stored Data Assets for **Train** and **Testing** data.


Lab Exam
MLOps Exam 2025 - EN
got-train-ready - Azure Mach
The specified Managed Com
rg-nathan-mpops - Microsoft
danyukezz/mlops_exam
ml.azure.com/data/got-train-ready/1/details?wsid=/subscriptions/7c50f9c3-289b-4ae0-a075-08784b3b9042/resourceGroups/rg-nathan-mpops/providers/Microsoft.MachineLearningServices/...
Microsoft Foundry | Azure Machine Learning
MCT STUDENT
danylo-bordunov-mlops-exam
Hogeschool West-Vlaanderen > danylo-bordunov-mlops-exam > Data > got-train-ready
All workspaces
Home
Model catalog
Authoring
Notebooks
Automated ML
Designer
Prompt flow
Assets
Data
Jobs
Components
Pipelines
Environments
Models
Endpoints
Manage
Compute
Monitoring
Data Labeling
Connections
Details
Consume
Explore
Jobs
New version
Refresh
Archive
Restore

got-train-ready

Version: 1 (latest)

Attributes

Type
Table (mltable)

Named asset URI
azureml:got-train-ready:1

Created by
Danylo Bordunov

Current version
1

Latest version
1

Created time
Dec 17, 2025 11:08 AM

Modified time
Dec 17, 2025 11:08 AM

Created by job
64ebb93f-0306-495e-bbc4-cee211c38a63

Tags

No tags

Description

Click edit icon to add a description

Data sources

Datastore
workspaceblobstore

Relative path
azureml/64ebb93f-0306-495e-bbc4-cee211c38a63/out_train/

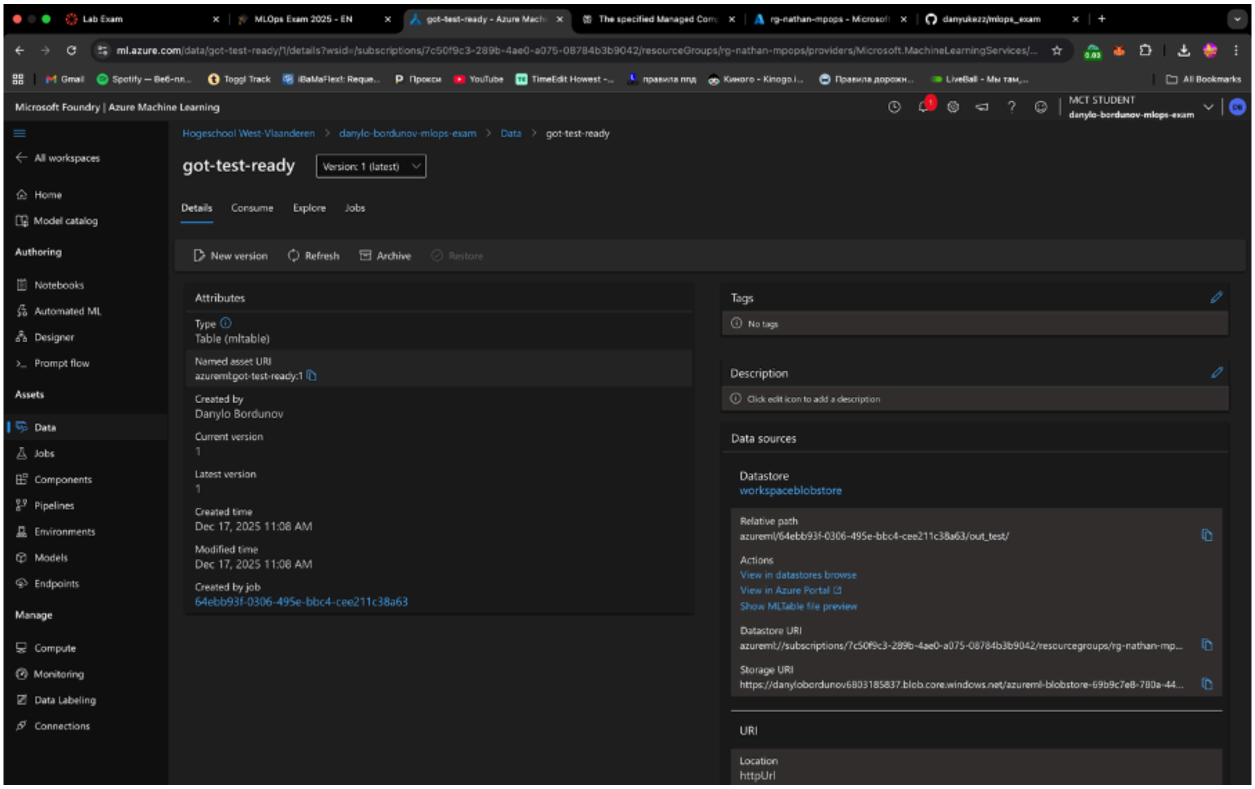
Actions
View in datastores browse
View in Azure Portal
Show MLTable file preview

Datastore URI
azureml://subscriptions/7c50f9c3-289b-4ae0-a075-08784b3b9042/resourcegroups/rg-nathan-mp...

Storage URI
https://danylobordunov6803185837.blob.core.windows.net/azureml-blobstore-69b9c7e8-780a-44...

URI

Location
httpUrl


Lab Exam
MLOps Exam 2025 - EN
got-test-ready - Azure Mach
The specified Managed Com
rg-nathan-mpops - Microsoft
danyukezz/mlops_exam
ml.azure.com/data/got-test-ready/1/details?wsid=/subscriptions/7c50f9c3-289b-4ae0-a075-08784b3b9042/resourceGroups/rg-nathan-mpops/providers/Microsoft.MachineLearningServices/...
Microsoft Foundry | Azure Machine Learning
MCT STUDENT
danylo-bordunov-mlops-exam
Hogeschool West-Vlaanderen > danylo-bordunov-mlops-exam > Data > got-test-ready
All workspaces
Home
Model catalog
Authoring
Notebooks
Automated ML
Designer
Prompt flow
Assets
Data
Jobs
Components
Pipelines
Environments
Models
Endpoints
Manage
Compute
Monitoring
Data Labeling
Connections
Details
Consume
Explore
Jobs
New version
Refresh
Archive
Restore

got-test-ready

Version: 1 (latest)

Attributes

Type
Table (mltable)

Named asset URI
azureml:got-test-ready:1

Created by
Danylo Bordunov

Current version
1

Latest version
1

Created time
Dec 17, 2025 11:08 AM

Modified time
Dec 17, 2025 11:08 AM

Created by job
64ebb93f-0306-495e-bbc4-cee211c38a63

Tags

No tags

Description

Click edit icon to add a description

Data sources

Datastore
workspaceblobstore

Relative path
azureml/64ebb93f-0306-495e-bbc4-cee211c38a63/out_test/

Actions
View in datastores browse
View in Azure Portal
Show MLTable file preview

Datastore URI
azureml://subscriptions/7c50f9c3-289b-4ae0-a075-08784b3b9042/resourcegroups/rg-nathan-mp...

Storage URI
https://danylobordunov6803185837.blob.core.windows.net/azureml-blobstore-69b9c7e8-780a-44...

URI

Location
httpUrl

Step 2 - AI Training - Decision Tree Classifier

Points: 7

For the AI model, we work with a Decision Tree Classifier. The Environment for this is also given, as well as the code to initiate the training itself.

```
name: got-train-dt-env
channels:
  - conda-forge
dependencies:
  - python=3.10
  - pip
  - pip:
      - pandas
      - scikit-learn
      - joblib
```

`training.py` - Code has been given, not yet in the right structure to do everything in a component manner though

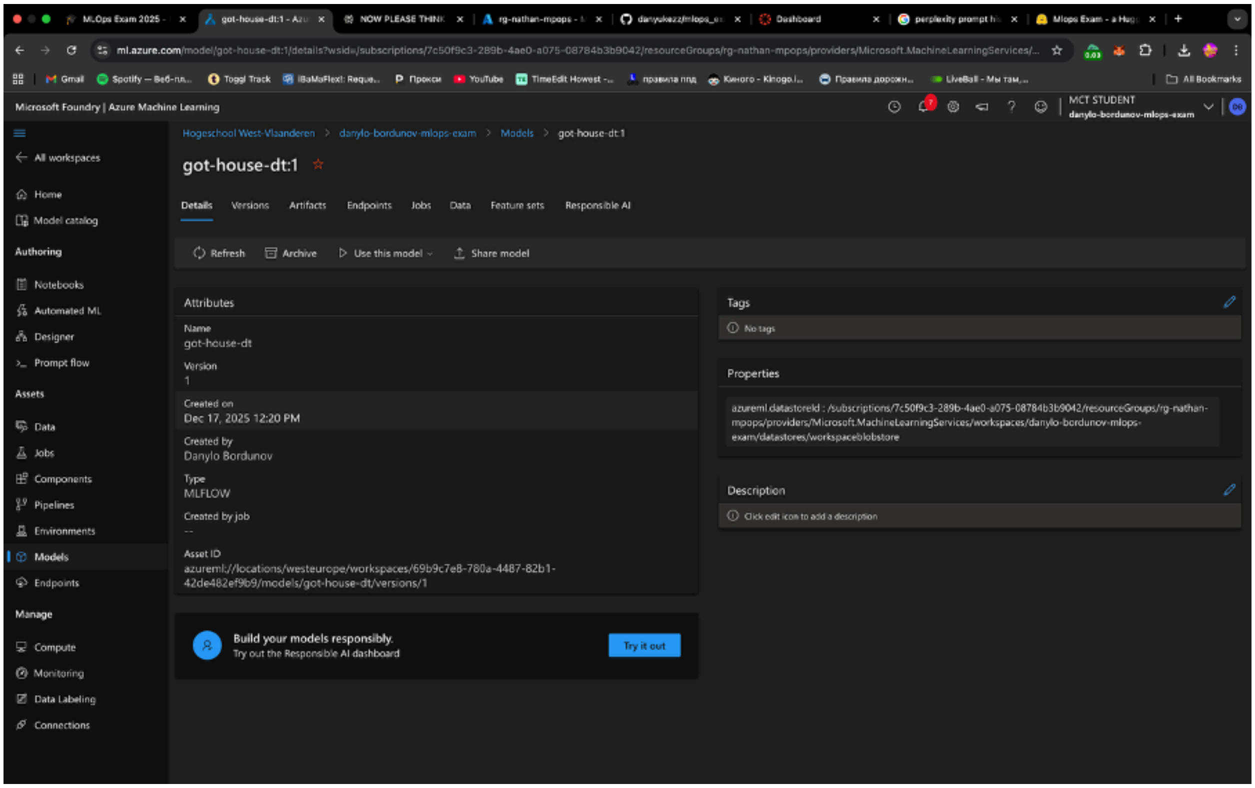
```
# Load data
X_train_path = os.path.join(args.train_ready, "X_train.csv")
y_train_path = os.path.join(args.train_ready, "y_train.csv")
X_test_path  = os.path.join(args.test_ready, "X_test.csv")
y_test_path  = os.path.join(args.test_ready, "y_test.csv")

for p in [X_train_path, y_train_path, X_test_path, y_test_path]:
    if not os.path.exists(p):
        raise FileNotFoundError(f"Missing required file: {p}")

X_train = pd.read_csv(X_train_path)
y_train = pd.read_csv(y_train_path)[args.target_col].astype(str)

X_test = pd.read_csv(X_test_path)
y_test = pd.read_csv(y_test_path)[args.target_col].astype(str)

# Train model
```

```
clf = DecisionTreeClassifier(
    max_depth=args.max_depth,
    min_samples_split=args.min_samples_split,
    min_samples_leaf=args.min_samples_leaf,
    random_state=args.random_state,
)
clf.fit(X_train, y_train)

# Evaluate
y_pred = clf.predict(X_test)
acc = float(accuracy_score(y_test, y_pred))

report = classification_report(y_test, y_pred, output_dict=True, zero_division=0)
```

Checklist

- ☑ ~~Log the metrics from the Classification Report into the Azure ML Experiments using **MLFlow Logging**~~
- ☑ ~~Make sure your AI model gets stored and registered in your Azure ML Pipeline~~
    - ☑ ~~This is done in a separate step with this component: `azureml://registries/azureml/components/register_model/versions/0.0.21`~~

🖥️ **Screenshot (1 point)**
Your registered AI model in the **Model** view.



ml.azure.com/model/got-house-dt:1/details?wsid=/subscriptions/7c50f9c3-289b-4ae0-a075-08784b3b9042/resourceGroups/rg-nathan-mpops/providers/Microsoft.MachineLearningServices/...

got-house-dt:1

Attributes

Name
got-house-dt

Version
1

Created on
Dec 17, 2025 12:20 PM

Created by
Danylo Bordunov

Type
MLFLOW

Created by job
--

Asset ID
azureml://locations/westeurope/workspaces/69b9c7e8-780a-4487-82b1-42de482ef9b9/models/got-house-dt/versions/1

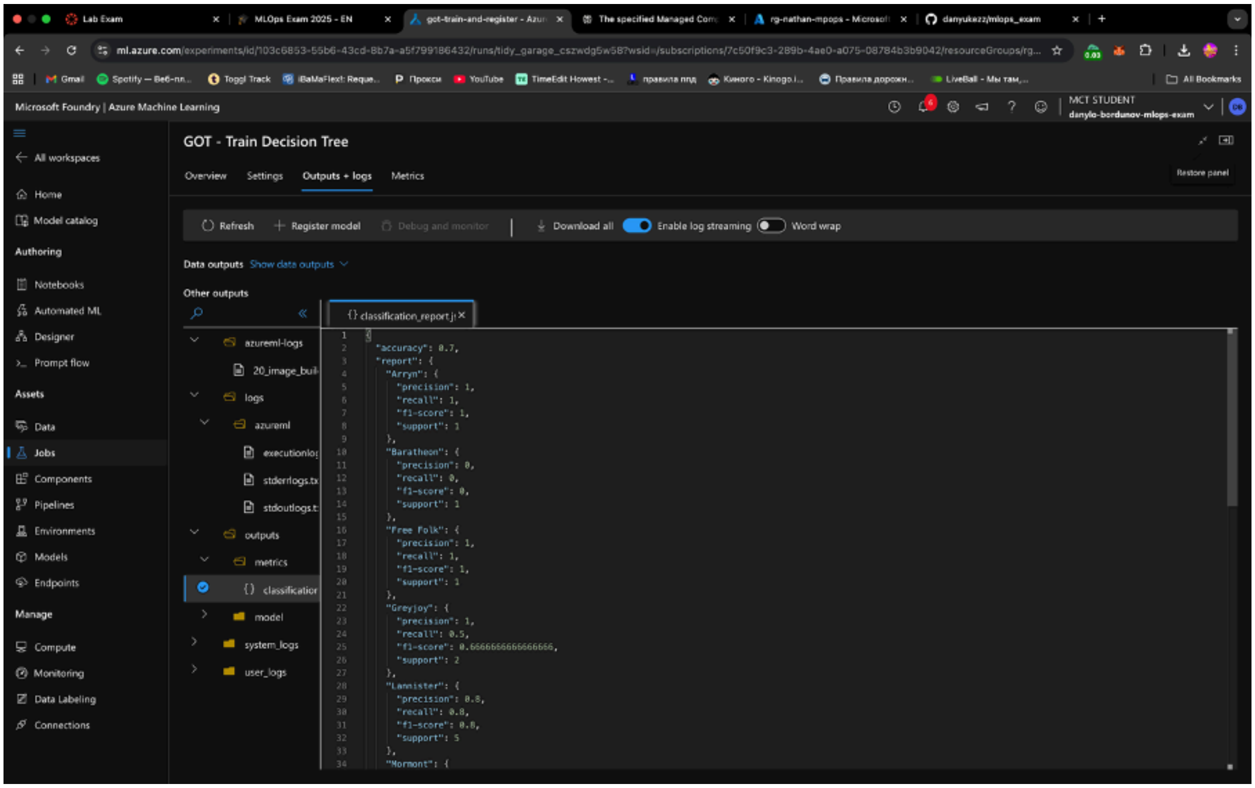
Build your models responsibly.
Try out the Responsible AI dashboard

Tags
No tags

Properties
azureml.datastoreId : /subscriptions/7c50f9c3-289b-4ae0-a075-08784b3b9042/resourceGroups/rg-nathan-mpops/providers/Microsoft.MachineLearningServices/workspaces/danylo-bordunov-mlops-exam/datastores/workspaceblobstore

Description
Click edit icon to add a description

Screenshot (2 points)
Your Classification Report metrics from MLFlow

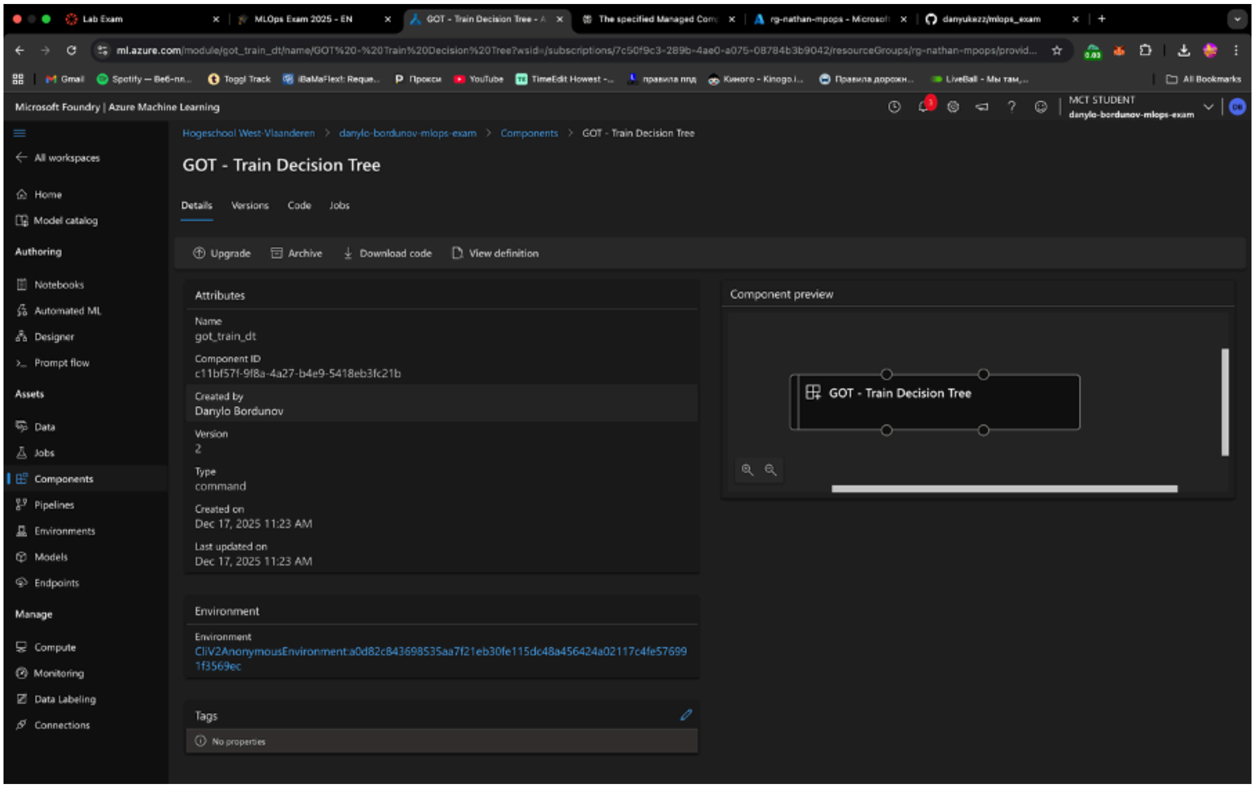


GOT - Train Decision Tree

```
{
  "accuracy": 0.7,
  "report": {
    "Arryn": {
      "precision": 1,
      "recall": 1,
      "f1-score": 1,
      "support": 1
    },
    "Baratheon": {
      "precision": 0,
      "recall": 0,
      "f1-score": 0,
      "support": 1
    },
    "Free Folk": {
      "precision": 1,
      "recall": 1,
      "f1-score": 1,
      "support": 1
    },
    "Greyjoy": {
      "precision": 1,
      "recall": 0.5,
      "f1-score": 0.6666666666666666,
      "support": 2
    },
    "Lannister": {
      "precision": 0.8,
      "recall": 0.8,
      "f1-score": 0.8,
      "support": 5
    },
    "Mormont": {
```

Screenshot (2 points)

Your component for AI Training.



ml.azure.com/module/got_train_dt/name/GOT%20-%20Train%20Decision%20Tree?wsid=/subscriptions/7c50f9c3-289b-4ae0-a075-08784b3b9042/resourceGroups/rg-nathan-mpops/provid...

GOT - Train Decision Tree

Attributes

Name
got_train_dt

Component ID
c11bf57f-9f8a-4a27-b4e9-5418eb3fc21b

Created by
Danylo Bordunov

Version
2

Type
command

Created on
Dec 17, 2025 11:23 AM

Last updated on
Dec 17, 2025 11:23 AM

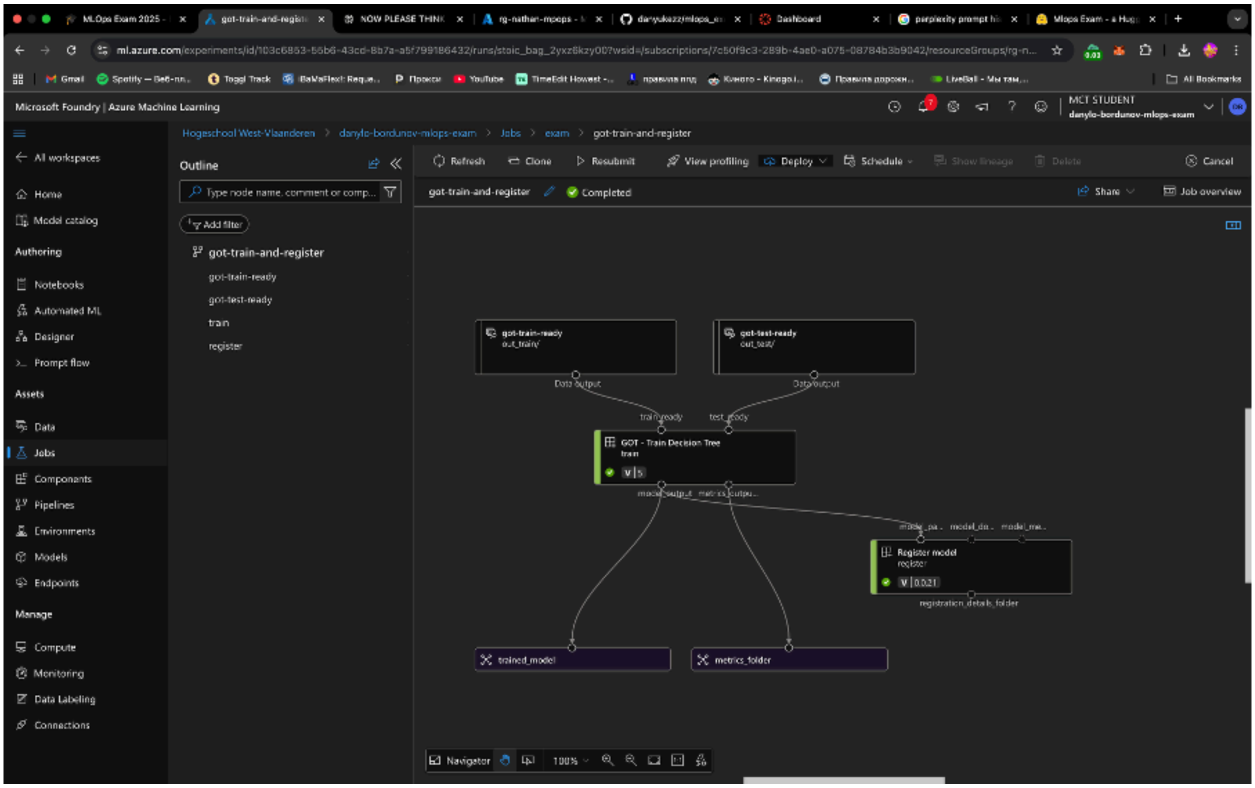
Environment

Environment
CliV2AnonymousEnvironment:a0d82c843698535aa7f21eb30fe115dc48a456424a02117c4fe57699 1f3569ec

Tags

No properties

Screenshot (2 points)
Your **working** pipeline with **Input** and **Output**.



Step 3 - Deployment

Points: 5

💡 You can perform this step on your local PC, without using Azure Machine Learning.

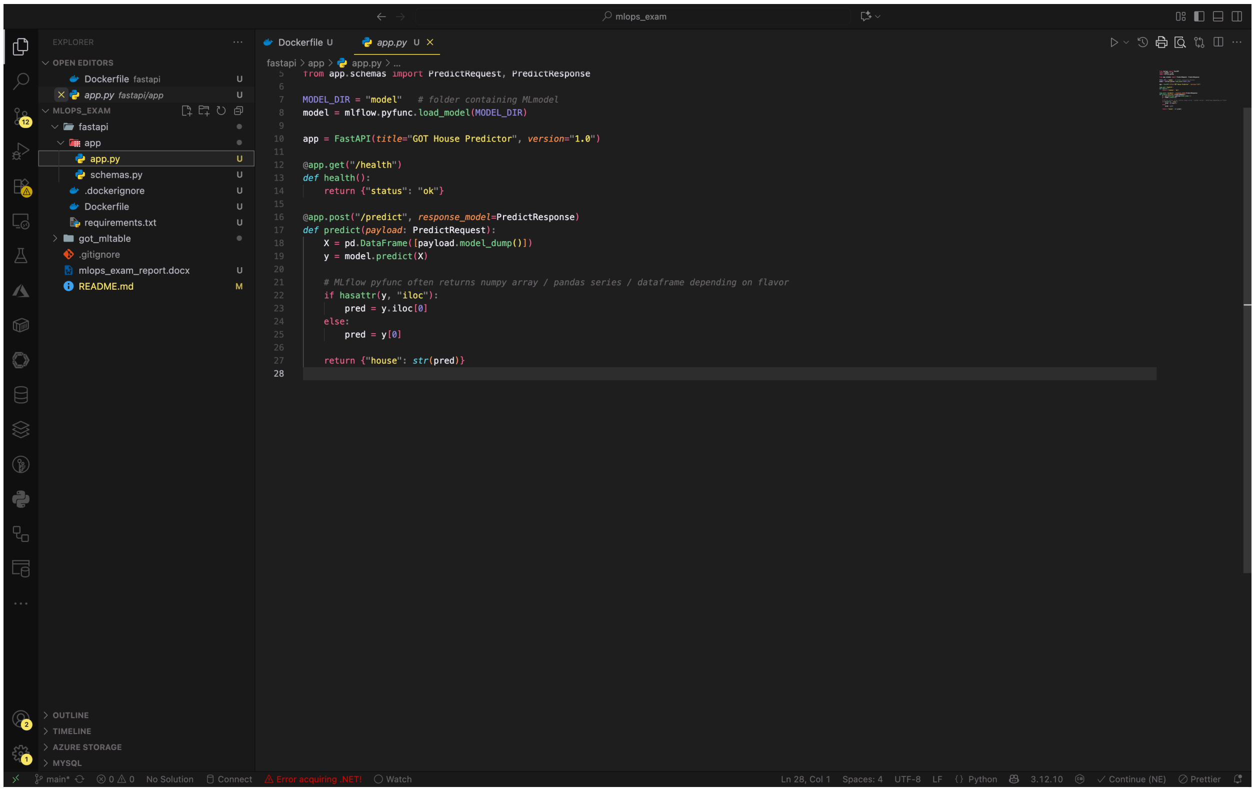
I do think it would be nice if there is an API in FastAPI or Gradio where the AI model can be applied.

You will need to send a request to an API endpoint `(/predict`) with the JSON payload below.

I expect a response where you show which **House** a character belongs to based on the JSON input entered.

Also post this on HuggingFace The necessary inputs can then be entered as follows:

```
{
  "region": "The North",
  "primary_role": "Commander",
  "alignment": "Lawful Good",
  "status": "Alive",
  "species": "Human",
  "honour_1to5": 4,
  "ruthlessness_1to5": 2,
  "intelligence_1to5": 3,
  "combat_skill_1to5": 4,
  "diplomacy_1to5": 3,
  "leadership_1to5": 4,
  "trait_loyal": true,
  "trait_scheming": false
}
```



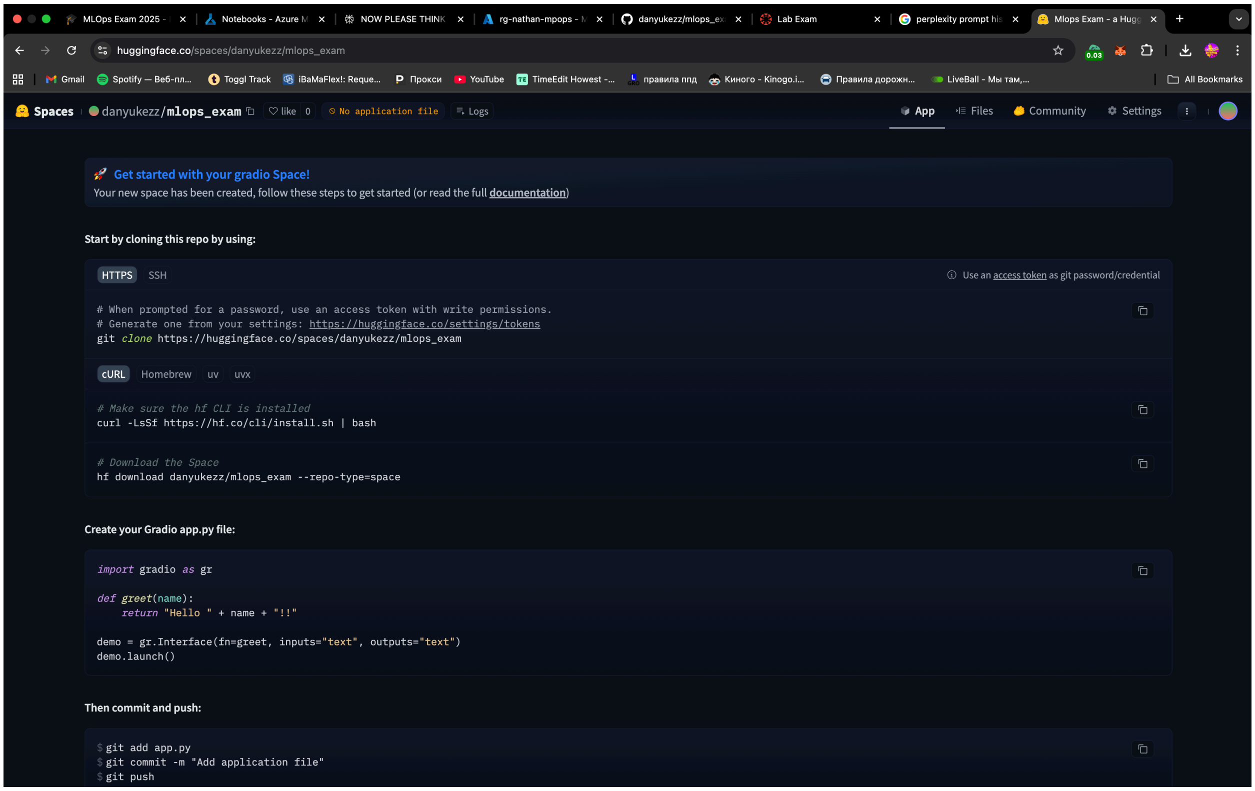
```python
from app.schemas import PredictRequest, PredictResponse

MODEL_DIR = "model"   # folder containing MLmodel
model = mlflow.pyfunc.load_model(MODEL_DIR)

app = FastAPI(title="GOT House Predictor", version="1.0")

@app.get("/health")
def health():
    return {"status": "ok"}

@app.post("/predict", response_model=PredictResponse)
def predict(payload: PredictRequest):
    X = pd.DataFrame([payload.model_dump()])
    y = model.predict(X)

    # MLflow pyfunc often returns numpy array / pandas series / dataframe depending on flavor
    if hasattr(y, "iloc"):
        pred = y.iloc[0]
    else:
        pred = y[0]

    return {"house": str(pred)}

```

Screenshot (3 points)

Your `/docs` endpoint showing a successful execution of the input data with output results. If you chose Gradio, you may also show a screenshot of your frontend, preferably with input and output visible.

Screenshot (2 points)

Show your HuggingFace Space where this is running.



Get started with your gradio Space!

Your new space has been created, follow these steps to get started (or read the full documentation)

Start by cloning this repo by using:

HTTPS SSH — Use an access token as git password/credential

```
# When prompted for a password, use an access token with write permissions.
# Generate one from your settings: https://huggingface.co/settings/tokens
git clone https://huggingface.co/spaces/danyukezz/mlops_exam
```

cURL Homebrew uv uvx

```
# Make sure the hf CLI is installed
curl -LsSf https://hf.co/cli/install.sh | bash
```

```
# Download the Space
hf download danyukezz/mlops_exam --repo-type=space
```

Create your Gradio app.py file:

```
import gradio as gr

def greet(name):
    return "Hello " + name + "!!"

demo = gr.Interface(fn=greet, inputs="text", outputs="text")
demo.launch()
```

Then commit and push:

```
$ git add app.py
$ git commit -m "Add application file"
$ git push
```

Submit

Submission instructions are at the top of this document. Will you check the following?

- ☑ ~~Have you posted the necessary screenshots everywhere?~~
- ☑ ~~Did you turn off your Azure services at the end of the exam?~~

- ☑ ~~Did you download all your code on your PC, and push it on GitHub?~~
- ☑ ~~Did you compress the total project into a `.zip` folder?~~

If you did everything, export this document as a `.pdf` and upload it along with your `.zip` folder on Leho.

Additional

You may also provide your feedback on this course in the document 😊.

Questions?

You can send me a message directly on Teams, or raise your hand in the classroom.

Technical problems with your computer or the network, you should deal with them through the Helpdesk. They can solve those problems.

Good luck!