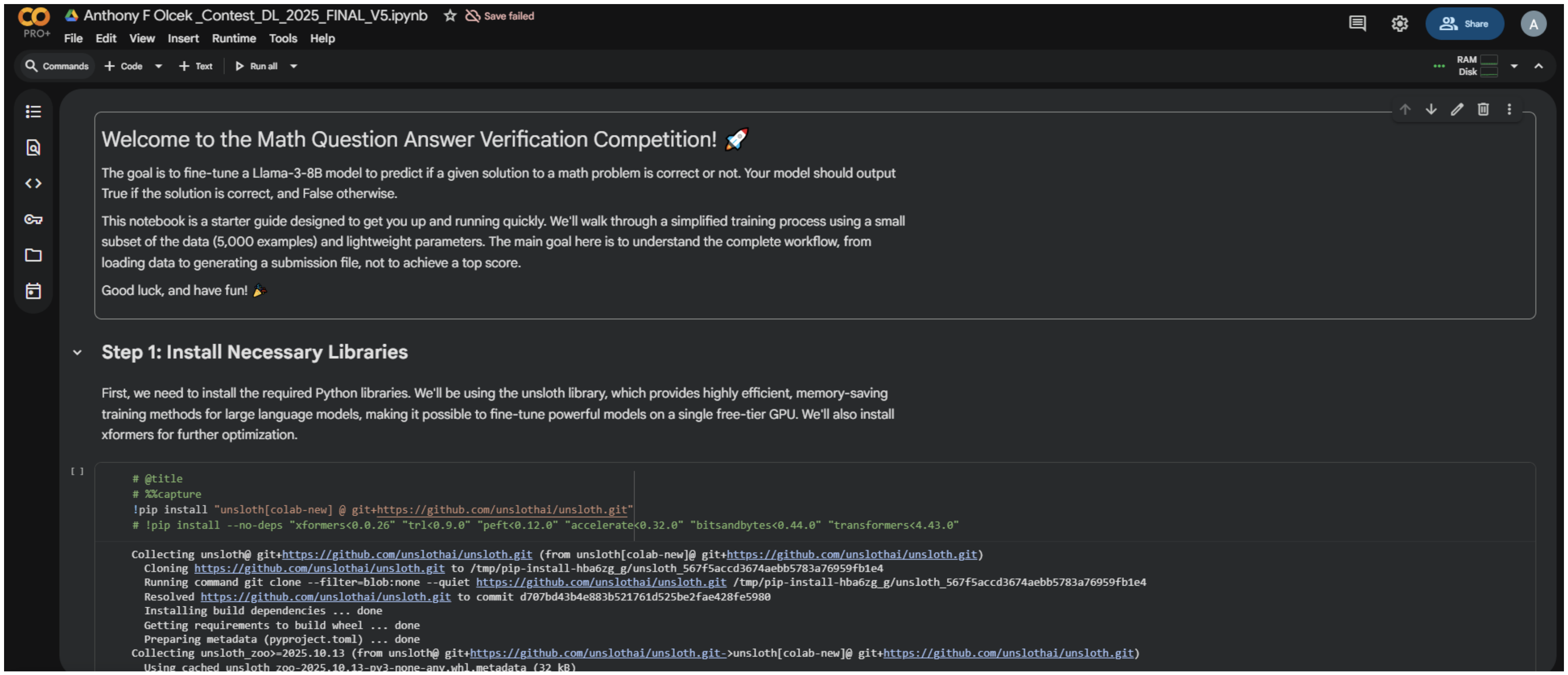

Anthony F Olcek _Contest_DL_2025_FINAL_V5.ipynb
Save failed
File Edit View Insert Runtime Tools Help
Commands
+ Code
+ Text
Run all
RAM
Disk
Share

Welcome to the Math Question Answer Verification Competition! 🚀

The goal is to fine-tune a Llama-3-8B model to predict if a given solution to a math problem is correct or not. Your model should output True if the solution is correct, and False otherwise.

This notebook is a starter guide designed to get you up and running quickly. We'll walk through a simplified training process using a small subset of the data (5,000 examples) and lightweight parameters. The main goal here is to understand the complete workflow, from loading data to generating a submission file, not to achieve a top score.

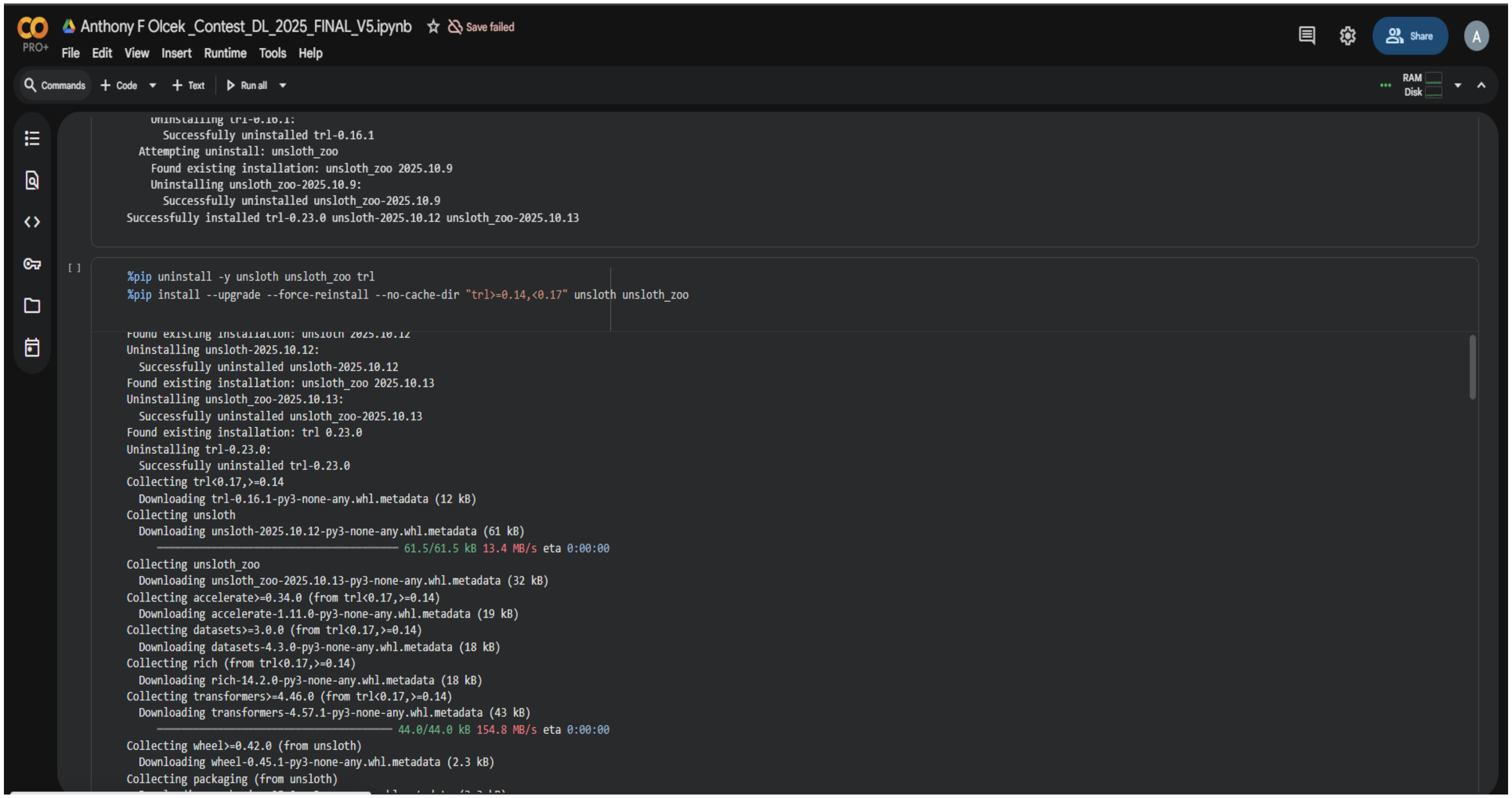
Good luck, and have fun! 🎉

Step 1: Install Necessary Libraries

First, we need to install the required Python libraries. We'll be using the unsloth library, which provides highly efficient, memory-saving training methods for large language models, making it possible to fine-tune powerful models on a single free-tier GPU. We'll also install xformers for further optimization.

```
[ ]
# @title
# %%capture
!pip install "unsloth[colab-new] @ git+https://github.com/unslothai/unsloth.git"
# !pip install --no-deps "xformers<0.0.26" "trl<0.9.0" "peft<0.12.0" "accelerate<0.32.0" "bitsandbytes<0.44.0" "transformers<4.43.0"
```

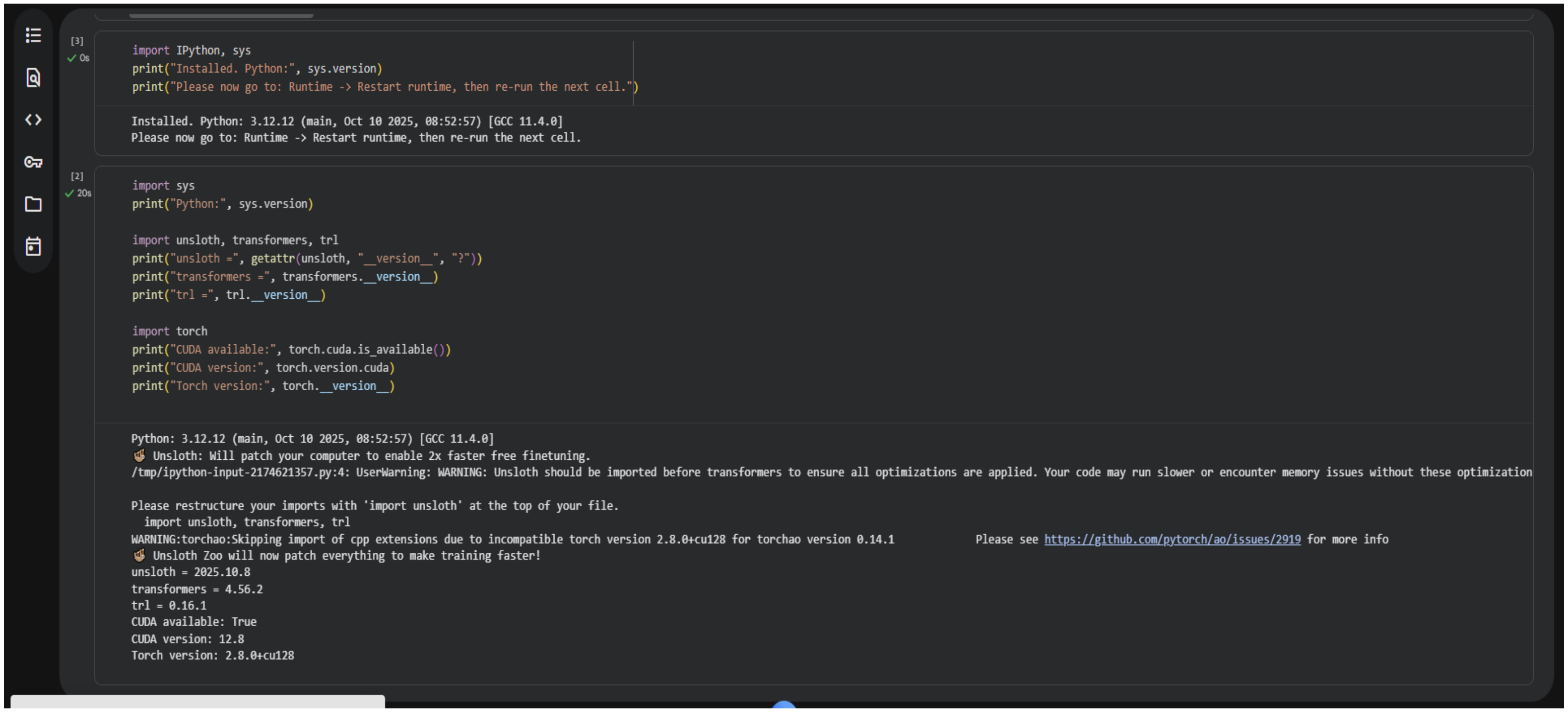
```
Collecting unsloth@ git+https://github.com/unslothai/unsloth.git (from unsloth[colab-new]@ git+https://github.com/unslothai/unsloth.git)
  Cloning https://github.com/unslothai/unsloth.git to /tmp/pip-install-hba6zg_g/unsloth_567f5accd3674aebb5783a76959fb1e4
  Running command git clone --filter=blob:none --quiet https://github.com/unslothai/unsloth.git /tmp/pip-install-hba6zg_g/unsloth_567f5accd3674aebb5783a76959fb1e4
  Resolved https://github.com/unslothai/unsloth.git to commit d707bd43b4e883b521761d525be2fae428fe5980
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
Collecting unsloth_zoo>=2025.10.13 (from unsloth@ git+https://github.com/unslothai/unsloth.git->unsloth[colab-new]@ git+https://github.com/unslothai/unsloth.git)
  Using cached unsloth_zoo-2025.10.13-py3-none-any.whl.metadata (32 kB)
```


Anthony F Olcek _Contest_DL_2025_FINAL_V5.ipynb
Save failed
File Edit View Insert Runtime Tools Help
Commands
Code
Text
Run all
Share
RAM
Disk

```
      Uninstalling trl-0.16.1:
        Successfully uninstalled trl-0.16.1
    Attempting uninstall: unsloth_zoo
      Found existing installation: unsloth_zoo 2025.10.9
      Uninstalling unsloth_zoo-2025.10.9:
        Successfully uninstalled unsloth_zoo-2025.10.9
Successfully installed trl-0.23.0 unsloth-2025.10.12 unsloth_zoo-2025.10.13
```

```
%pip uninstall -y unsloth unsloth_zoo trl
%pip install --upgrade --force-reinstall --no-cache-dir "trl>=0.14,<0.17" unsloth unsloth_zoo
```

```
Found existing installation: unsloth 2025.10.12
Uninstalling unsloth-2025.10.12:
  Successfully uninstalled unsloth-2025.10.12
Found existing installation: unsloth_zoo 2025.10.13
Uninstalling unsloth_zoo-2025.10.13:
  Successfully uninstalled unsloth_zoo-2025.10.13
Found existing installation: trl 0.23.0
Uninstalling trl-0.23.0:
  Successfully uninstalled trl-0.23.0
Collecting trl<0.17,>=0.14
  Downloading trl-0.16.1-py3-none-any.whl.metadata (12 kB)
Collecting unsloth
  Downloading unsloth-2025.10.12-py3-none-any.whl.metadata (61 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 61.5/61.5 kB 13.4 MB/s eta 0:00:00
Collecting unsloth_zoo
  Downloading unsloth_zoo-2025.10.13-py3-none-any.whl.metadata (32 kB)
Collecting accelerate>=0.34.0 (from trl<0.17,>=0.14)
  Downloading accelerate-1.11.0-py3-none-any.whl.metadata (19 kB)
Collecting datasets>=3.0.0 (from trl<0.17,>=0.14)
  Downloading datasets-4.3.0-py3-none-any.whl.metadata (18 kB)
Collecting rich (from trl<0.17,>=0.14)
  Downloading rich-14.2.0-py3-none-any.whl.metadata (18 kB)
Collecting transformers>=4.46.0 (from trl<0.17,>=0.14)
  Downloading transformers-4.57.1-py3-none-any.whl.metadata (43 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 44.0/44.0 kB 154.8 MB/s eta 0:00:00
Collecting wheel>=0.42.0 (from unsloth)
  Downloading wheel-0.45.1-py3-none-any.whl.metadata (2.3 kB)
Collecting packaging (from unsloth)
```



[3]

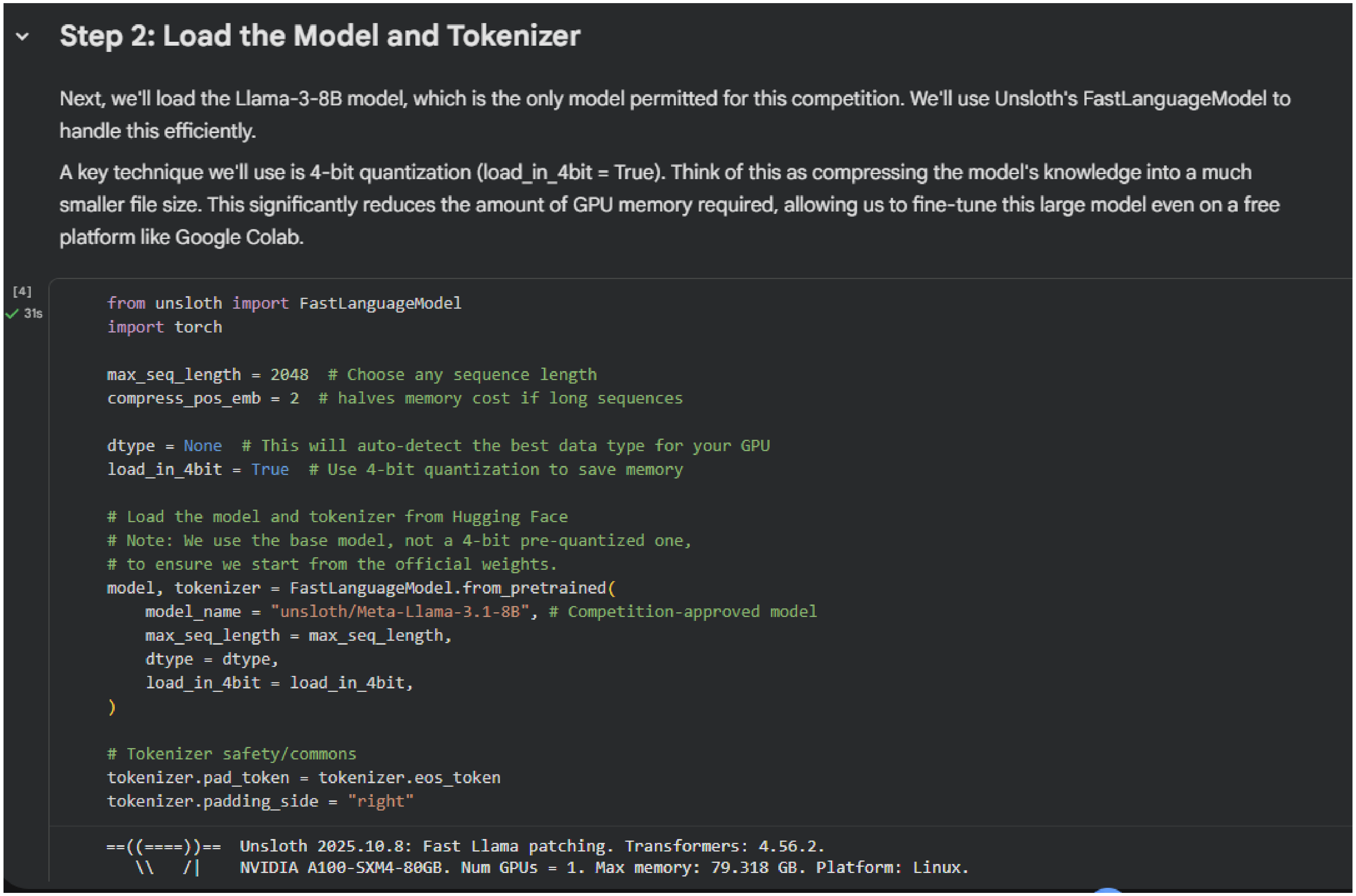
```
import IPython, sys
print("Installed. Python:", sys.version)
print("Please now go to: Runtime -> Restart runtime, then re-run the next cell.")
```

```
Installed. Python: 3.12.12 (main, Oct 10 2025, 08:52:57) [GCC 11.4.0]
Please now go to: Runtime -> Restart runtime, then re-run the next cell.
```

[2]

```
import sys
print("Python:", sys.version)

import unsloth, transformers, trl
print("unsloth =", getattr(unsloth, "__version__", "?"))
print("transformers =", transformers.__version__)
print("trl =", trl.__version__)

import torch
print("CUDA available:", torch.cuda.is_available())
print("CUDA version:", torch.version.cuda)
print("Torch version:", torch.__version__)
```

```
Python: 3.12.12 (main, Oct 10 2025, 08:52:57) [GCC 11.4.0]
🦥 Unsloth: Will patch your computer to enable 2x faster free finetuning.
/tmp/ipython-input-2174621357.py:4: UserWarning: WARNING: Unsloth should be imported before transformers to ensure all optimizations are applied. Your code may run slower or encounter memory issues without these optimization

Please restructure your imports with 'import unsloth' at the top of your file.
  import unsloth, transformers, trl
WARNING:torchao:Skipping import of cpp extensions due to incompatible torch version 2.8.0+cu128 for torchao version 0.14.1            Please see https://github.com/pytorch/ao/issues/2919 for more info
🦥 Unsloth Zoo will now patch everything to make training faster!
unsloth = 2025.10.8
transformers = 4.56.2
trl = 0.16.1
CUDA available: True
CUDA version: 12.8
Torch version: 2.8.0+cu128
```



Step 2: Load the Model and Tokenizer

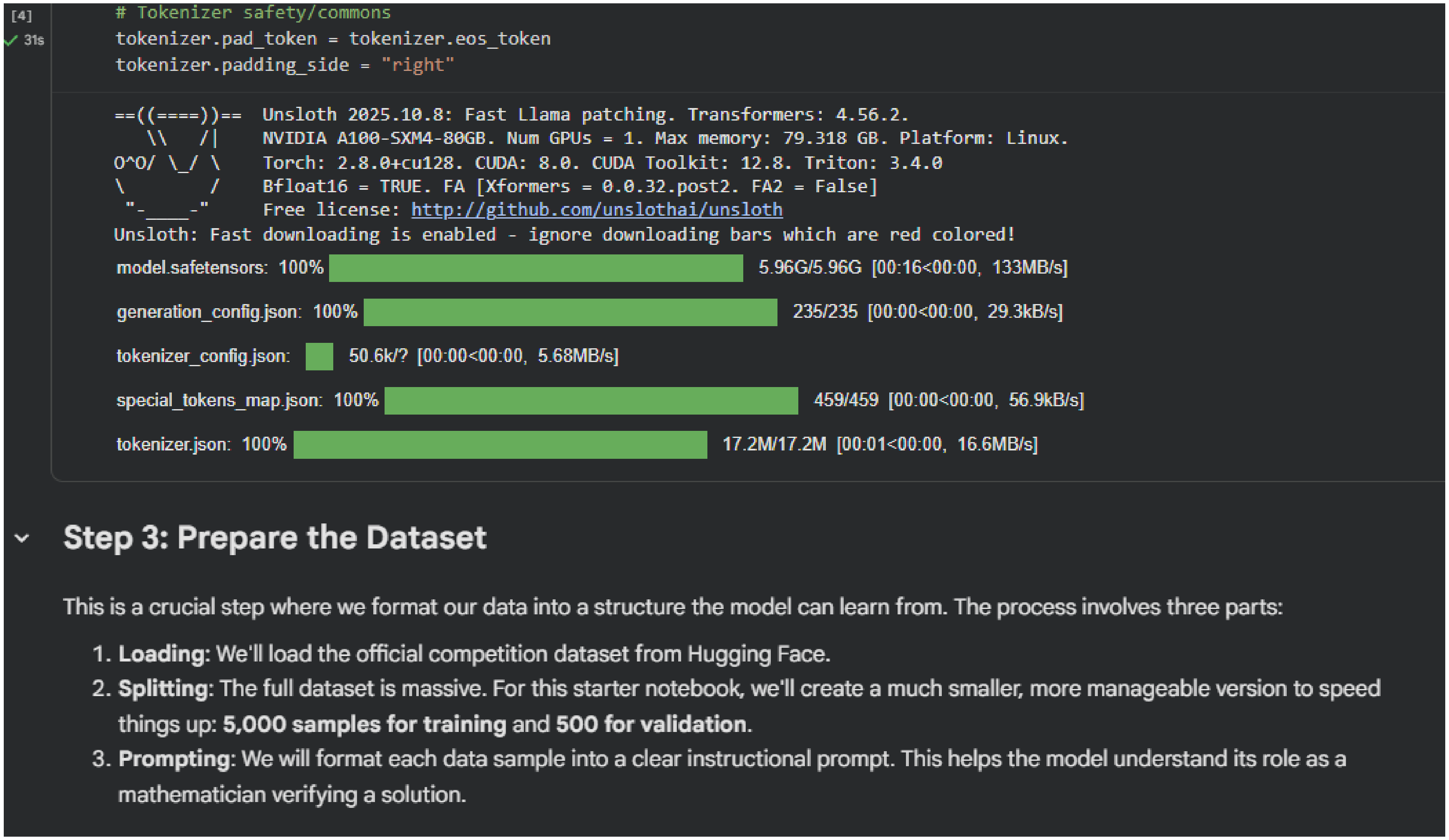
Next, we'll load the Llama-3-8B model, which is the only model permitted for this competition. We'll use Unsloth's FastLanguageModel to handle this efficiently.

A key technique we'll use is 4-bit quantization (load_in_4bit = True). Think of this as compressing the model's knowledge into a much smaller file size. This significantly reduces the amount of GPU memory required, allowing us to fine-tune this large model even on a free platform like Google Colab.

[4]
✓ 31s

```python
from unsloth import FastLanguageModel
import torch

max_seq_length = 2048  # Choose any sequence length
compress_pos_emb = 2  # halves memory cost if long sequences

dtype = None  # This will auto-detect the best data type for your GPU
load_in_4bit = True  # Use 4-bit quantization to save memory

# Load the model and tokenizer from Hugging Face
# Note: We use the base model, not a 4-bit pre-quantized one,
# to ensure we start from the official weights.
model, tokenizer = FastLanguageModel.from_pretrained(
    model_name = "unsloth/Meta-Llama-3.1-8B", # Competition-approved model
    max_seq_length = max_seq_length,
    dtype = dtype,
    load_in_4bit = load_in_4bit,
)

# Tokenizer safety/commons
tokenizer.pad_token = tokenizer.eos_token
tokenizer.padding_side = "right"
```

```
==((====))==  Unsloth 2025.10.8: Fast Llama patching. Transformers: 4.56.2.
   \\   /|    NVIDIA A100-SXM4-80GB. Num GPUs = 1. Max memory: 79.318 GB. Platform: Linux.
```



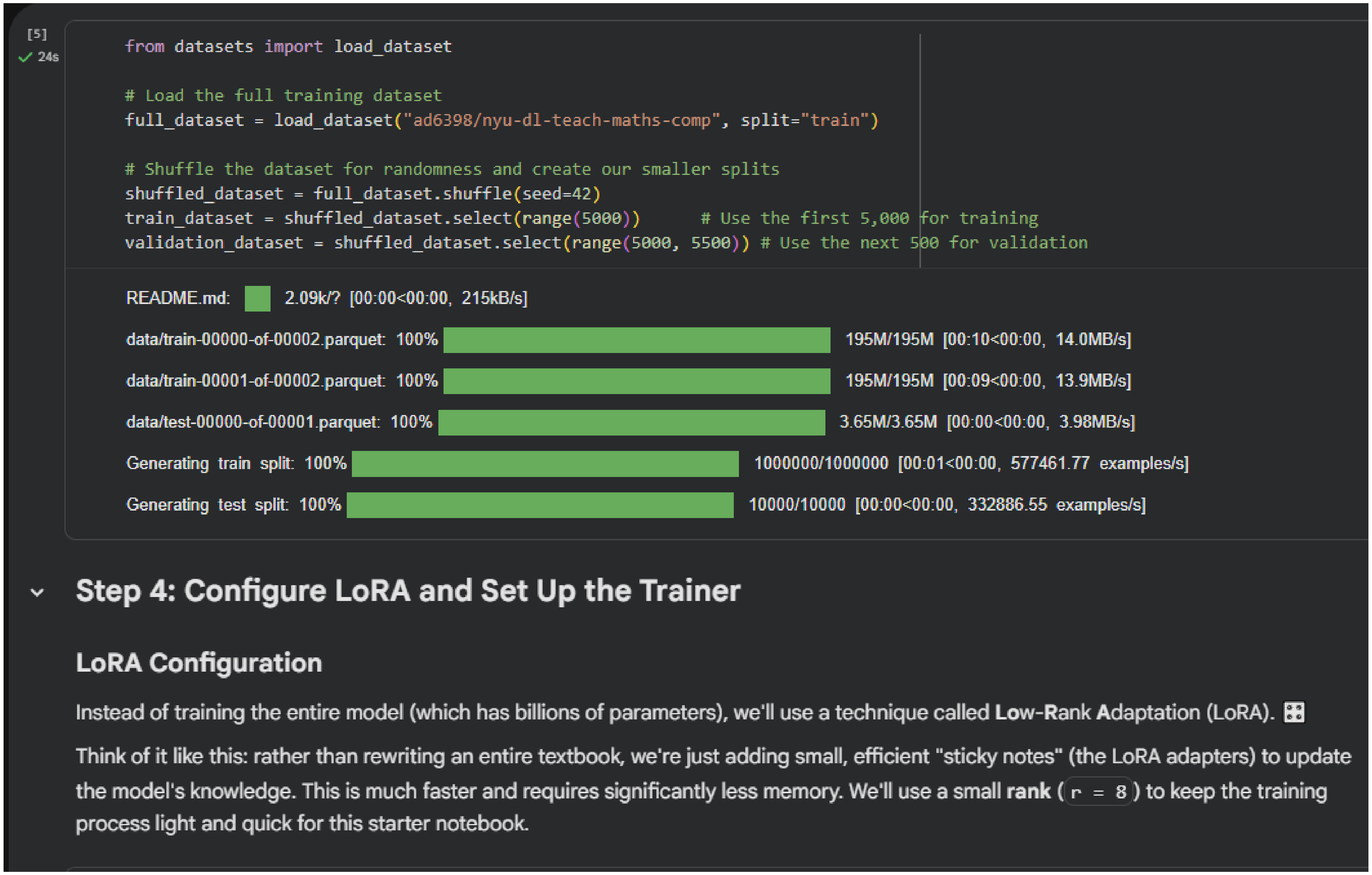
```
[4]
✓ 31s
    # Tokenizer safety/commons
    tokenizer.pad_token = tokenizer.eos_token
    tokenizer.padding_side = "right"
```

```
==((====))==  Unsloth 2025.10.8: Fast Llama patching. Transformers: 4.56.2.
   \\   /|    NVIDIA A100-SXM4-80GB. Num GPUs = 1. Max memory: 79.318 GB. Platform: Linux.
O^O/ \_/ \    Torch: 2.8.0+cu128. CUDA: 8.0. CUDA Toolkit: 12.8. Triton: 3.4.0
\        /    Bfloat16 = TRUE. FA [Xformers = 0.0.32.post2. FA2 = False]
 "-____-"     Free license: http://github.com/unslothai/unsloth
Unsloth: Fast downloading is enabled - ignore downloading bars which are red colored!
model.safetensors: 100%  5.96G/5.96G [00:16<00:00, 133MB/s]
generation_config.json: 100%  235/235 [00:00<00:00, 29.3kB/s]
tokenizer_config.json:  50.6k/? [00:00<00:00, 5.68MB/s]
special_tokens_map.json: 100%  459/459 [00:00<00:00, 56.9kB/s]
tokenizer.json: 100%  17.2M/17.2M [00:01<00:00, 16.6MB/s]
```

Step 3: Prepare the Dataset

This is a crucial step where we format our data into a structure the model can learn from. The process involves three parts:

1. **Loading**: We'll load the official competition dataset from Hugging Face.
2. **Splitting**: The full dataset is massive. For this starter notebook, we'll create a much smaller, more manageable version to speed things up: **5,000 samples for training** and **500 for validation**.
3. **Prompting**: We will format each data sample into a clear instructional prompt. This helps the model understand its role as a mathematician verifying a solution.


```
[5]
✓ 24s
    from datasets import load_dataset

    # Load the full training dataset
    full_dataset = load_dataset("ad6398/nyu-dl-teach-maths-comp", split="train")

    # Shuffle the dataset for randomness and create our smaller splits
    shuffled_dataset = full_dataset.shuffle(seed=42)
    train_dataset = shuffled_dataset.select(range(5000))      # Use the first 5,000 for training
    validation_dataset = shuffled_dataset.select(range(5000, 5500)) # Use the next 500 for validation
```

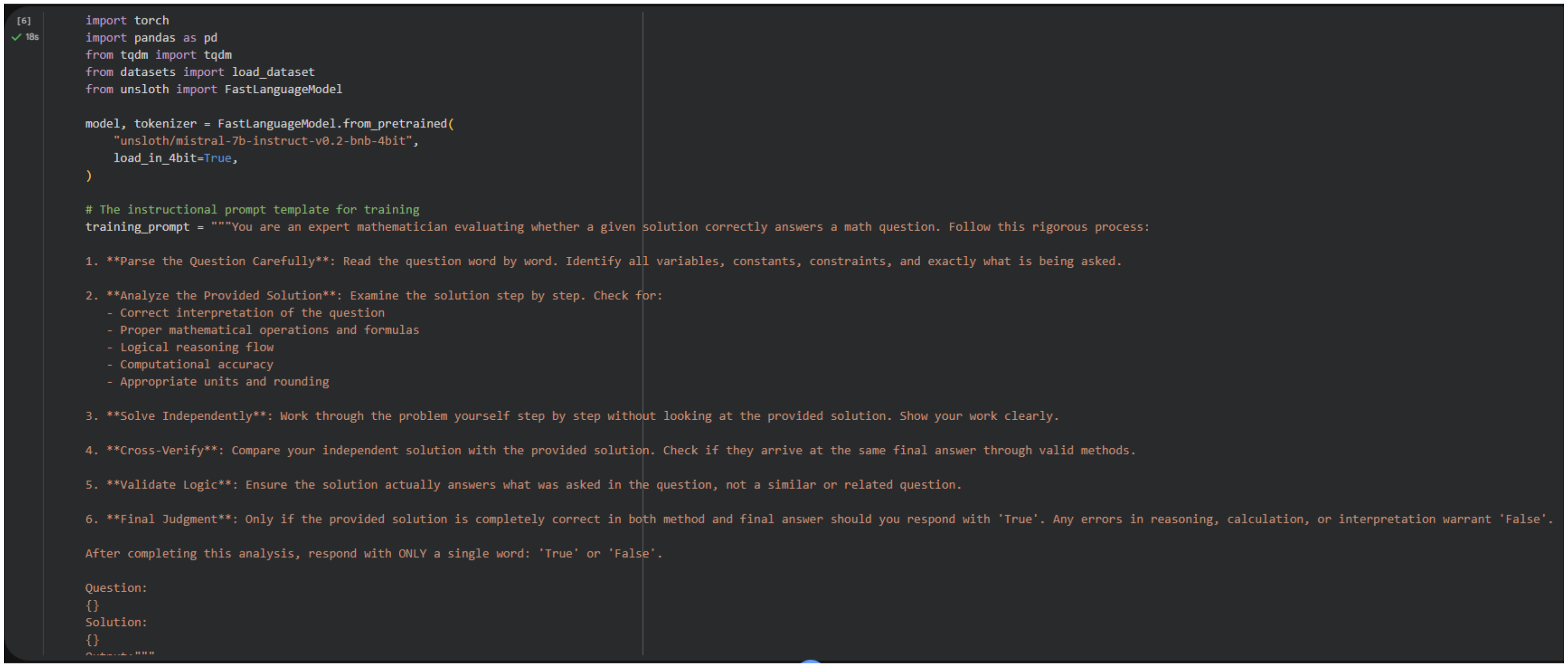
```
README.md:  2.09k/? [00:00<00:00, 215kB/s]
data/train-00000-of-00002.parquet: 100%  195M/195M [00:10<00:00, 14.0MB/s]
data/train-00001-of-00002.parquet: 100%  195M/195M [00:09<00:00, 13.9MB/s]
data/test-00000-of-00001.parquet: 100%  3.65M/3.65M [00:00<00:00, 3.98MB/s]
Generating train split: 100%  1000000/1000000 [00:01<00:00, 577461.77 examples/s]
Generating test split: 100%  10000/10000 [00:00<00:00, 332886.55 examples/s]
```

Step 4: Configure LoRA and Set Up the Trainer

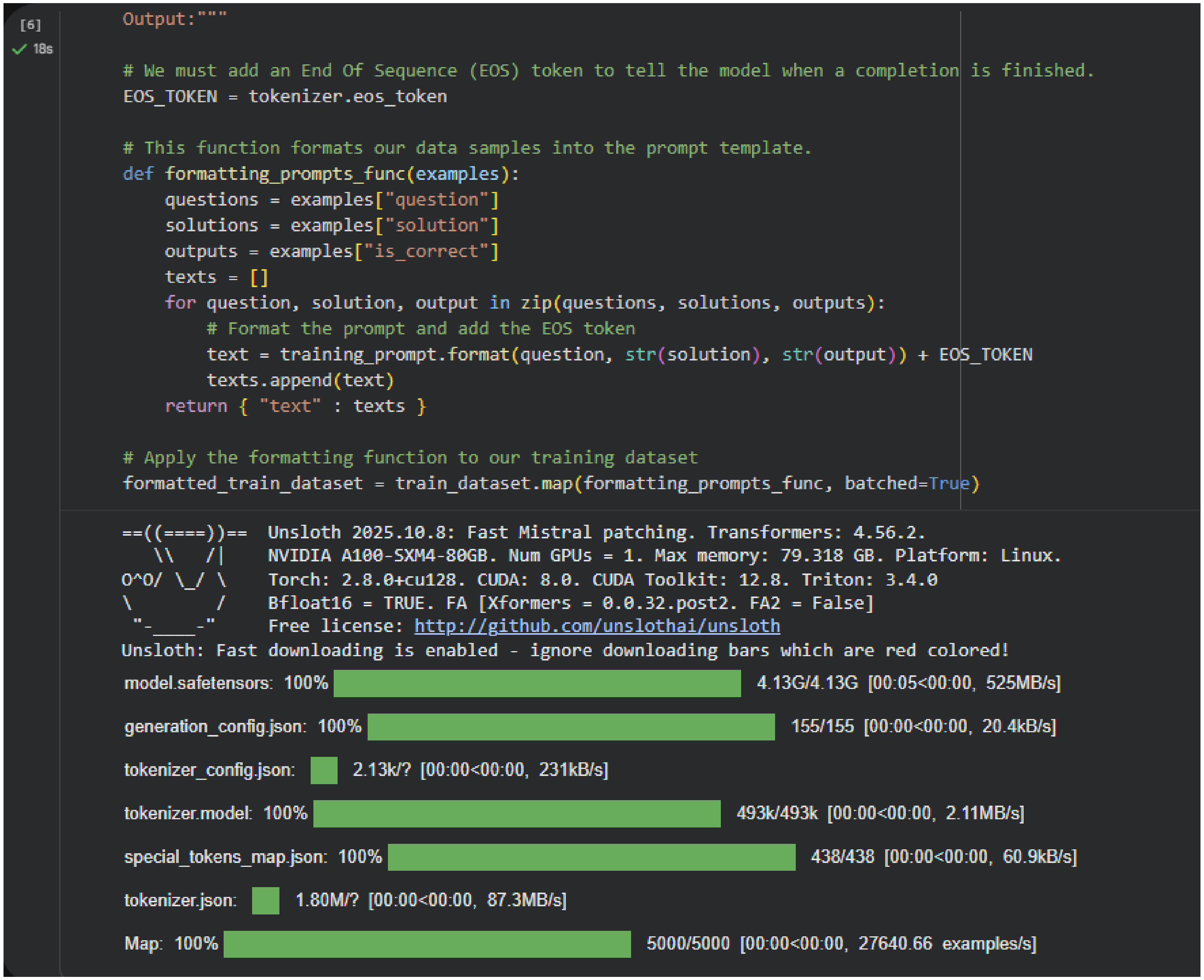
LoRA Configuration

Instead of training the entire model (which has billions of parameters), we'll use a technique called **Lo**w-**R**ank **A**daptation (LoRA).

Think of it like this: rather than rewriting an entire textbook, we're just adding small, efficient "sticky notes" (the LoRA adapters) to update the model's knowledge. This is much faster and requires significantly less memory. We'll use a small **rank** (`r = 8`) to keep the training process light and quick for this starter notebook.

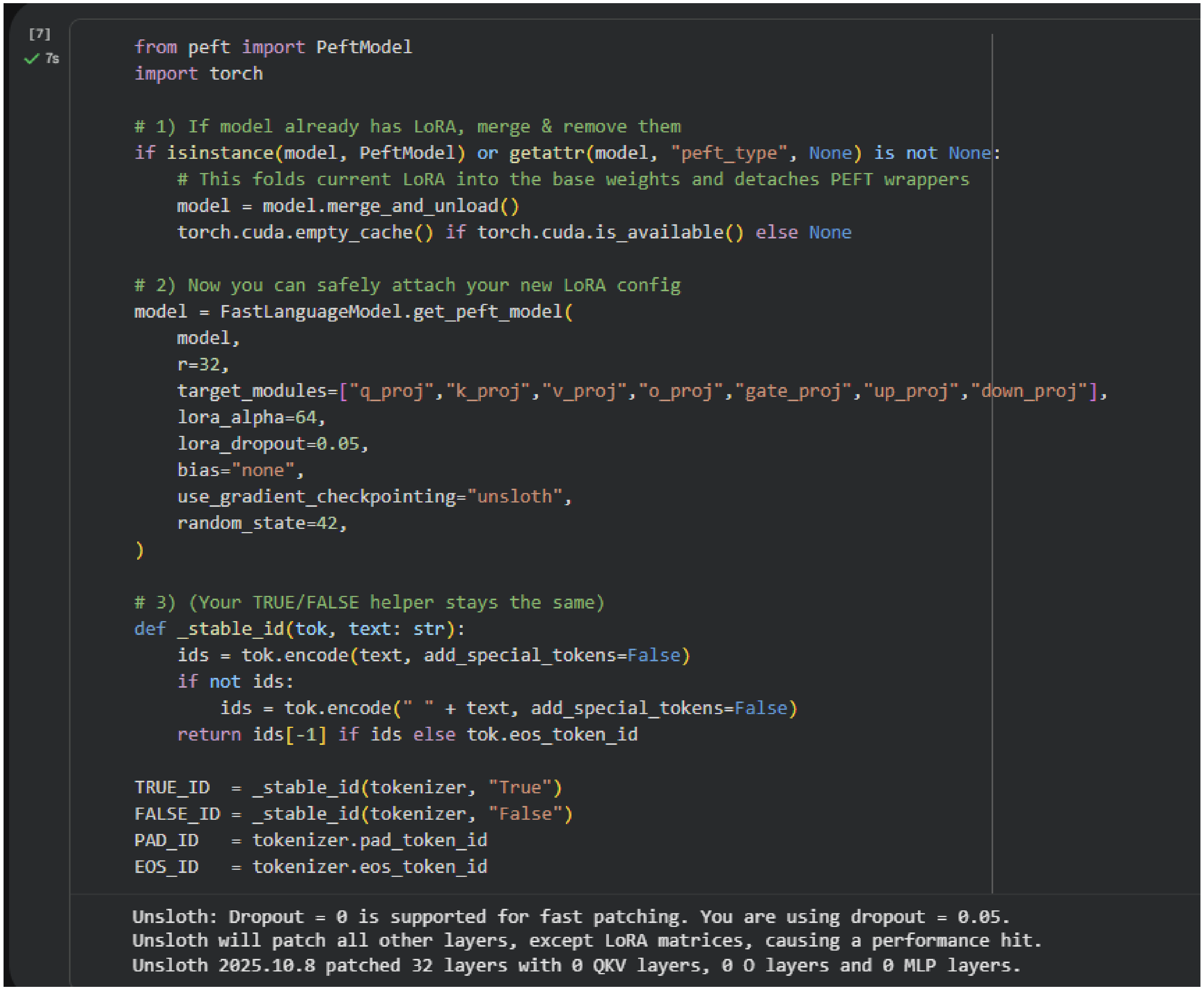


```python
import torch
import pandas as pd
from tqdm import tqdm
from datasets import load_dataset
from unsloth import FastLanguageModel

model, tokenizer = FastLanguageModel.from_pretrained(
    "unsloth/mistral-7b-instruct-v0.2-bnb-4bit",
    load_in_4bit=True,
)

# The instructional prompt template for training
training_prompt = """You are an expert mathematician evaluating whether a given solution correctly answers a math question. Follow this rigorous process:

1. **Parse the Question Carefully**: Read the question word by word. Identify all variables, constants, constraints, and exactly what is being asked.

2. **Analyze the Provided Solution**: Examine the solution step by step. Check for:
   - Correct interpretation of the question
   - Proper mathematical operations and formulas
   - Logical reasoning flow
   - Computational accuracy
   - Appropriate units and rounding

3. **Solve Independently**: Work through the problem yourself step by step without looking at the provided solution. Show your work clearly.

4. **Cross-Verify**: Compare your independent solution with the provided solution. Check if they arrive at the same final answer through valid methods.

5. **Validate Logic**: Ensure the solution actually answers what was asked in the question, not a similar or related question.

6. **Final Judgment**: Only if the provided solution is completely correct in both method and final answer should you respond with 'True'. Any errors in reasoning, calculation, or interpretation warrant 'False'.

After completing this analysis, respond with ONLY a single word: 'True' or 'False'.

Question:
{}
Solution:
{}
Output:"""
```



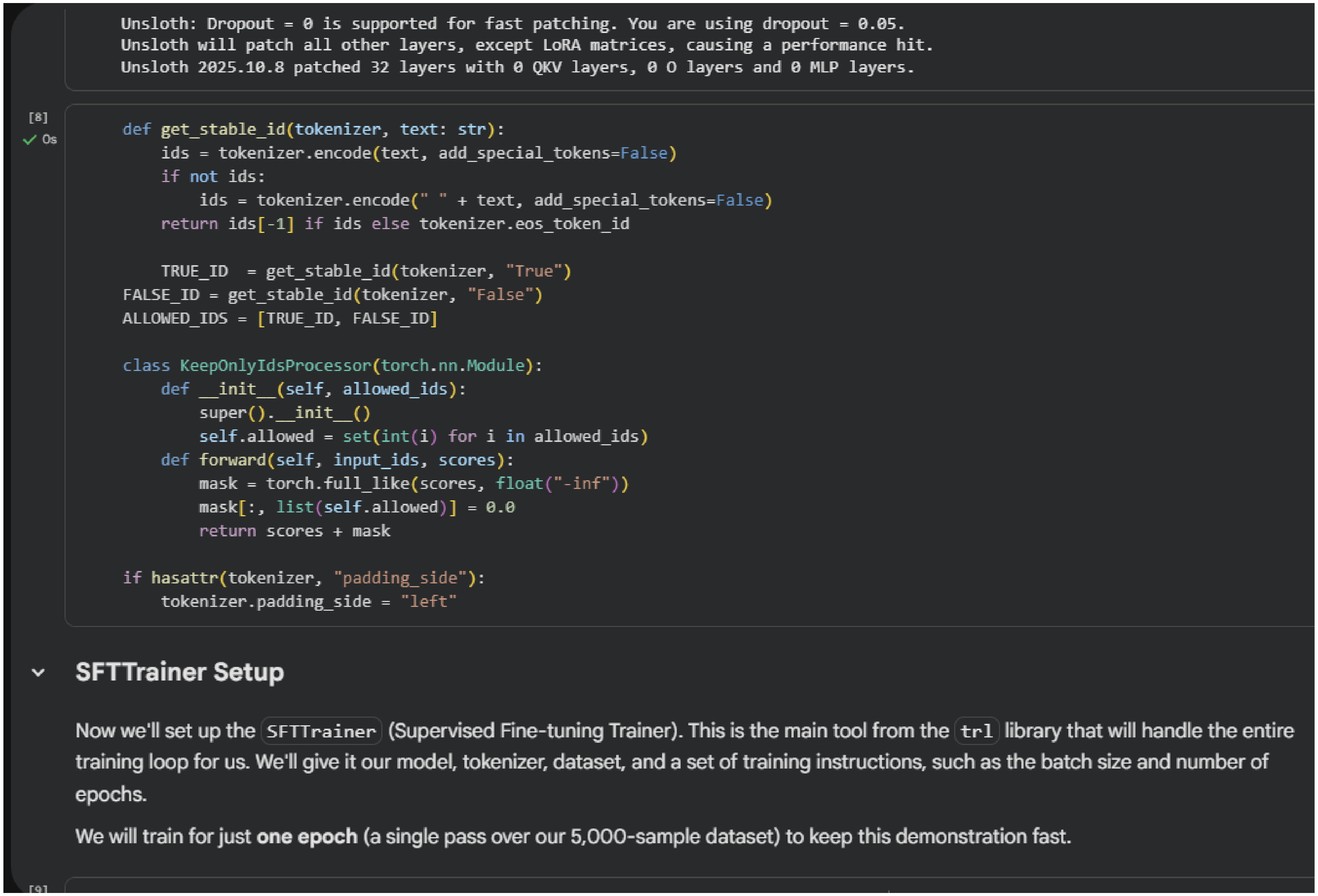
```
[6]
✓ 18s
    Output:"""

    # We must add an End Of Sequence (EOS) token to tell the model when a completion is finished.
    EOS_TOKEN = tokenizer.eos_token

    # This function formats our data samples into the prompt template.
    def formatting_prompts_func(examples):
        questions = examples["question"]
        solutions = examples["solution"]
        outputs = examples["is_correct"]
        texts = []
        for question, solution, output in zip(questions, solutions, outputs):
            # Format the prompt and add the EOS token
            text = training_prompt.format(question, str(solution), str(output)) + EOS_TOKEN
            texts.append(text)
        return { "text" : texts }

    # Apply the formatting function to our training dataset
    formatted_train_dataset = train_dataset.map(formatting_prompts_func, batched=True)
```

```
==((====))==  Unsloth 2025.10.8: Fast Mistral patching. Transformers: 4.56.2.
   \\   /|    NVIDIA A100-SXM4-80GB. Num GPUs = 1. Max memory: 79.318 GB. Platform: Linux.
O^O/ \_/ \    Torch: 2.8.0+cu128. CUDA: 8.0. CUDA Toolkit: 12.8. Triton: 3.4.0
\        /    Bfloat16 = TRUE. FA [Xformers = 0.0.32.post2. FA2 = False]
 "-____-"     Free license: http://github.com/unslothai/unsloth
Unsloth: Fast downloading is enabled - ignore downloading bars which are red colored!
model.safetensors: 100% 4.13G/4.13G [00:05<00:00, 525MB/s]
generation_config.json: 100% 155/155 [00:00<00:00, 20.4kB/s]
tokenizer_config.json: 2.13k/? [00:00<00:00, 231kB/s]
tokenizer.model: 100% 493k/493k [00:00<00:00, 2.11MB/s]
special_tokens_map.json: 100% 438/438 [00:00<00:00, 60.9kB/s]
tokenizer.json: 1.80M/? [00:00<00:00, 87.3MB/s]
Map: 100% 5000/5000 [00:00<00:00, 27640.66 examples/s]
```



```python
from peft import PeftModel
import torch

# 1) If model already has LoRA, merge & remove them
if isinstance(model, PeftModel) or getattr(model, "peft_type", None) is not None:
    # This folds current LoRA into the base weights and detaches PEFT wrappers
    model = model.merge_and_unload()
    torch.cuda.empty_cache() if torch.cuda.is_available() else None

# 2) Now you can safely attach your new LoRA config
model = FastLanguageModel.get_peft_model(
    model,
    r=32,
    target_modules=["q_proj","k_proj","v_proj","o_proj","gate_proj","up_proj","down_proj"],
    lora_alpha=64,
    lora_dropout=0.05,
    bias="none",
    use_gradient_checkpointing="unsloth",
    random_state=42,
)

# 3) (Your TRUE/FALSE helper stays the same)
def _stable_id(tok, text: str):
    ids = tok.encode(text, add_special_tokens=False)
    if not ids:
        ids = tok.encode(" " + text, add_special_tokens=False)
    return ids[-1] if ids else tok.eos_token_id

TRUE_ID  = _stable_id(tokenizer, "True")
FALSE_ID = _stable_id(tokenizer, "False")
PAD_ID   = tokenizer.pad_token_id
EOS_ID   = tokenizer.eos_token_id
```

```
Unsloth: Dropout = 0 is supported for fast patching. You are using dropout = 0.05.
Unsloth will patch all other layers, except LoRA matrices, causing a performance hit.
Unsloth 2025.10.8 patched 32 layers with 0 QKV layers, 0 O layers and 0 MLP layers.
```



```
Unsloth: Dropout = 0 is supported for fast patching. You are using dropout = 0.05.
Unsloth will patch all other layers, except LoRA matrices, causing a performance hit.
Unsloth 2025.10.8 patched 32 layers with 0 QKV layers, 0 O layers and 0 MLP layers.
```

[8]
✓ 0s

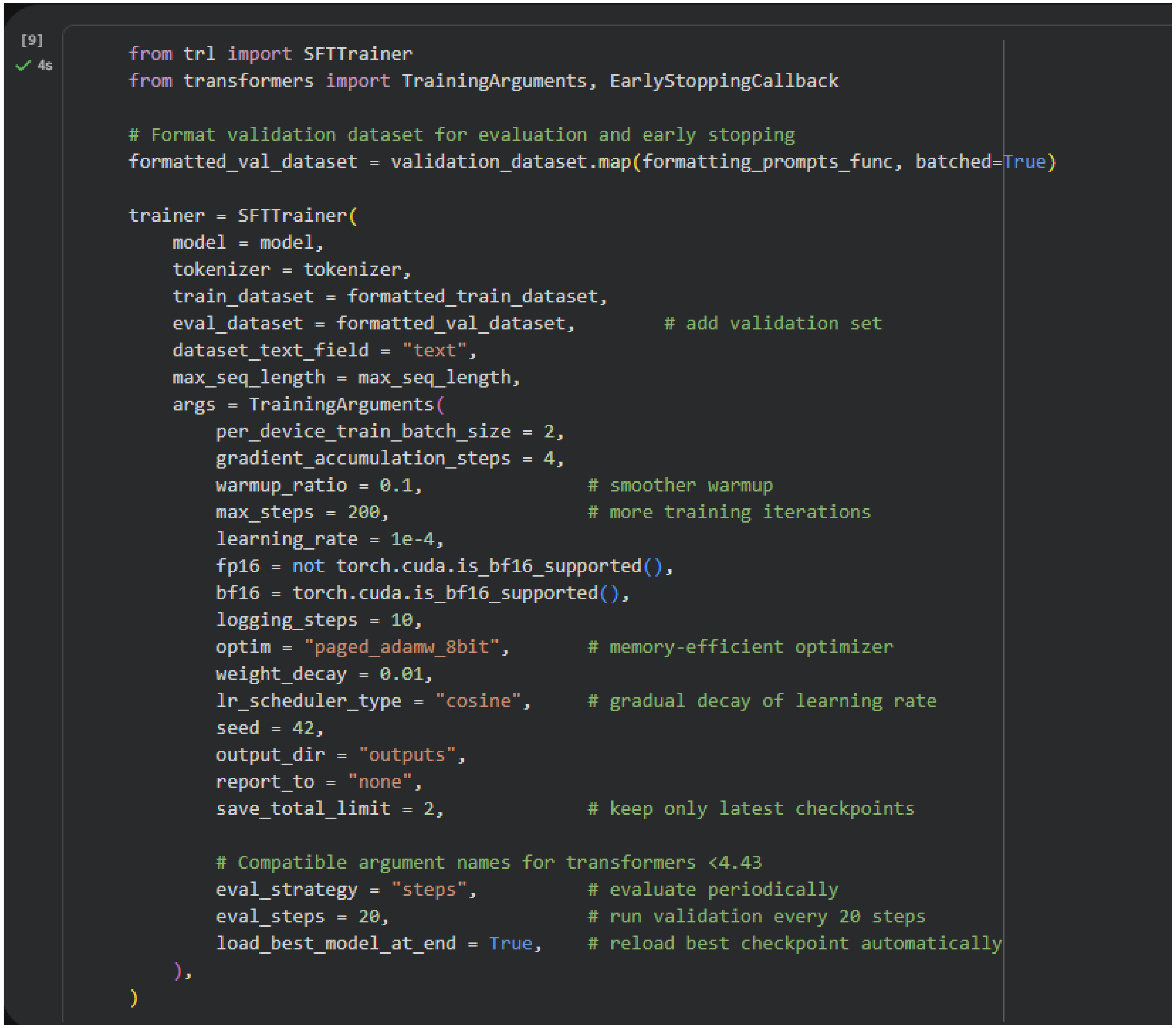
```
def get_stable_id(tokenizer, text: str):
    ids = tokenizer.encode(text, add_special_tokens=False)
    if not ids:
        ids = tokenizer.encode(" " + text, add_special_tokens=False)
    return ids[-1] if ids else tokenizer.eos_token_id

    TRUE_ID  = get_stable_id(tokenizer, "True")
FALSE_ID = get_stable_id(tokenizer, "False")
ALLOWED_IDS = [TRUE_ID, FALSE_ID]

class KeepOnlyIdsProcessor(torch.nn.Module):
    def __init__(self, allowed_ids):
        super().__init__()
        self.allowed = set(int(i) for i in allowed_ids)
    def forward(self, input_ids, scores):
        mask = torch.full_like(scores, float("-inf"))
        mask[:, list(self.allowed)] = 0.0
        return scores + mask

if hasattr(tokenizer, "padding_side"):
    tokenizer.padding_side = "left"
```

SFTTrainer Setup

Now we'll set up the `SFTTrainer` (Supervised Fine-tuning Trainer). This is the main tool from the `trl` library that will handle the entire training loop for us. We'll give it our model, tokenizer, dataset, and a set of training instructions, such as the batch size and number of epochs.

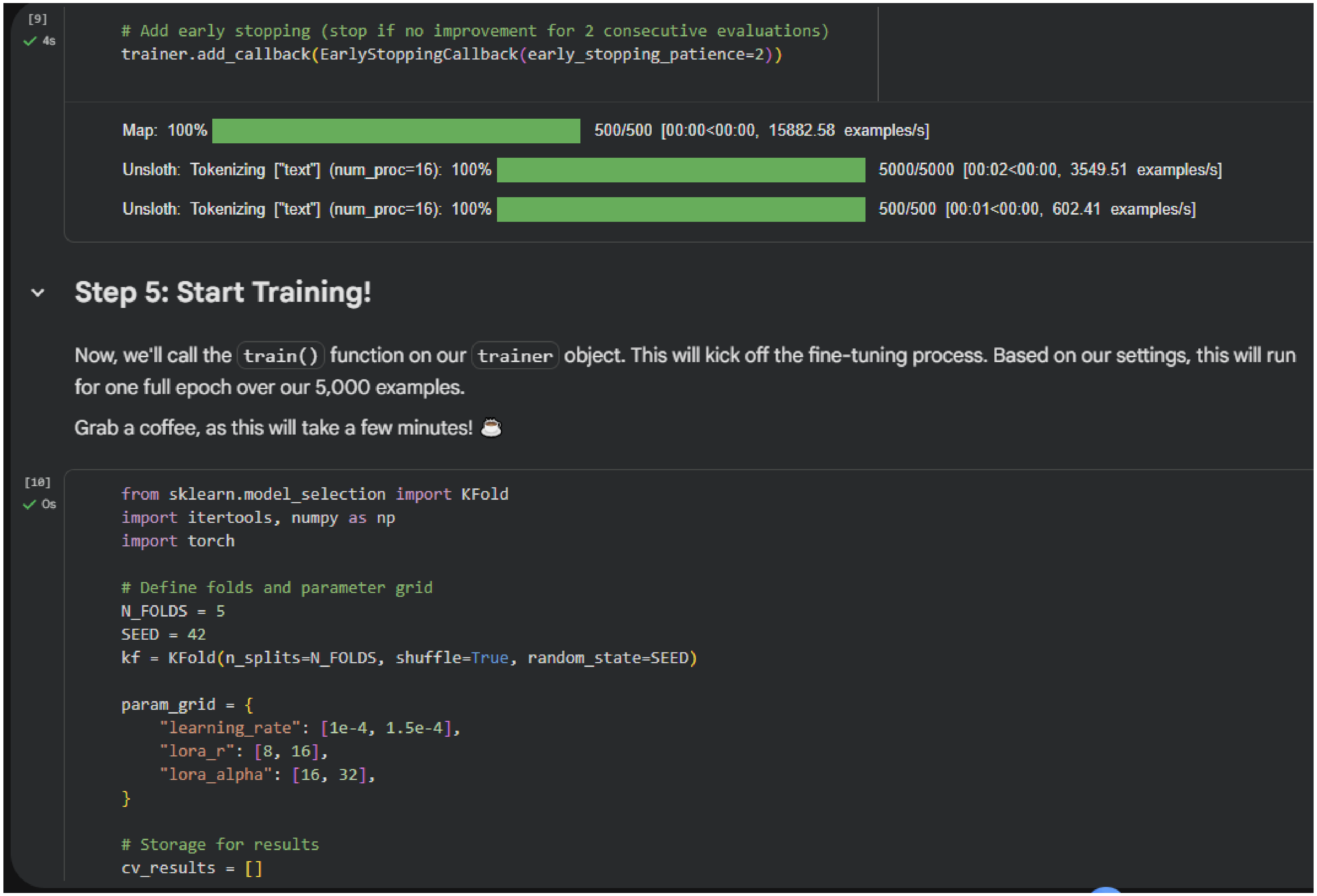
We will train for just **one epoch** (a single pass over our 5,000-sample dataset) to keep this demonstration fast.

[9]



[9]
✓ 4s

```python
from trl import SFTTrainer
from transformers import TrainingArguments, EarlyStoppingCallback

# Format validation dataset for evaluation and early stopping
formatted_val_dataset = validation_dataset.map(formatting_prompts_func, batched=True)

trainer = SFTTrainer(
    model = model,
    tokenizer = tokenizer,
    train_dataset = formatted_train_dataset,
    eval_dataset = formatted_val_dataset,        # add validation set
    dataset_text_field = "text",
    max_seq_length = max_seq_length,
    args = TrainingArguments(
        per_device_train_batch_size = 2,
        gradient_accumulation_steps = 4,
        warmup_ratio = 0.1,                  # smoother warmup
        max_steps = 200,                     # more training iterations
        learning_rate = 1e-4,
        fp16 = not torch.cuda.is_bf16_supported(),
        bf16 = torch.cuda.is_bf16_supported(),
        logging_steps = 10,
        optim = "paged_adamw_8bit",          # memory-efficient optimizer
        weight_decay = 0.01,
        lr_scheduler_type = "cosine",        # gradual decay of learning rate
        seed = 42,
        output_dir = "outputs",
        report_to = "none",
        save_total_limit = 2,                # keep only latest checkpoints

        # Compatible argument names for transformers <4.43
        eval_strategy = "steps",             # evaluate periodically
        eval_steps = 20,                     # run validation every 20 steps
        load_best_model_at_end = True,       # reload best checkpoint automatically
    ),
)
```



```
[9]
# Add early stopping (stop if no improvement for 2 consecutive evaluations)
trainer.add_callback(EarlyStoppingCallback(early_stopping_patience=2))
```

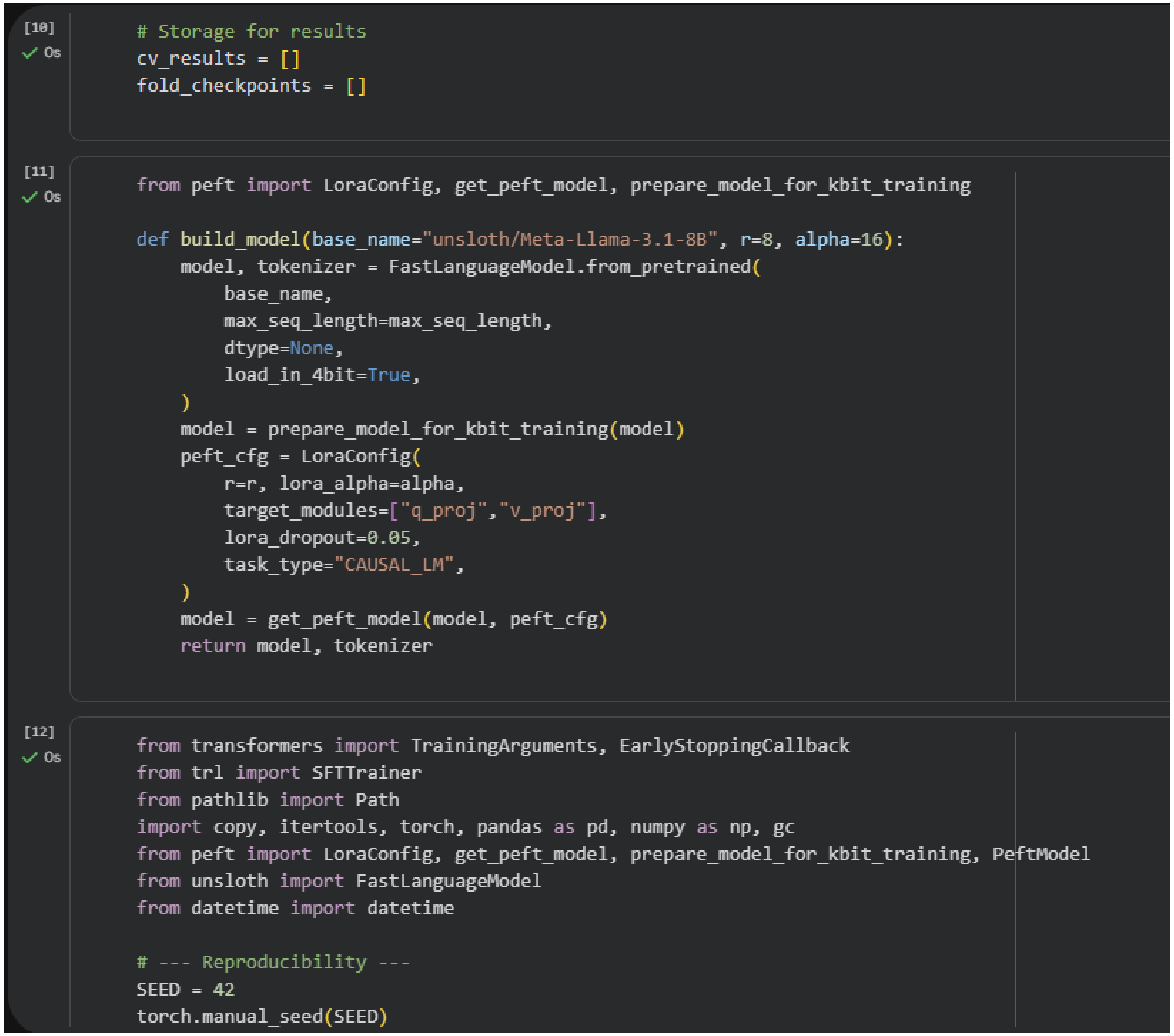
```
Map: 100% 500/500 [00:00<00:00, 15882.58 examples/s]
Unsloth: Tokenizing ["text"] (num_proc=16): 100% 5000/5000 [00:02<00:00, 3549.51 examples/s]
Unsloth: Tokenizing ["text"] (num_proc=16): 100% 500/500 [00:01<00:00, 602.41 examples/s]
```

Step 5: Start Training!

Now, we'll call the `train()` function on our `trainer` object. This will kick off the fine-tuning process. Based on our settings, this will run for one full epoch over our 5,000 examples.

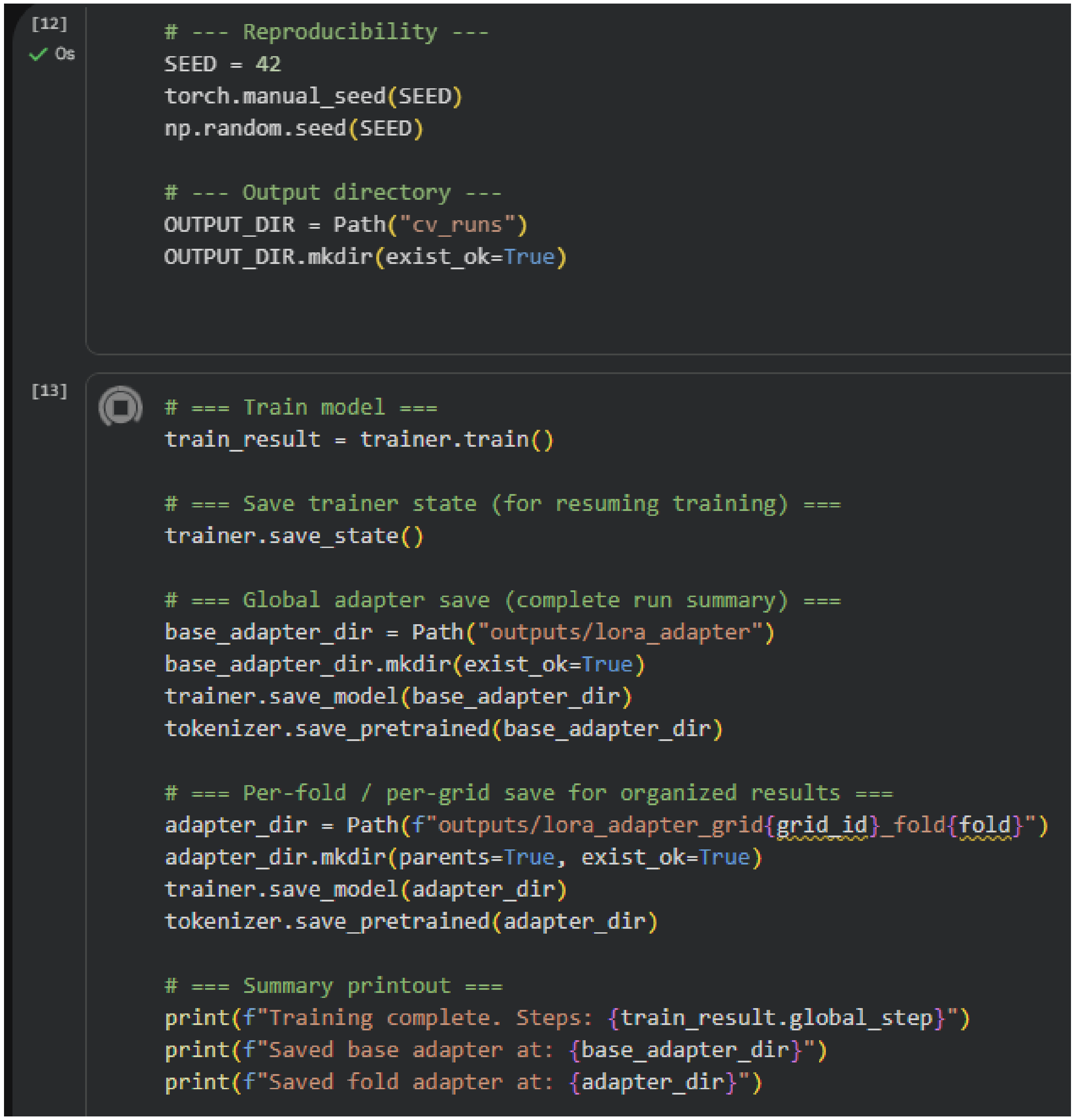
Grab a coffee, as this will take a few minutes! ☕

```
[10]
from sklearn.model_selection import KFold
import itertools, numpy as np
import torch

# Define folds and parameter grid
N_FOLDS = 5
SEED = 42
kf = KFold(n_splits=N_FOLDS, shuffle=True, random_state=SEED)

param_grid = {
    "learning_rate": [1e-4, 1.5e-4],
    "lora_r": [8, 16],
    "lora_alpha": [16, 32],
}

# Storage for results
cv_results = []
```


```
[10]
✓ 0s
    # Storage for results
    cv_results = []
    fold_checkpoints = []
```

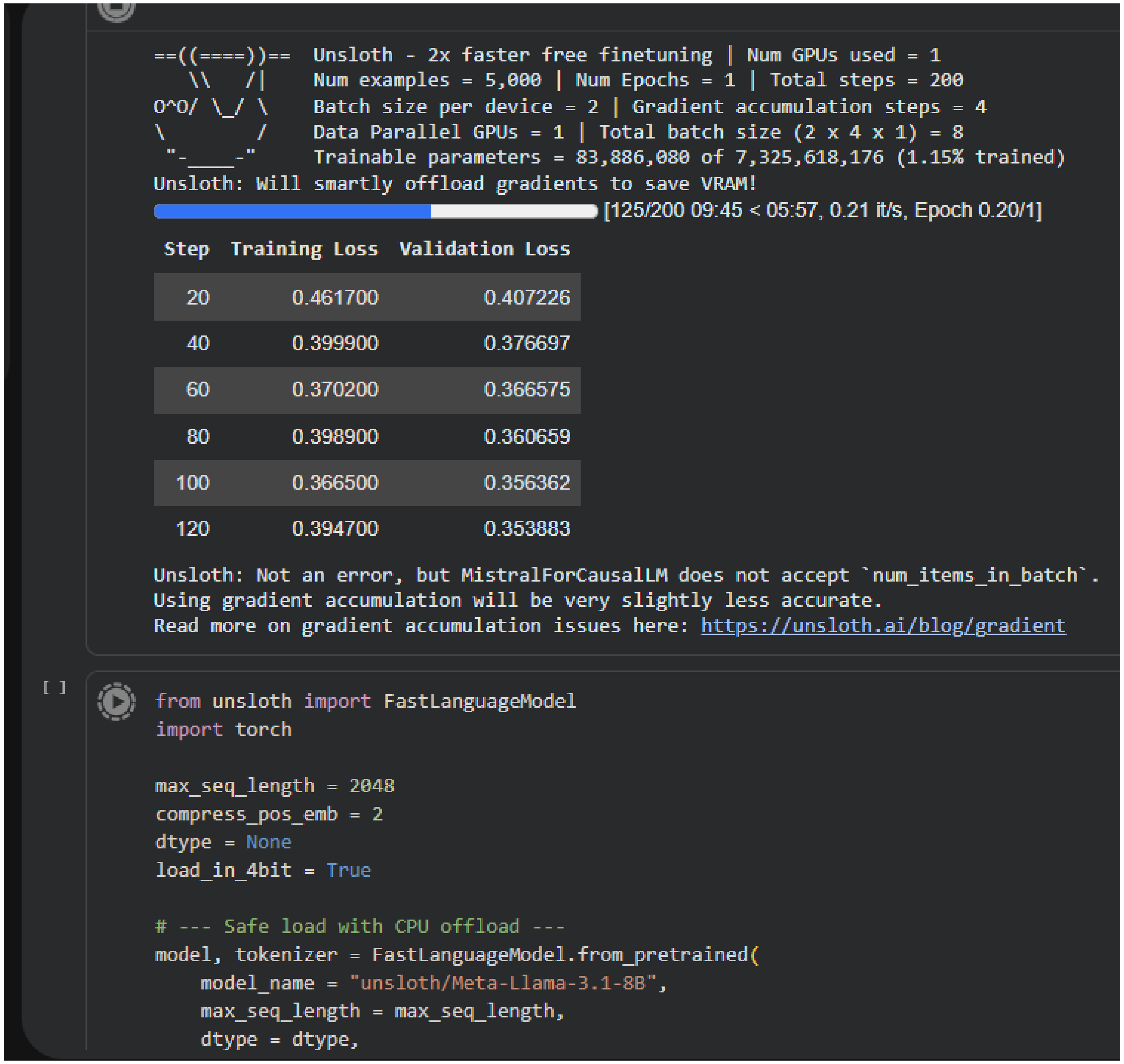
```
[11]
✓ 0s
    from peft import LoraConfig, get_peft_model, prepare_model_for_kbit_training

    def build_model(base_name="unsloth/Meta-Llama-3.1-8B", r=8, alpha=16):
        model, tokenizer = FastLanguageModel.from_pretrained(
            base_name,
            max_seq_length=max_seq_length,
            dtype=None,
            load_in_4bit=True,
        )
        model = prepare_model_for_kbit_training(model)
        peft_cfg = LoraConfig(
            r=r, lora_alpha=alpha,
            target_modules=["q_proj","v_proj"],
            lora_dropout=0.05,
            task_type="CAUSAL_LM",
        )
        model = get_peft_model(model, peft_cfg)
        return model, tokenizer
```

```
[12]
✓ 0s
    from transformers import TrainingArguments, EarlyStoppingCallback
    from trl import SFTTrainer
    from pathlib import Path
    import copy, itertools, torch, pandas as pd, numpy as np, gc
    from peft import LoraConfig, get_peft_model, prepare_model_for_kbit_training, PeftModel
    from unsloth import FastLanguageModel
    from datetime import datetime

    # --- Reproducibility ---
    SEED = 42
    torch.manual_seed(SEED)
```



```
[12]
✓ 0s
    # --- Reproducibility ---
    SEED = 42
    torch.manual_seed(SEED)
    np.random.seed(SEED)

    # --- Output directory ---
    OUTPUT_DIR = Path("cv_runs")
    OUTPUT_DIR.mkdir(exist_ok=True)
```

```
[13]
    # === Train model ===
    train_result = trainer.train()

    # === Save trainer state (for resuming training) ===
    trainer.save_state()

    # === Global adapter save (complete run summary) ===
    base_adapter_dir = Path("outputs/lora_adapter")
    base_adapter_dir.mkdir(exist_ok=True)
    trainer.save_model(base_adapter_dir)
    tokenizer.save_pretrained(base_adapter_dir)

    # === Per-fold / per-grid save for organized results ===
    adapter_dir = Path(f"outputs/lora_adapter_grid{grid_id}_fold{fold}")
    adapter_dir.mkdir(parents=True, exist_ok=True)
    trainer.save_model(adapter_dir)
    tokenizer.save_pretrained(adapter_dir)

    # === Summary printout ===
    print(f"Training complete. Steps: {train_result.global_step}")
    print(f"Saved base adapter at: {base_adapter_dir}")
    print(f"Saved fold adapter at: {adapter_dir}")
```



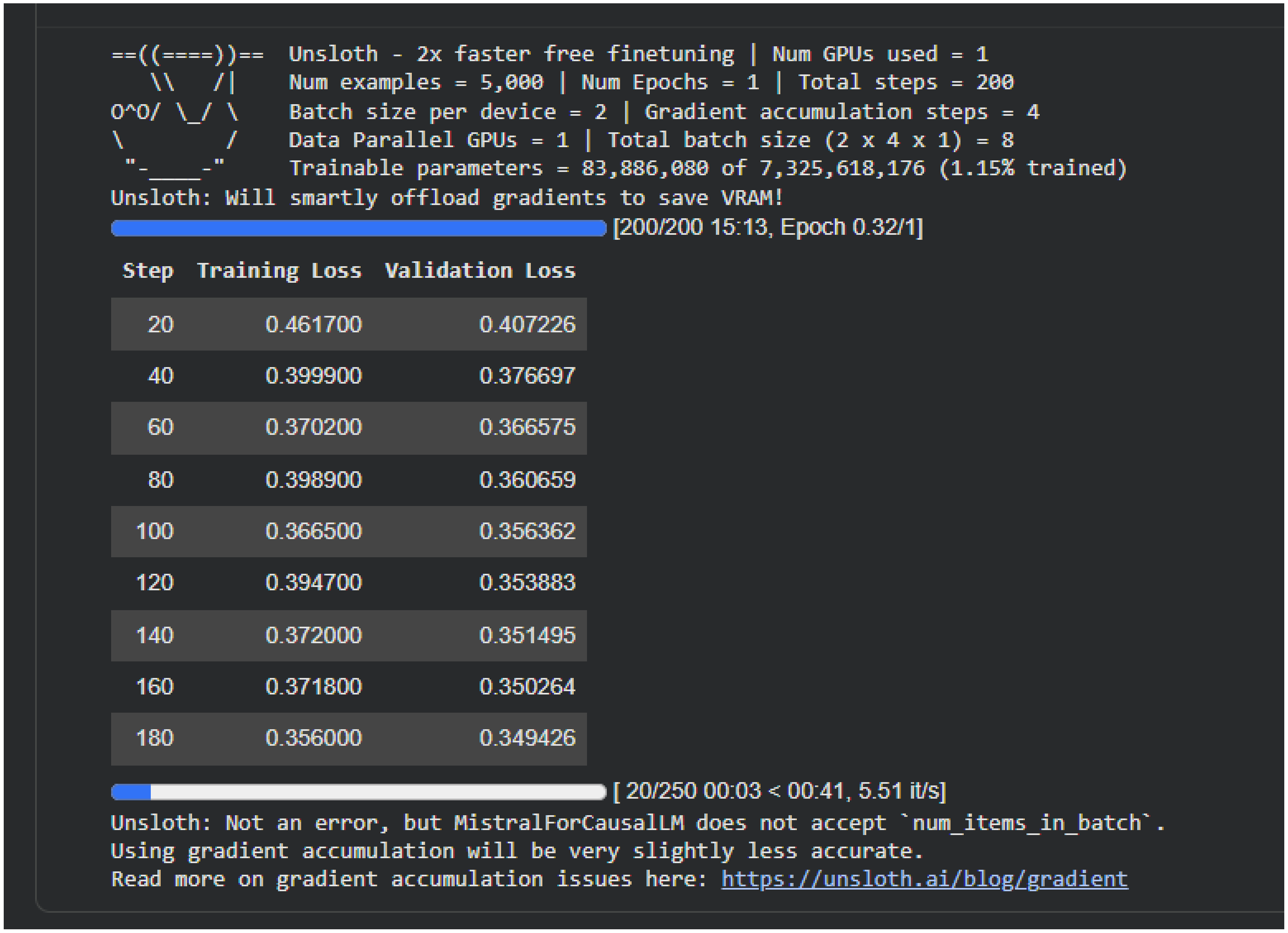
```
==((====))==  Unsloth - 2x faster free finetuning | Num GPUs used = 1
   \\   /|    Num examples = 5,000 | Num Epochs = 1 | Total steps = 200
O^O/ \_/ \    Batch size per device = 2 | Gradient accumulation steps = 4
\        /    Data Parallel GPUs = 1 | Total batch size (2 x 4 x 1) = 8
 "-____-"     Trainable parameters = 83,886,080 of 7,325,618,176 (1.15% trained)
Unsloth: Will smartly offload gradients to save VRAM!
[125/200 09:45 < 05:57, 0.21 it/s, Epoch 0.20/1]
```

Step	Training Loss	Validation Loss
20	0.461700	0.407226
40	0.399900	0.376697
60	0.370200	0.366575
80	0.398900	0.360659
100	0.366500	0.356362
120	0.394700	0.353883

```
Unsloth: Not an error, but MistralForCausalLM does not accept `num_items_in_batch`.
Using gradient accumulation will be very slightly less accurate.
Read more on gradient accumulation issues here: https://unsloth.ai/blog/gradient
```

[ ]

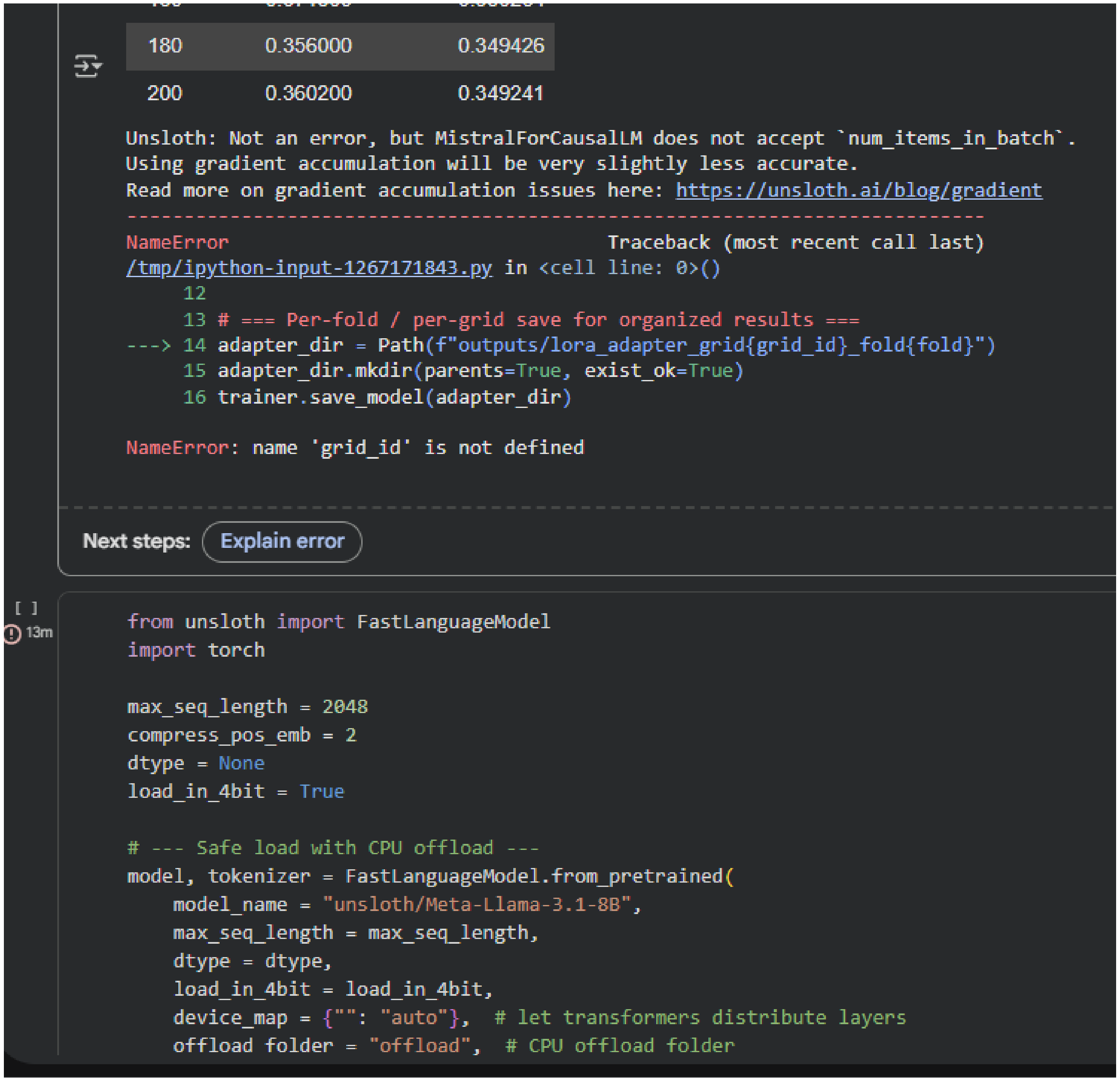
```python
from unsloth import FastLanguageModel
import torch

max_seq_length = 2048
compress_pos_emb = 2
dtype = None
load_in_4bit = True

# --- Safe load with CPU offload ---
model, tokenizer = FastLanguageModel.from_pretrained(
    model_name = "unsloth/Meta-Llama-3.1-8B",
    max_seq_length = max_seq_length,
    dtype = dtype,
```



```
==((====))==  Unsloth - 2x faster free finetuning | Num GPUs used = 1
   \\   /|    Num examples = 5,000 | Num Epochs = 1 | Total steps = 200
O^O/ \_/ \    Batch size per device = 2 | Gradient accumulation steps = 4
\        /    Data Parallel GPUs = 1 | Total batch size (2 x 4 x 1) = 8
 "-____-"     Trainable parameters = 83,886,080 of 7,325,618,176 (1.15% trained)
Unsloth: Will smartly offload gradients to save VRAM!
[200/200 15:13, Epoch 0.32/1]
```

Step	Training Loss	Validation Loss
20	0.461700	0.407226
40	0.399900	0.376697
60	0.370200	0.366575
80	0.398900	0.360659
100	0.366500	0.356362
120	0.394700	0.353883
140	0.372000	0.351495
160	0.371800	0.350264
180	0.356000	0.349426

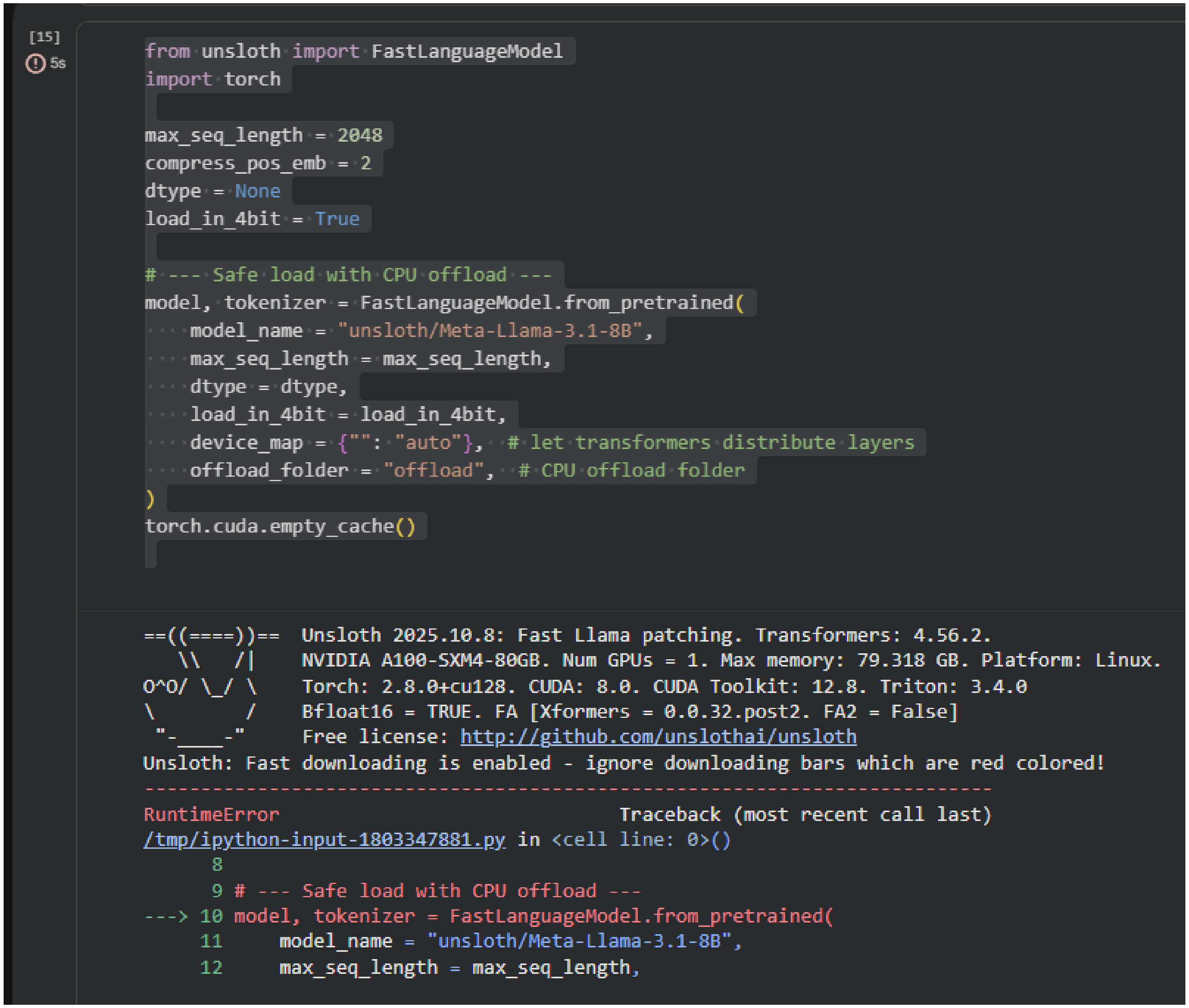
```
[ 20/250 00:03 < 00:41, 5.51 it/s]
Unsloth: Not an error, but MistralForCausalLM does not accept `num_items_in_batch`.
Using gradient accumulation will be very slightly less accurate.
Read more on gradient accumulation issues here: https://unsloth.ai/blog/gradient
```



```
180     0.356000     0.349426
200     0.360200     0.349241
```

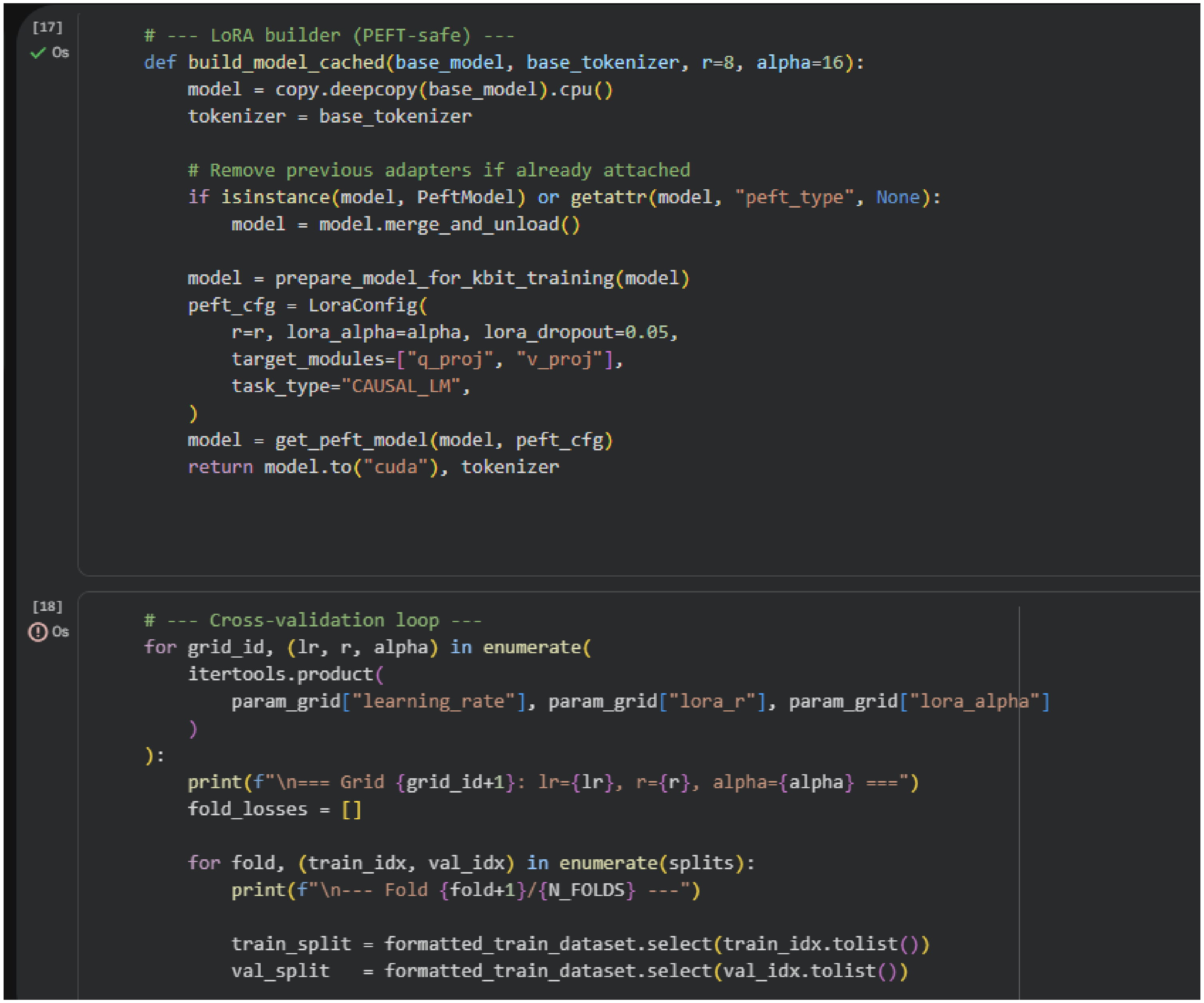
```
Unsloth: Not an error, but MistralForCausalLM does not accept `num_items_in_batch`.
Using gradient accumulation will be very slightly less accurate.
Read more on gradient accumulation issues here: https://unsloth.ai/blog/gradient
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
/tmp/ipython-input-1267171843.py in <cell line: 0>()
     12
     13 # === Per-fold / per-grid save for organized results ===
---> 14 adapter_dir = Path(f"outputs/lora_adapter_grid{grid_id}_fold{fold}")
     15 adapter_dir.mkdir(parents=True, exist_ok=True)
     16 trainer.save_model(adapter_dir)

NameError: name 'grid_id' is not defined
```

```
from unsloth import FastLanguageModel
import torch

max_seq_length = 2048
compress_pos_emb = 2
dtype = None
load_in_4bit = True

# --- Safe load with CPU offload ---
model, tokenizer = FastLanguageModel.from_pretrained(
    model_name = "unsloth/Meta-Llama-3.1-8B",
    max_seq_length = max_seq_length,
    dtype = dtype,
    load_in_4bit = load_in_4bit,
    device_map = {"": "auto"},  # let transformers distribute layers
    offload_folder = "offload",  # CPU offload folder
```


[15]
5s

```python
from unsloth import FastLanguageModel
import torch

max_seq_length = 2048
compress_pos_emb = 2
dtype = None
load_in_4bit = True

# --- Safe load with CPU offload ---
model, tokenizer = FastLanguageModel.from_pretrained(
    model_name = "unsloth/Meta-Llama-3.1-8B",
    max_seq_length = max_seq_length,
    dtype = dtype,
    load_in_4bit = load_in_4bit,
    device_map = {"": "auto"},  # let transformers distribute layers
    offload_folder = "offload",  # CPU offload folder
)
torch.cuda.empty_cache()
```

```
==((====))==  Unsloth 2025.10.8: Fast Llama patching. Transformers: 4.56.2.
   \\   /|    NVIDIA A100-SXM4-80GB. Num GPUs = 1. Max memory: 79.318 GB. Platform: Linux.
O^O/ \_/ \    Torch: 2.8.0+cu128. CUDA: 8.0. CUDA Toolkit: 12.8. Triton: 3.4.0
\        /    Bfloat16 = TRUE. FA [Xformers = 0.0.32.post2. FA2 = False]
 "-____-"     Free license: http://github.com/unslothai/unsloth
Unsloth: Fast downloading is enabled - ignore downloading bars which are red colored!
---------------------------------------------------------------------------
RuntimeError                              Traceback (most recent call last)
/tmp/ipython-input-1803347881.py in <cell line: 0>()
      8
      9 # --- Safe load with CPU offload ---
---> 10 model, tokenizer = FastLanguageModel.from_pretrained(
     11     model_name = "unsloth/Meta-Llama-3.1-8B",
     12     max_seq_length = max_seq_length,
```

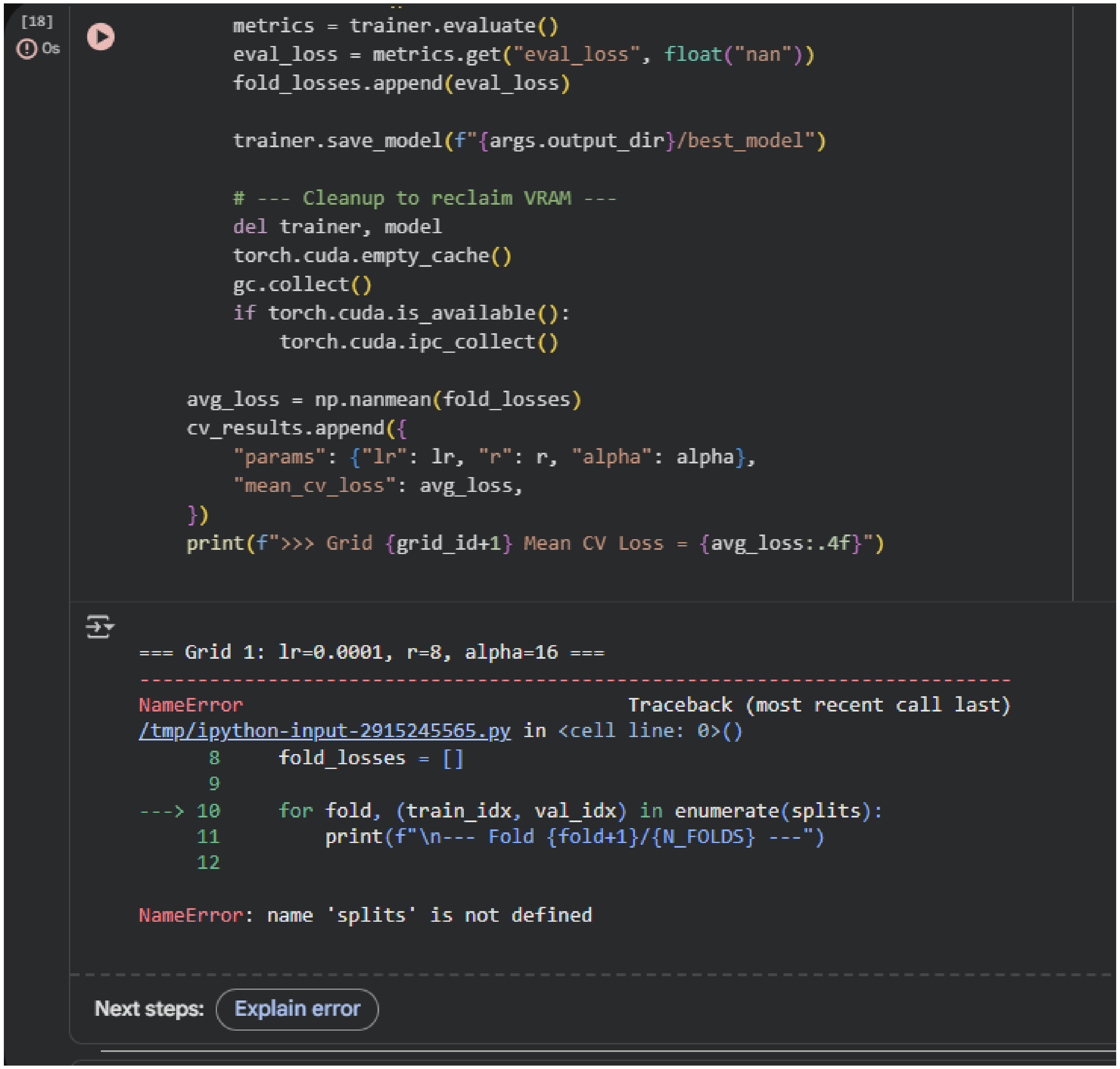


[17] ✓ 0s

```
# --- LoRA builder (PEFT-safe) ---
def build_model_cached(base_model, base_tokenizer, r=8, alpha=16):
    model = copy.deepcopy(base_model).cpu()
    tokenizer = base_tokenizer

    # Remove previous adapters if already attached
    if isinstance(model, PeftModel) or getattr(model, "peft_type", None):
        model = model.merge_and_unload()

    model = prepare_model_for_kbit_training(model)
    peft_cfg = LoraConfig(
        r=r, lora_alpha=alpha, lora_dropout=0.05,
        target_modules=["q_proj", "v_proj"],
        task_type="CAUSAL_LM",
    )
    model = get_peft_model(model, peft_cfg)
    return model.to("cuda"), tokenizer
```

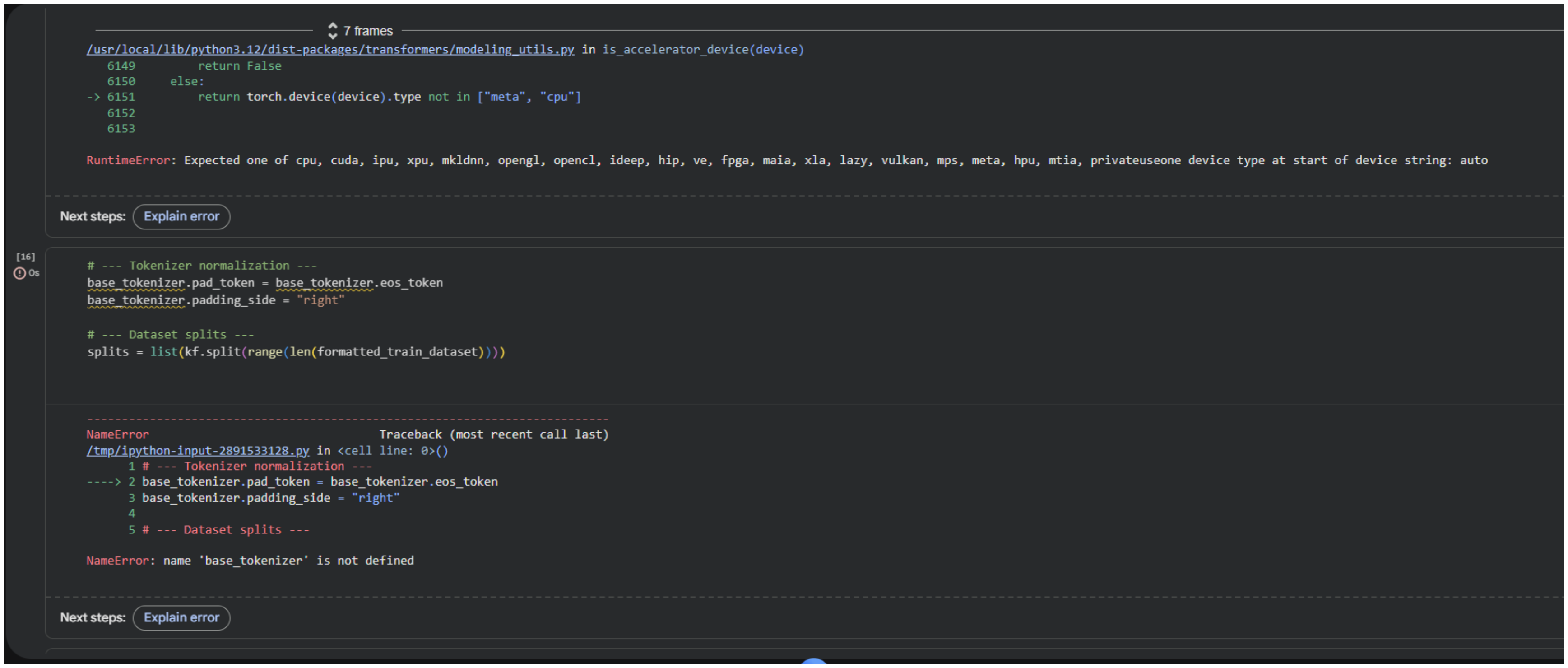
[18] ⓘ 0s

```
# --- Cross-validation loop ---
for grid_id, (lr, r, alpha) in enumerate(
    itertools.product(
        param_grid["learning_rate"], param_grid["lora_r"], param_grid["lora_alpha"]
    )
):
    print(f"\n=== Grid {grid_id+1}: lr={lr}, r={r}, alpha={alpha} ===")
    fold_losses = []

    for fold, (train_idx, val_idx) in enumerate(splits):
        print(f"\n--- Fold {fold+1}/{N_FOLDS} ---")

        train_split = formatted_train_dataset.select(train_idx.tolist())
        val_split   = formatted_train_dataset.select(val_idx.tolist())
```



```
        metrics = trainer.evaluate()
        eval_loss = metrics.get("eval_loss", float("nan"))
        fold_losses.append(eval_loss)

        trainer.save_model(f"{args.output_dir}/best_model")

        # --- Cleanup to reclaim VRAM ---
        del trainer, model
        torch.cuda.empty_cache()
        gc.collect()
        if torch.cuda.is_available():
            torch.cuda.ipc_collect()

    avg_loss = np.nanmean(fold_losses)
    cv_results.append({
        "params": {"lr": lr, "r": r, "alpha": alpha},
        "mean_cv_loss": avg_loss,
    })
    print(f">>> Grid {grid_id+1} Mean CV Loss = {avg_loss:.4f}")
```

```
=== Grid 1: lr=0.0001, r=8, alpha=16 ===
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
/tmp/ipython-input-2915245565.py in <cell line: 0>()
      8     fold_losses = []
      9 
---> 10     for fold, (train_idx, val_idx) in enumerate(splits):
     11         print(f"\n--- Fold {fold+1}/{N_FOLDS} ---")
     12 

NameError: name 'splits' is not defined
```



7 frames

```
/usr/local/lib/python3.12/dist-packages/transformers/modeling_utils.py in is_accelerator_device(device)
   6149        return False
   6150    else:
-> 6151        return torch.device(device).type not in ["meta", "cpu"]
   6152
   6153

RuntimeError: Expected one of cpu, cuda, ipu, xpu, mkldnn, opengl, opencl, ideep, hip, ve, fpga, maia, xla, lazy, vulkan, mps, meta, hpu, mtia, privateuseone device type at start of device string: auto
```

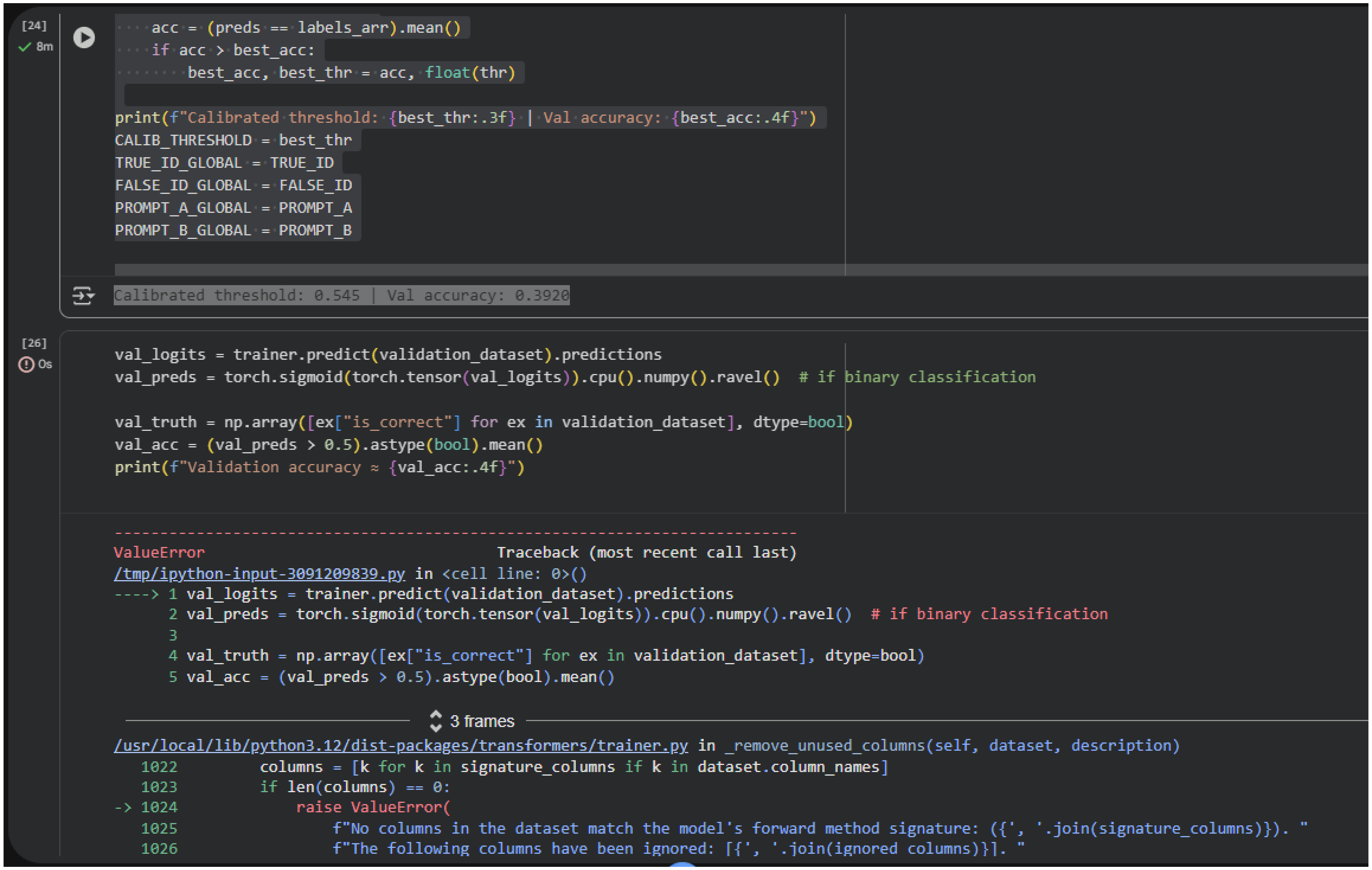
Next steps: Explain error

[16] 0s

```
# --- Tokenizer normalization ---
base_tokenizer.pad_token = base_tokenizer.eos_token
base_tokenizer.padding_side = "right"

# --- Dataset splits ---
splits = list(kf.split(range(len(formatted_train_dataset))))
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
/tmp/ipython-input-2891533128.py in <cell line: 0>()
      1 # --- Tokenizer normalization ---
----> 2 base_tokenizer.pad_token = base_tokenizer.eos_token
      3 base_tokenizer.padding_side = "right"
      4
      5 # --- Dataset splits ---

NameError: name 'base_tokenizer' is not defined
```

Next steps: Explain error

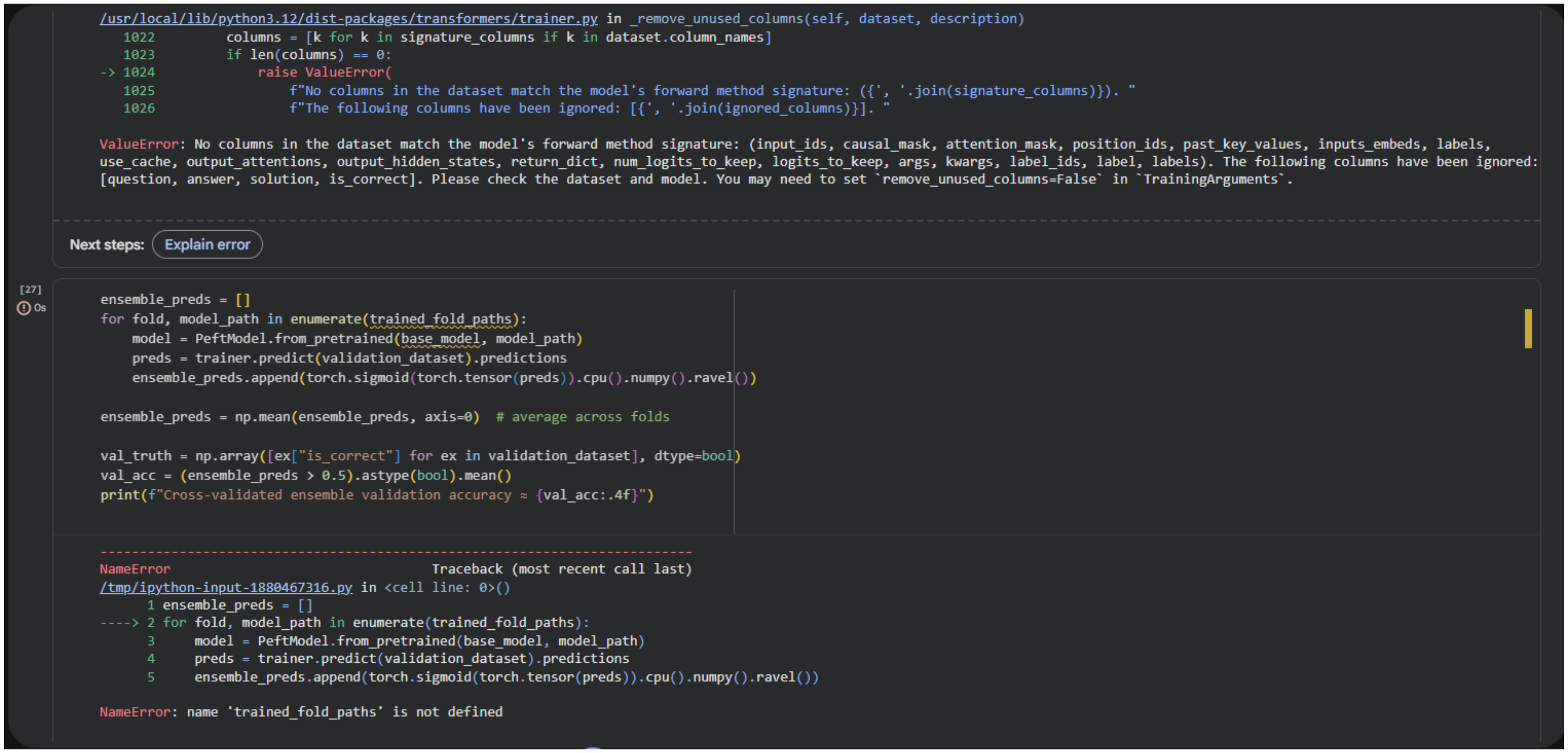


```
[24]
    acc = (preds == labels_arr).mean()
    if acc > best_acc:
        best_acc, best_thr = acc, float(thr)

print(f"Calibrated threshold: {best_thr:.3f} | Val accuracy: {best_acc:.4f}")
CALIB_THRESHOLD = best_thr
TRUE_ID_GLOBAL = TRUE_ID
FALSE_ID_GLOBAL = FALSE_ID
PROMPT_A_GLOBAL = PROMPT_A
PROMPT_B_GLOBAL = PROMPT_B
```

```
Calibrated threshold: 0.545 | Val accuracy: 0.3920
```

```
[26]
val_logits = trainer.predict(validation_dataset).predictions
val_preds = torch.sigmoid(torch.tensor(val_logits)).cpu().numpy().ravel()  # if binary classification

val_truth = np.array([ex["is_correct"] for ex in validation_dataset], dtype=bool)
val_acc = (val_preds > 0.5).astype(bool).mean()
print(f"Validation accuracy ≈ {val_acc:.4f}")
```

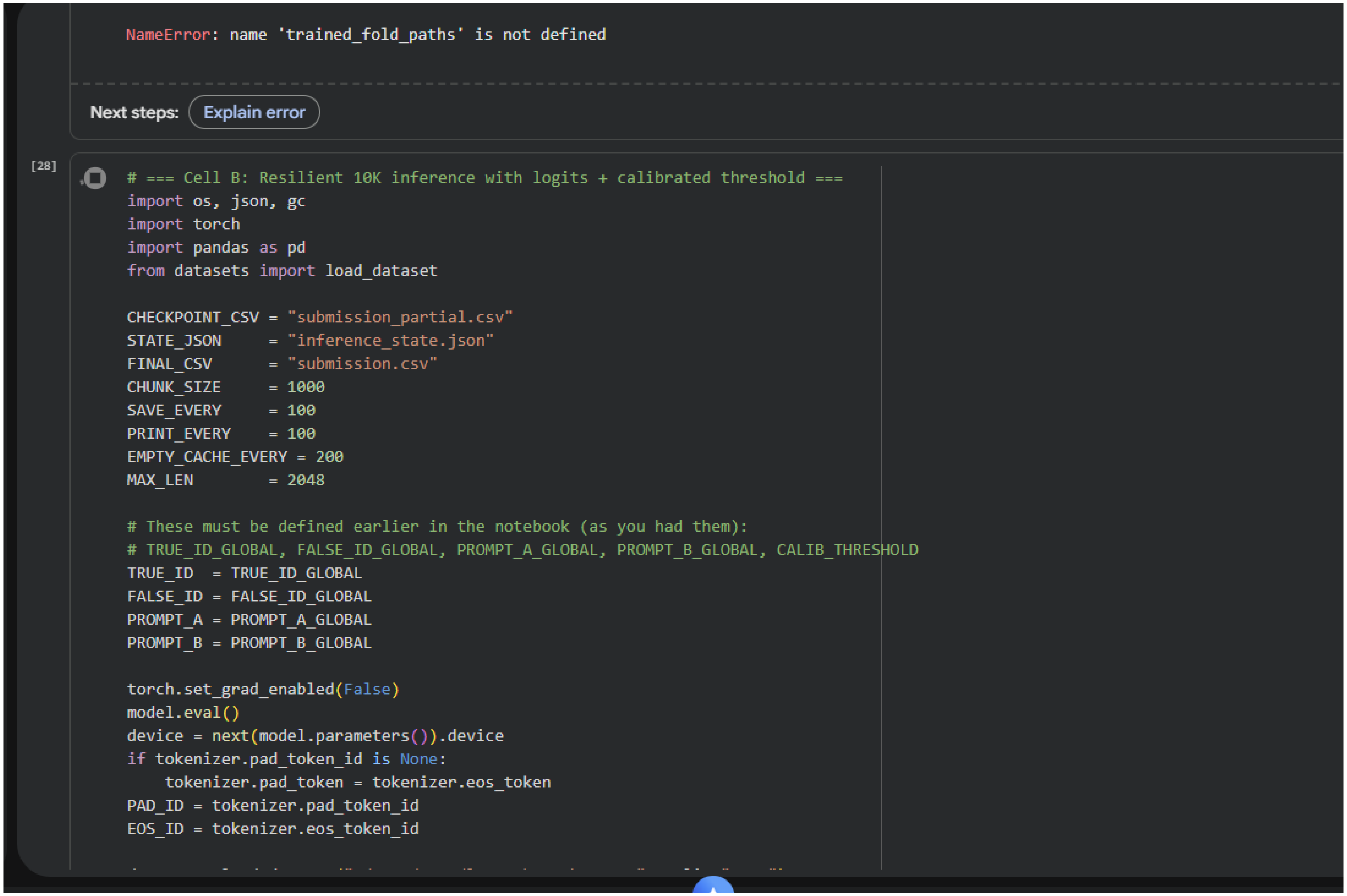
```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/tmp/ipython-input-3091209839.py in <cell line: 0>()
----> 1 val_logits = trainer.predict(validation_dataset).predictions
      2 val_preds = torch.sigmoid(torch.tensor(val_logits)).cpu().numpy().ravel()  # if binary classification
      3 
      4 val_truth = np.array([ex["is_correct"] for ex in validation_dataset], dtype=bool)
      5 val_acc = (val_preds > 0.5).astype(bool).mean()

3 frames
/usr/local/lib/python3.12/dist-packages/transformers/trainer.py in _remove_unused_columns(self, dataset, description)
   1022         columns = [k for k in signature_columns if k in dataset.column_names]
   1023         if len(columns) == 0:
-> 1024             raise ValueError(
   1025                 f"No columns in the dataset match the model's forward method signature: ({', '.join(signature_columns)}). "
   1026                 f"The following columns have been ignored: [{', '.join(ignored_columns)}]. "
```



```
/usr/local/lib/python3.12/dist-packages/transformers/trainer.py in _remove_unused_columns(self, dataset, description)
   1022         columns = [k for k in signature_columns if k in dataset.column_names]
   1023         if len(columns) == 0:
-> 1024             raise ValueError(
   1025                 f"No columns in the dataset match the model's forward method signature: ({', '.join(signature_columns)}). "
   1026                 f"The following columns have been ignored: [{', '.join(ignored_columns)}]. "

ValueError: No columns in the dataset match the model's forward method signature: (input_ids, causal_mask, attention_mask, position_ids, past_key_values, inputs_embeds, labels,
use_cache, output_attentions, output_hidden_states, return_dict, num_logits_to_keep, logits_to_keep, args, kwargs, label_ids, label, labels). The following columns have been ignored:
[question, answer, solution, is_correct]. Please check the dataset and model. You may need to set `remove_unused_columns=False` in `TrainingArguments`.
```

```python
ensemble_preds = []
for fold, model_path in enumerate(trained_fold_paths):
    model = PeftModel.from_pretrained(base_model, model_path)
    preds = trainer.predict(validation_dataset).predictions
    ensemble_preds.append(torch.sigmoid(torch.tensor(preds)).cpu().numpy().ravel())

ensemble_preds = np.mean(ensemble_preds, axis=0)  # average across folds

val_truth = np.array([ex["is_correct"] for ex in validation_dataset], dtype=bool)
val_acc = (ensemble_preds > 0.5).astype(bool).mean()
print(f"Cross-validated ensemble validation accuracy ≈ {val_acc:.4f}")
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
/tmp/ipython-input-1880467316.py in <cell line: 0>()
      1 ensemble_preds = []
----> 2 for fold, model_path in enumerate(trained_fold_paths):
      3     model = PeftModel.from_pretrained(base_model, model_path)
      4     preds = trainer.predict(validation_dataset).predictions
      5     ensemble_preds.append(torch.sigmoid(torch.tensor(preds)).cpu().numpy().ravel())

NameError: name 'trained_fold_paths' is not defined
```

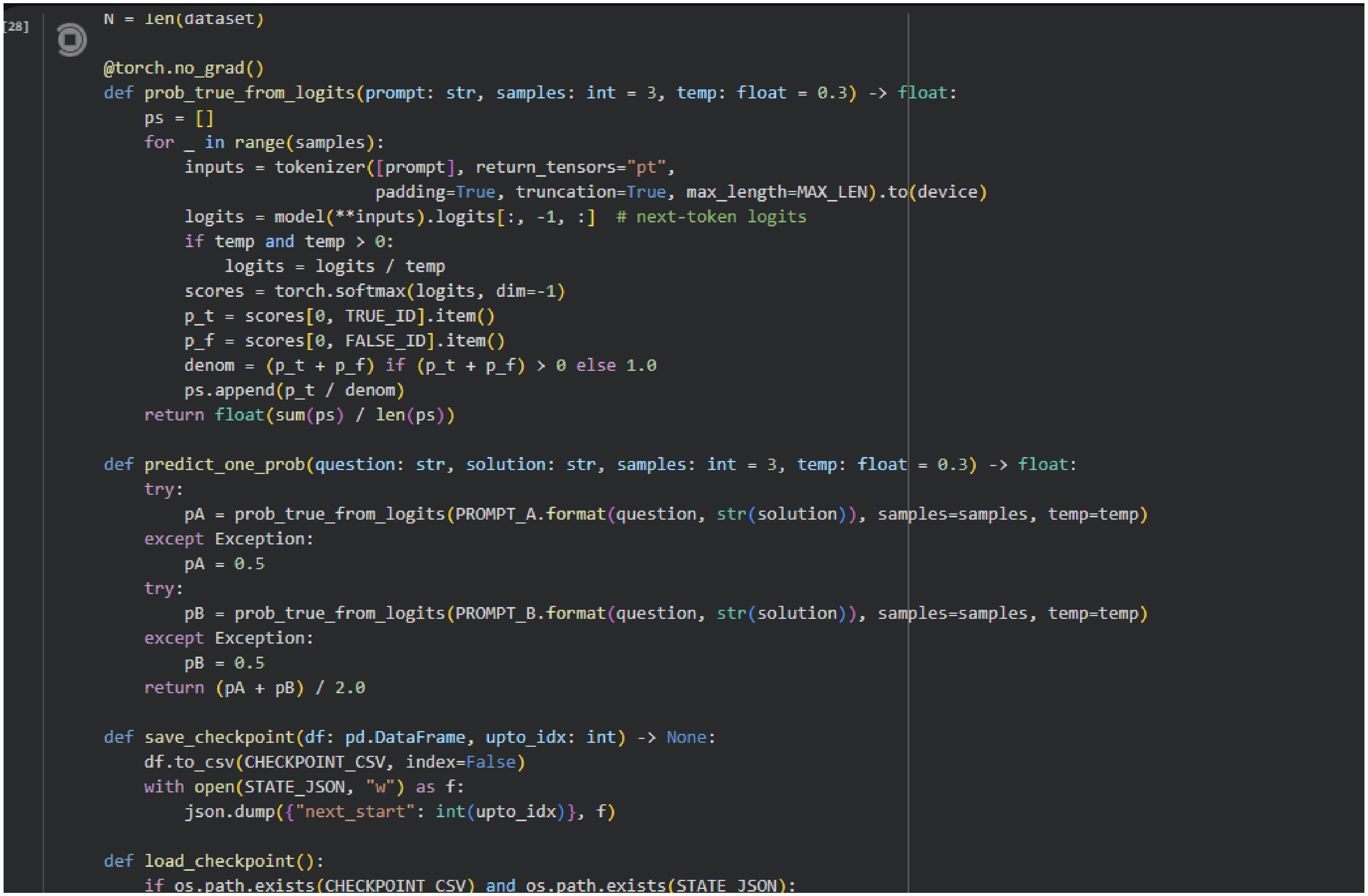


```
NameError: name 'trained_fold_paths' is not defined
```

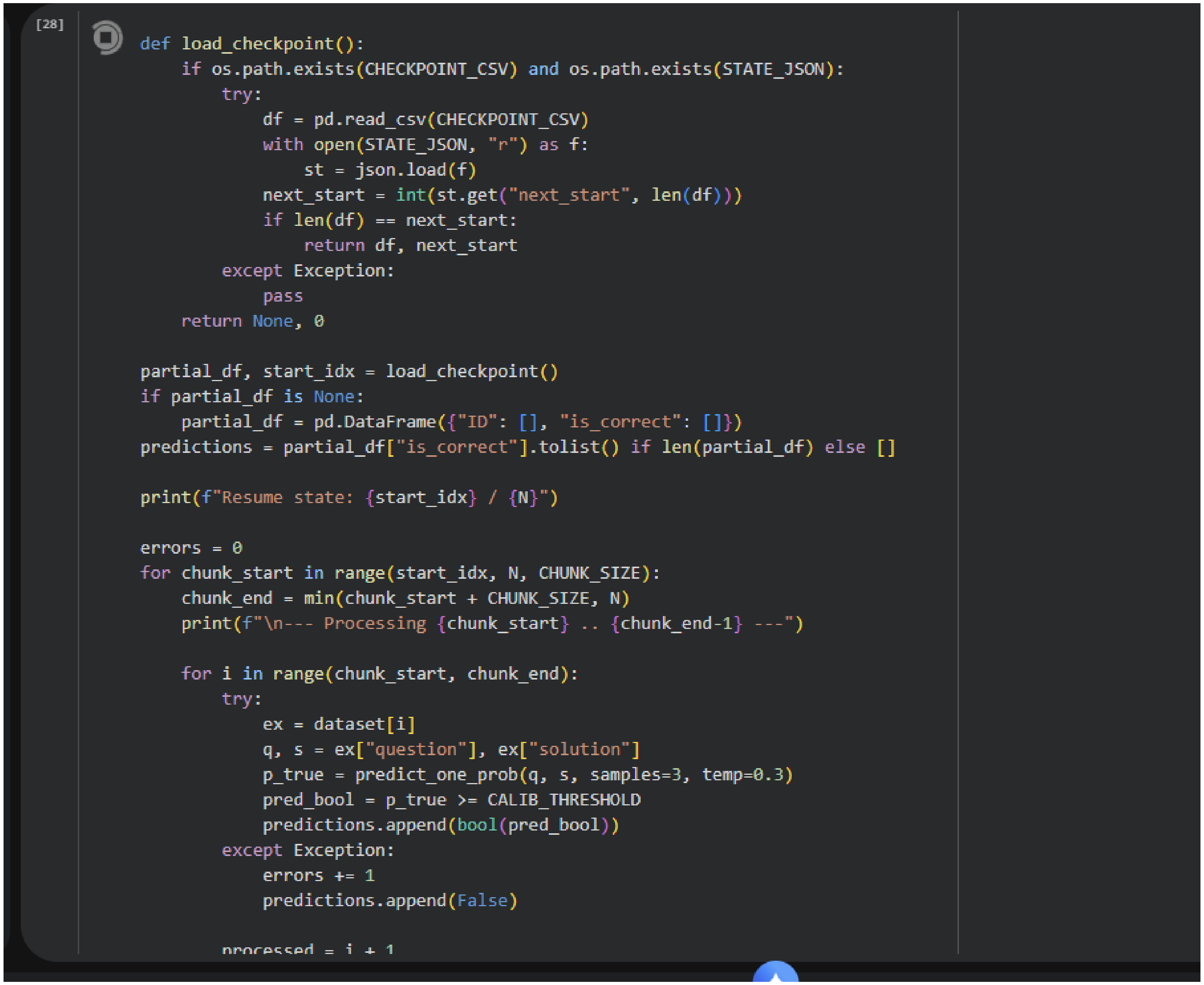
Next steps: Explain error

[28]

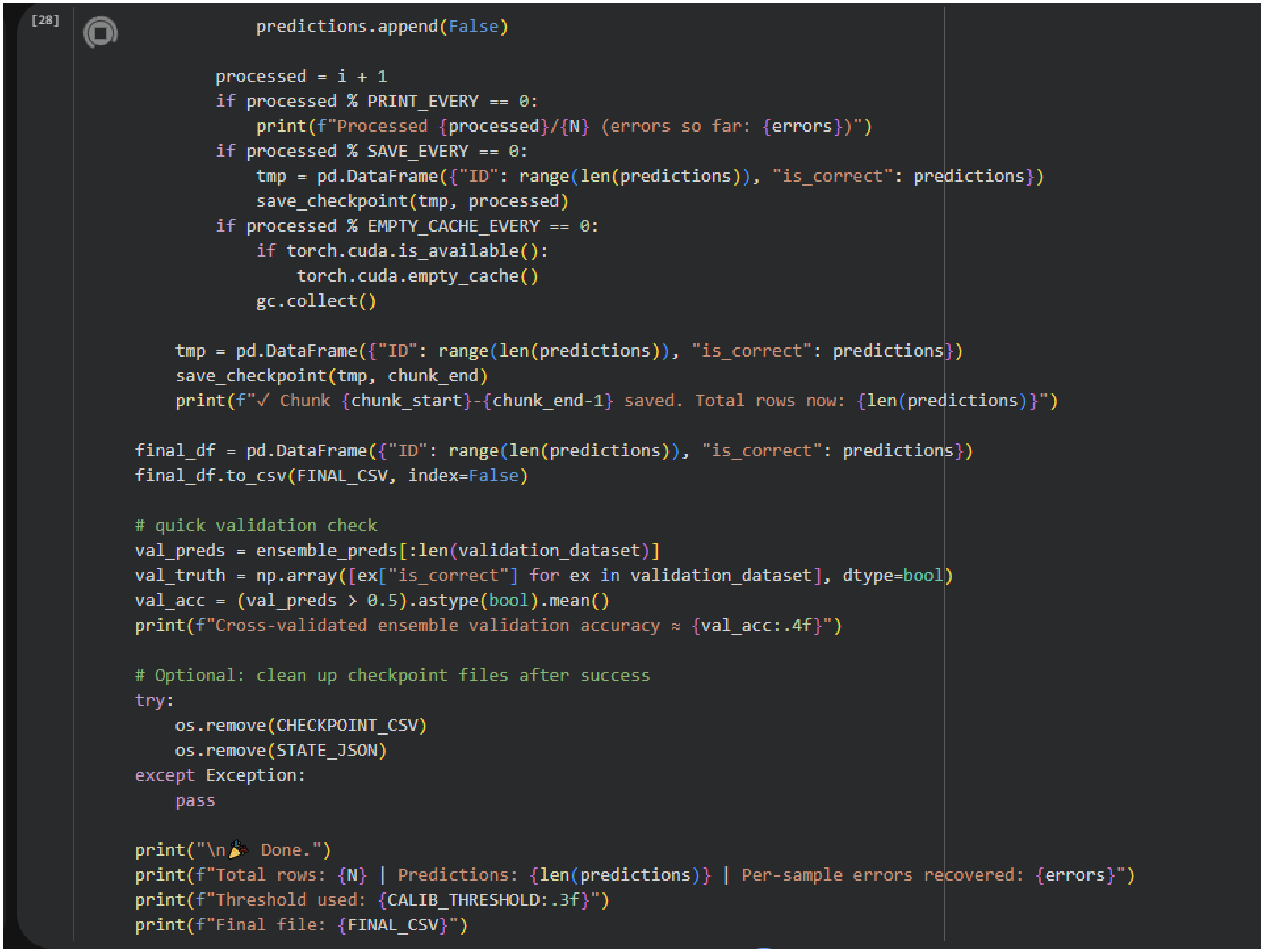
```
# === Cell B: Resilient 10K inference with logits + calibrated threshold ===
import os, json, gc
import torch
import pandas as pd
from datasets import load_dataset

CHECKPOINT_CSV = "submission_partial.csv"
STATE_JSON     = "inference_state.json"
FINAL_CSV      = "submission.csv"
CHUNK_SIZE     = 1000
SAVE_EVERY     = 100
PRINT_EVERY    = 100
EMPTY_CACHE_EVERY = 200
MAX_LEN        = 2048

# These must be defined earlier in the notebook (as you had them):
# TRUE_ID_GLOBAL, FALSE_ID_GLOBAL, PROMPT_A_GLOBAL, PROMPT_B_GLOBAL, CALIB_THRESHOLD
TRUE_ID  = TRUE_ID_GLOBAL
FALSE_ID = FALSE_ID_GLOBAL
PROMPT_A = PROMPT_A_GLOBAL
PROMPT_B = PROMPT_B_GLOBAL

torch.set_grad_enabled(False)
model.eval()
device = next(model.parameters()).device
if tokenizer.pad_token_id is None:
    tokenizer.pad_token = tokenizer.eos_token
PAD_ID = tokenizer.pad_token_id
EOS_ID = tokenizer.eos_token_id
```



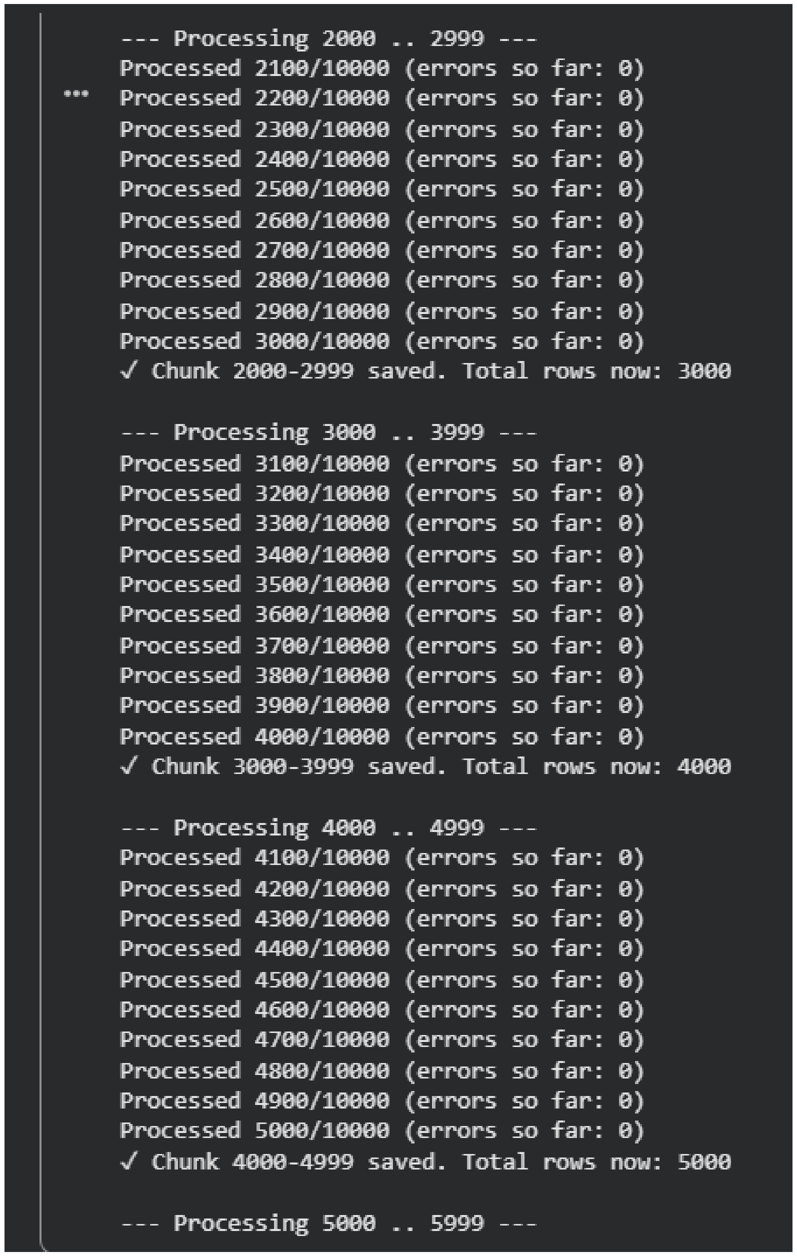
```
[28]
    N = len(dataset)

    @torch.no_grad()
    def prob_true_from_logits(prompt: str, samples: int = 3, temp: float = 0.3) -> float:
        ps = []
        for _ in range(samples):
            inputs = tokenizer([prompt], return_tensors="pt",
                               padding=True, truncation=True, max_length=MAX_LEN).to(device)
            logits = model(**inputs).logits[:, -1, :]  # next-token logits
            if temp and temp > 0:
                logits = logits / temp
            scores = torch.softmax(logits, dim=-1)
            p_t = scores[0, TRUE_ID].item()
            p_f = scores[0, FALSE_ID].item()
            denom = (p_t + p_f) if (p_t + p_f) > 0 else 1.0
            ps.append(p_t / denom)
        return float(sum(ps) / len(ps))

    def predict_one_prob(question: str, solution: str, samples: int = 3, temp: float = 0.3) -> float:
        try:
            pA = prob_true_from_logits(PROMPT_A.format(question, str(solution)), samples=samples, temp=temp)
        except Exception:
            pA = 0.5
        try:
            pB = prob_true_from_logits(PROMPT_B.format(question, str(solution)), samples=samples, temp=temp)
        except Exception:
            pB = 0.5
        return (pA + pB) / 2.0

    def save_checkpoint(df: pd.DataFrame, upto_idx: int) -> None:
        df.to_csv(CHECKPOINT_CSV, index=False)
        with open(STATE_JSON, "w") as f:
            json.dump({"next_start": int(upto_idx)}, f)

    def load_checkpoint():
        if os.path.exists(CHECKPOINT_CSV) and os.path.exists(STATE_JSON):
```


[28]

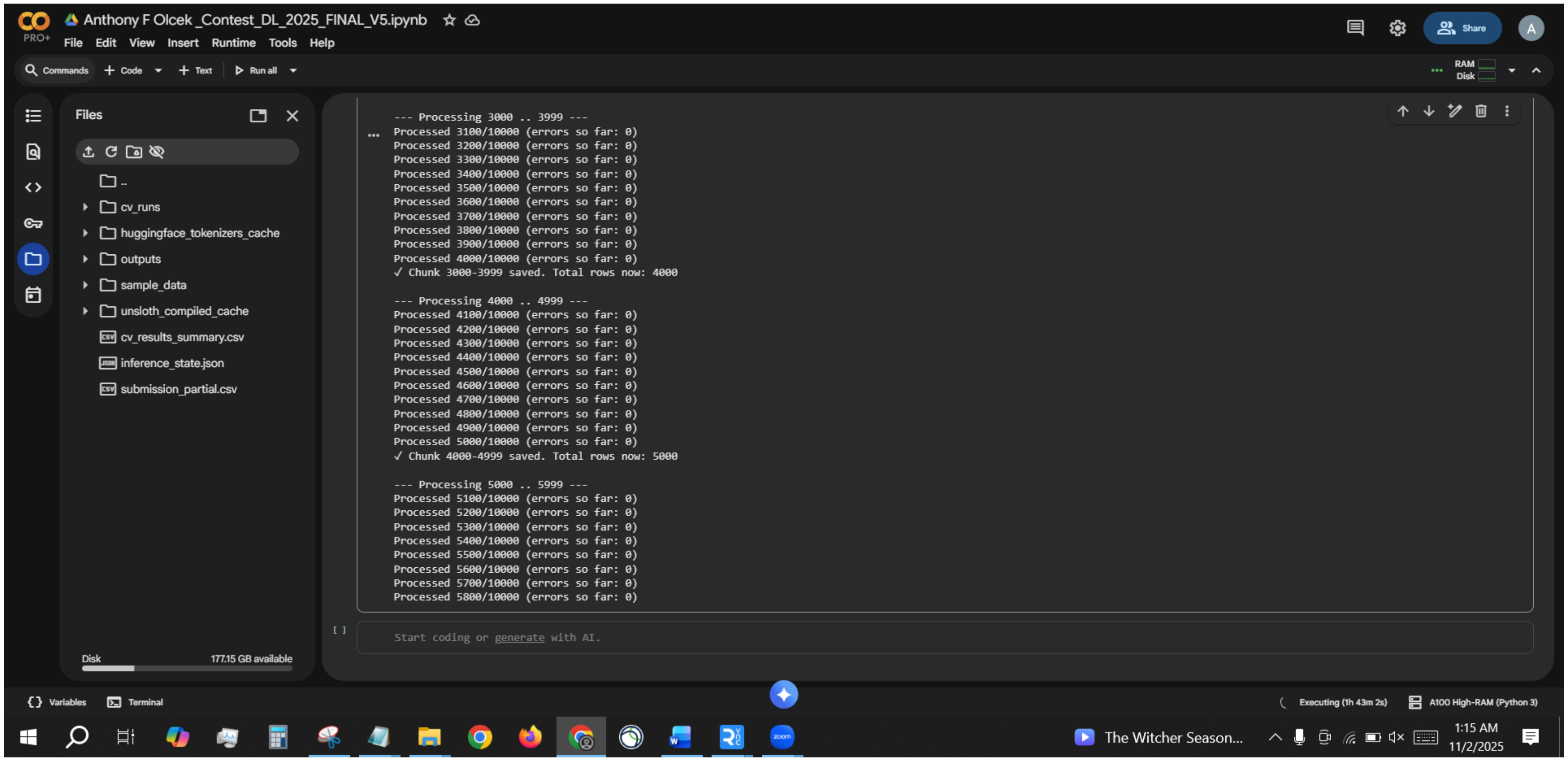
```
def load_checkpoint():
    if os.path.exists(CHECKPOINT_CSV) and os.path.exists(STATE_JSON):
        try:
            df = pd.read_csv(CHECKPOINT_CSV)
            with open(STATE_JSON, "r") as f:
                st = json.load(f)
            next_start = int(st.get("next_start", len(df)))
            if len(df) == next_start:
                return df, next_start
        except Exception:
            pass
    return None, 0

partial_df, start_idx = load_checkpoint()
if partial_df is None:
    partial_df = pd.DataFrame({"ID": [], "is_correct": []})
predictions = partial_df["is_correct"].tolist() if len(partial_df) else []

print(f"Resume state: {start_idx} / {N}")

errors = 0
for chunk_start in range(start_idx, N, CHUNK_SIZE):
    chunk_end = min(chunk_start + CHUNK_SIZE, N)
    print(f"\n--- Processing {chunk_start} .. {chunk_end-1} ---")

    for i in range(chunk_start, chunk_end):
        try:
            ex = dataset[i]
            q, s = ex["question"], ex["solution"]
            p_true = predict_one_prob(q, s, samples=3, temp=0.3)
            pred_bool = p_true >= CALIB_THRESHOLD
            predictions.append(bool(pred_bool))
        except Exception:
            errors += 1
            predictions.append(False)

        processed = i + 1
```



```
[28]
                predictions.append(False)

            processed = i + 1
            if processed % PRINT_EVERY == 0:
                print(f"Processed {processed}/{N} (errors so far: {errors})")
            if processed % SAVE_EVERY == 0:
                tmp = pd.DataFrame({"ID": range(len(predictions)), "is_correct": predictions})
                save_checkpoint(tmp, processed)
            if processed % EMPTY_CACHE_EVERY == 0:
                if torch.cuda.is_available():
                    torch.cuda.empty_cache()
                gc.collect()

        tmp = pd.DataFrame({"ID": range(len(predictions)), "is_correct": predictions})
        save_checkpoint(tmp, chunk_end)
        print(f"✓ Chunk {chunk_start}-{chunk_end-1} saved. Total rows now: {len(predictions)}")

    final_df = pd.DataFrame({"ID": range(len(predictions)), "is_correct": predictions})
    final_df.to_csv(FINAL_CSV, index=False)

    # quick validation check
    val_preds = ensemble_preds[:len(validation_dataset)]
    val_truth = np.array([ex["is_correct"] for ex in validation_dataset], dtype=bool)
    val_acc = (val_preds > 0.5).astype(bool).mean()
    print(f"Cross-validated ensemble validation accuracy ≈ {val_acc:.4f}")

    # Optional: clean up checkpoint files after success
    try:
        os.remove(CHECKPOINT_CSV)
        os.remove(STATE_JSON)
    except Exception:
        pass

    print("\n🎉 Done.")
    print(f"Total rows: {N} | Predictions: {len(predictions)} | Per-sample errors recovered: {errors}")
    print(f"Threshold used: {CALIB_THRESHOLD:.3f}")
    print(f"Final file: {FINAL_CSV}")
```

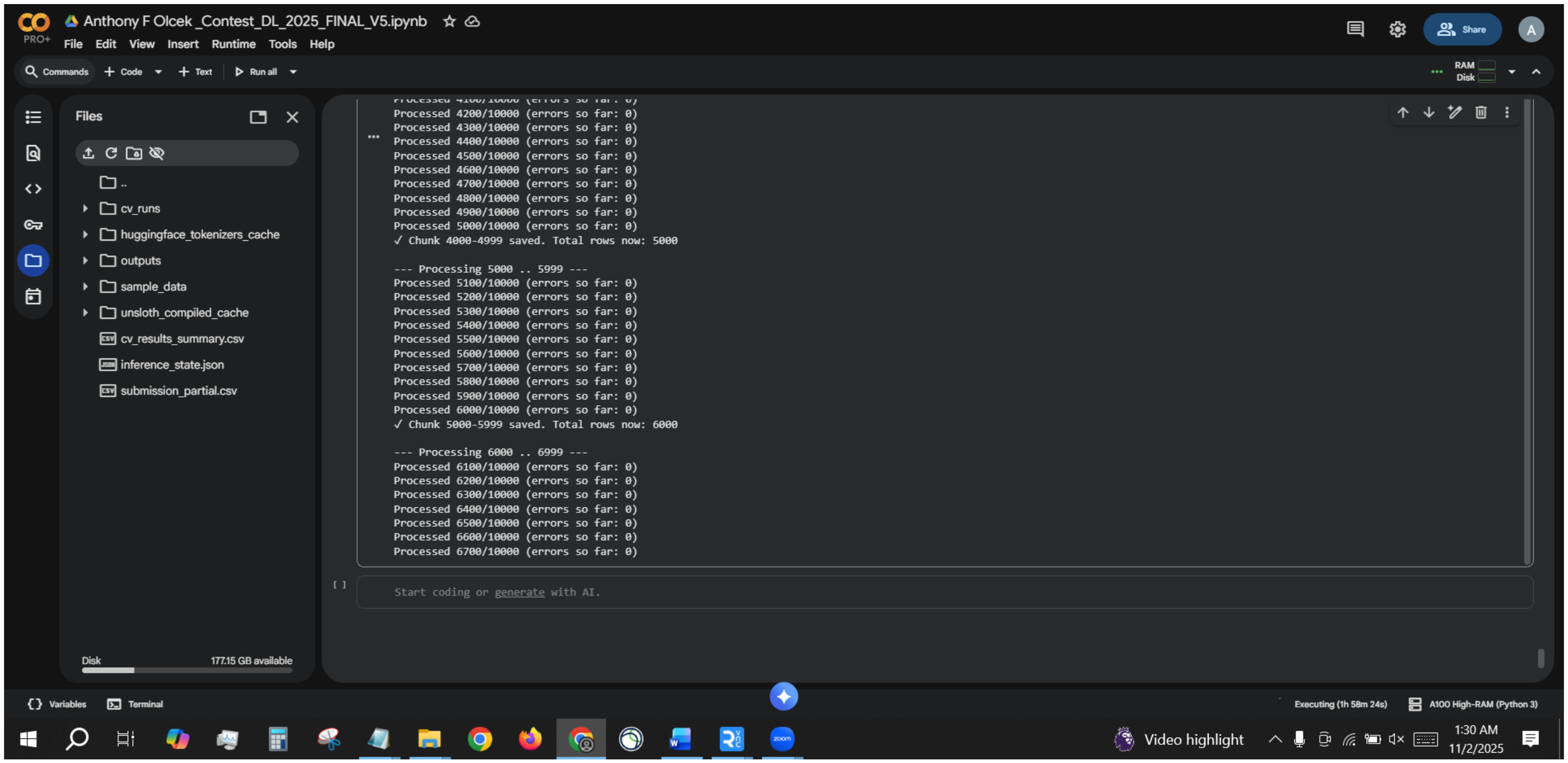


```
--- Processing 2000 .. 2999 ---
Processed 2100/10000 (errors so far: 0)
Processed 2200/10000 (errors so far: 0)
Processed 2300/10000 (errors so far: 0)
Processed 2400/10000 (errors so far: 0)
Processed 2500/10000 (errors so far: 0)
Processed 2600/10000 (errors so far: 0)
Processed 2700/10000 (errors so far: 0)
Processed 2800/10000 (errors so far: 0)
Processed 2900/10000 (errors so far: 0)
Processed 3000/10000 (errors so far: 0)
✓ Chunk 2000-2999 saved. Total rows now: 3000

--- Processing 3000 .. 3999 ---
Processed 3100/10000 (errors so far: 0)
Processed 3200/10000 (errors so far: 0)
Processed 3300/10000 (errors so far: 0)
Processed 3400/10000 (errors so far: 0)
Processed 3500/10000 (errors so far: 0)
Processed 3600/10000 (errors so far: 0)
Processed 3700/10000 (errors so far: 0)
Processed 3800/10000 (errors so far: 0)
Processed 3900/10000 (errors so far: 0)
Processed 4000/10000 (errors so far: 0)
✓ Chunk 3000-3999 saved. Total rows now: 4000

--- Processing 4000 .. 4999 ---
Processed 4100/10000 (errors so far: 0)
Processed 4200/10000 (errors so far: 0)
Processed 4300/10000 (errors so far: 0)
Processed 4400/10000 (errors so far: 0)
Processed 4500/10000 (errors so far: 0)
Processed 4600/10000 (errors so far: 0)
Processed 4700/10000 (errors so far: 0)
Processed 4800/10000 (errors so far: 0)
Processed 4900/10000 (errors so far: 0)
Processed 5000/10000 (errors so far: 0)
✓ Chunk 4000-4999 saved. Total rows now: 5000

--- Processing 5000 .. 5999 ---
```



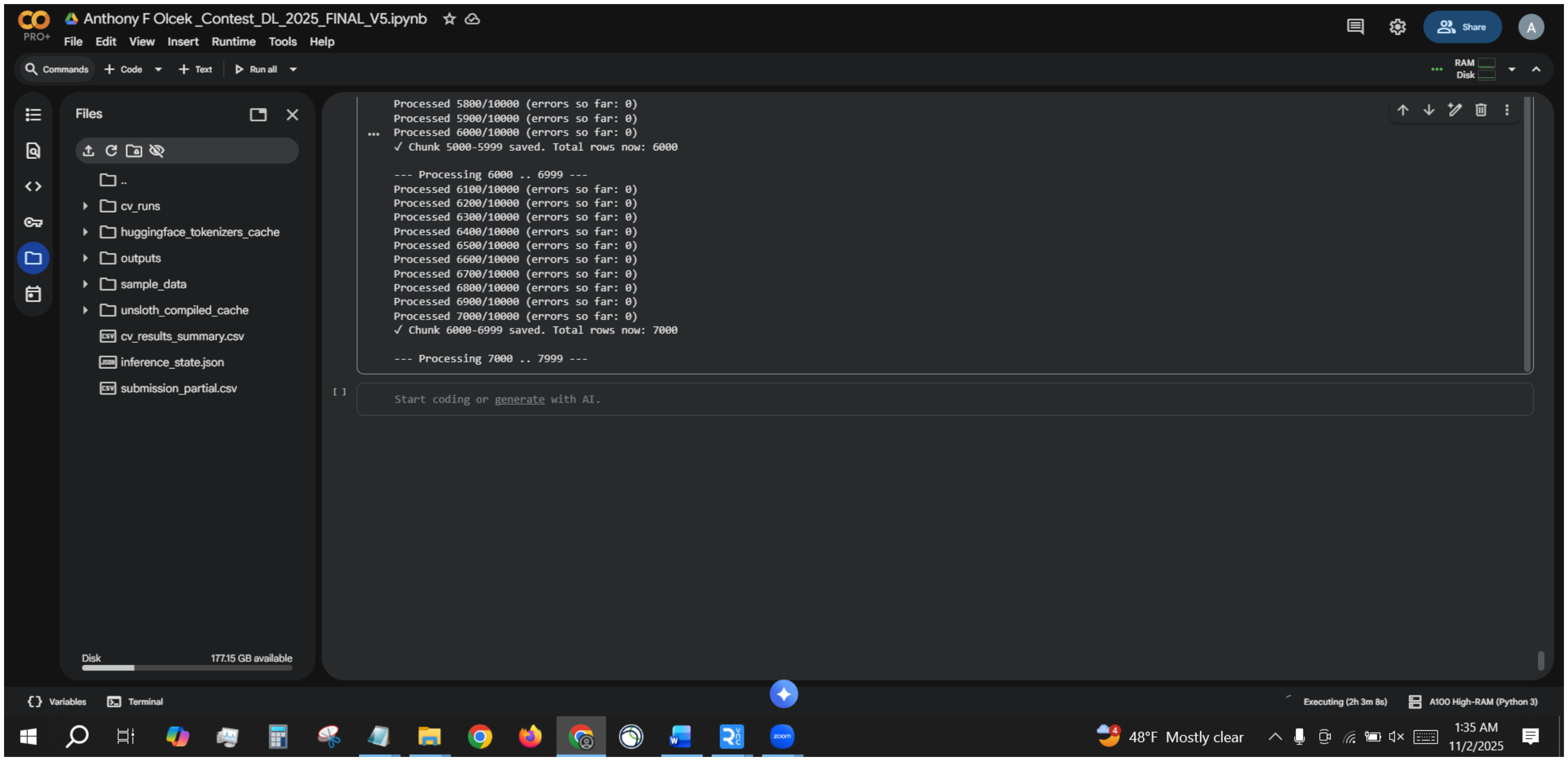
```
--- Processing 3000 .. 3999 ---
Processed 3100/10000 (errors so far: 0)
Processed 3200/10000 (errors so far: 0)
Processed 3300/10000 (errors so far: 0)
Processed 3400/10000 (errors so far: 0)
Processed 3500/10000 (errors so far: 0)
Processed 3600/10000 (errors so far: 0)
Processed 3700/10000 (errors so far: 0)
Processed 3800/10000 (errors so far: 0)
Processed 3900/10000 (errors so far: 0)
Processed 4000/10000 (errors so far: 0)
✓ Chunk 3000-3999 saved. Total rows now: 4000

--- Processing 4000 .. 4999 ---
Processed 4100/10000 (errors so far: 0)
Processed 4200/10000 (errors so far: 0)
Processed 4300/10000 (errors so far: 0)
Processed 4400/10000 (errors so far: 0)
Processed 4500/10000 (errors so far: 0)
Processed 4600/10000 (errors so far: 0)
Processed 4700/10000 (errors so far: 0)
Processed 4800/10000 (errors so far: 0)
Processed 4900/10000 (errors so far: 0)
Processed 5000/10000 (errors so far: 0)
✓ Chunk 4000-4999 saved. Total rows now: 5000

--- Processing 5000 .. 5999 ---
Processed 5100/10000 (errors so far: 0)
Processed 5200/10000 (errors so far: 0)
Processed 5300/10000 (errors so far: 0)
Processed 5400/10000 (errors so far: 0)
Processed 5500/10000 (errors so far: 0)
Processed 5600/10000 (errors so far: 0)
Processed 5700/10000 (errors so far: 0)
Processed 5800/10000 (errors so far: 0)
```

```
Start coding or generate with AI.
```



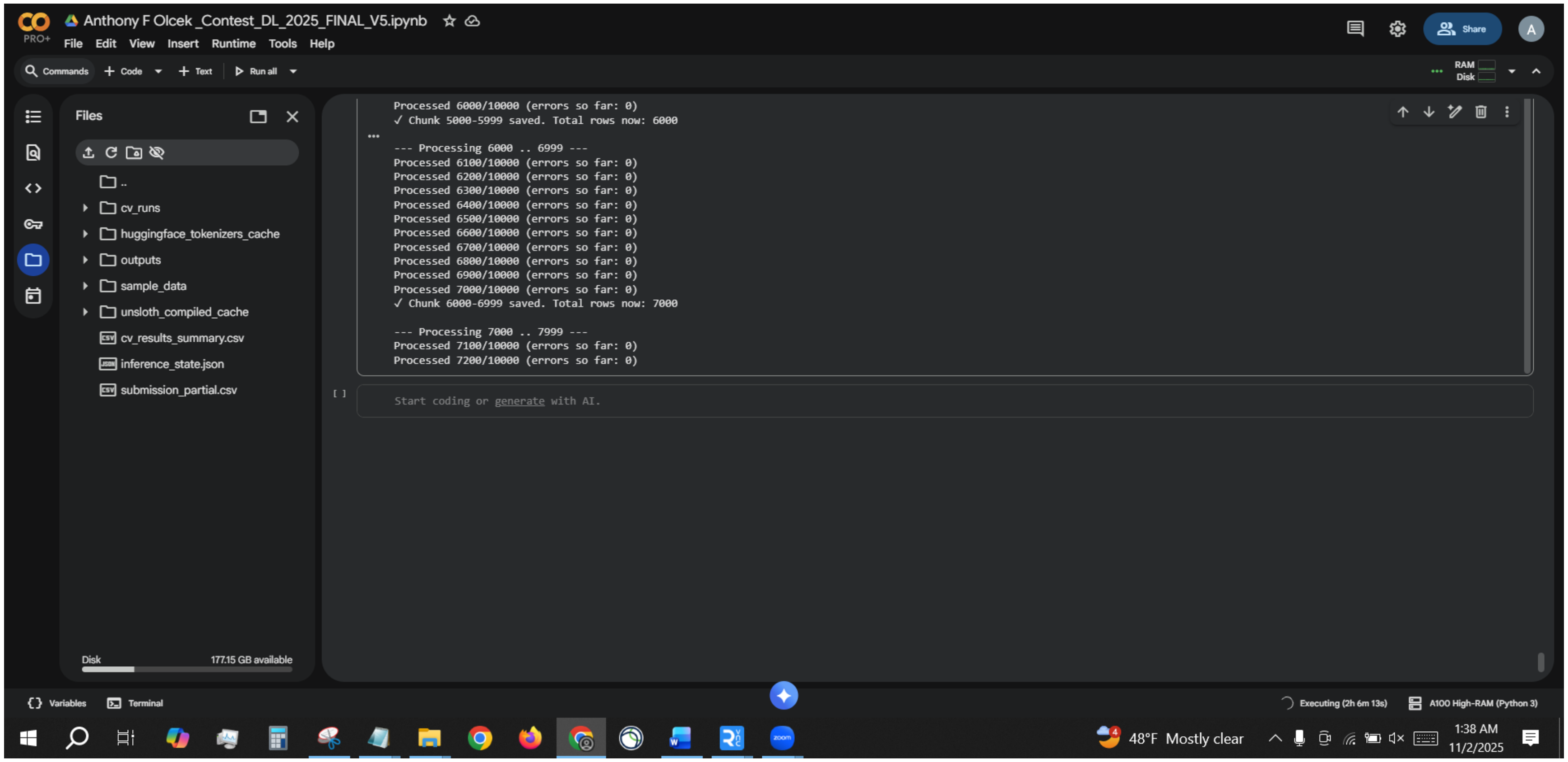
```
Processed 4200/10000 (errors so far: 0)
Processed 4300/10000 (errors so far: 0)
Processed 4400/10000 (errors so far: 0)
Processed 4500/10000 (errors so far: 0)
Processed 4600/10000 (errors so far: 0)
Processed 4700/10000 (errors so far: 0)
Processed 4800/10000 (errors so far: 0)
Processed 4900/10000 (errors so far: 0)
Processed 5000/10000 (errors so far: 0)
✓ Chunk 4000-4999 saved. Total rows now: 5000

--- Processing 5000 .. 5999 ---
Processed 5100/10000 (errors so far: 0)
Processed 5200/10000 (errors so far: 0)
Processed 5300/10000 (errors so far: 0)
Processed 5400/10000 (errors so far: 0)
Processed 5500/10000 (errors so far: 0)
Processed 5600/10000 (errors so far: 0)
Processed 5700/10000 (errors so far: 0)
Processed 5800/10000 (errors so far: 0)
Processed 5900/10000 (errors so far: 0)
Processed 6000/10000 (errors so far: 0)
✓ Chunk 5000-5999 saved. Total rows now: 6000

--- Processing 6000 .. 6999 ---
Processed 6100/10000 (errors so far: 0)
Processed 6200/10000 (errors so far: 0)
Processed 6300/10000 (errors so far: 0)
Processed 6400/10000 (errors so far: 0)
Processed 6500/10000 (errors so far: 0)
Processed 6600/10000 (errors so far: 0)
Processed 6700/10000 (errors so far: 0)
```

```
Start coding or generate with AI.
```



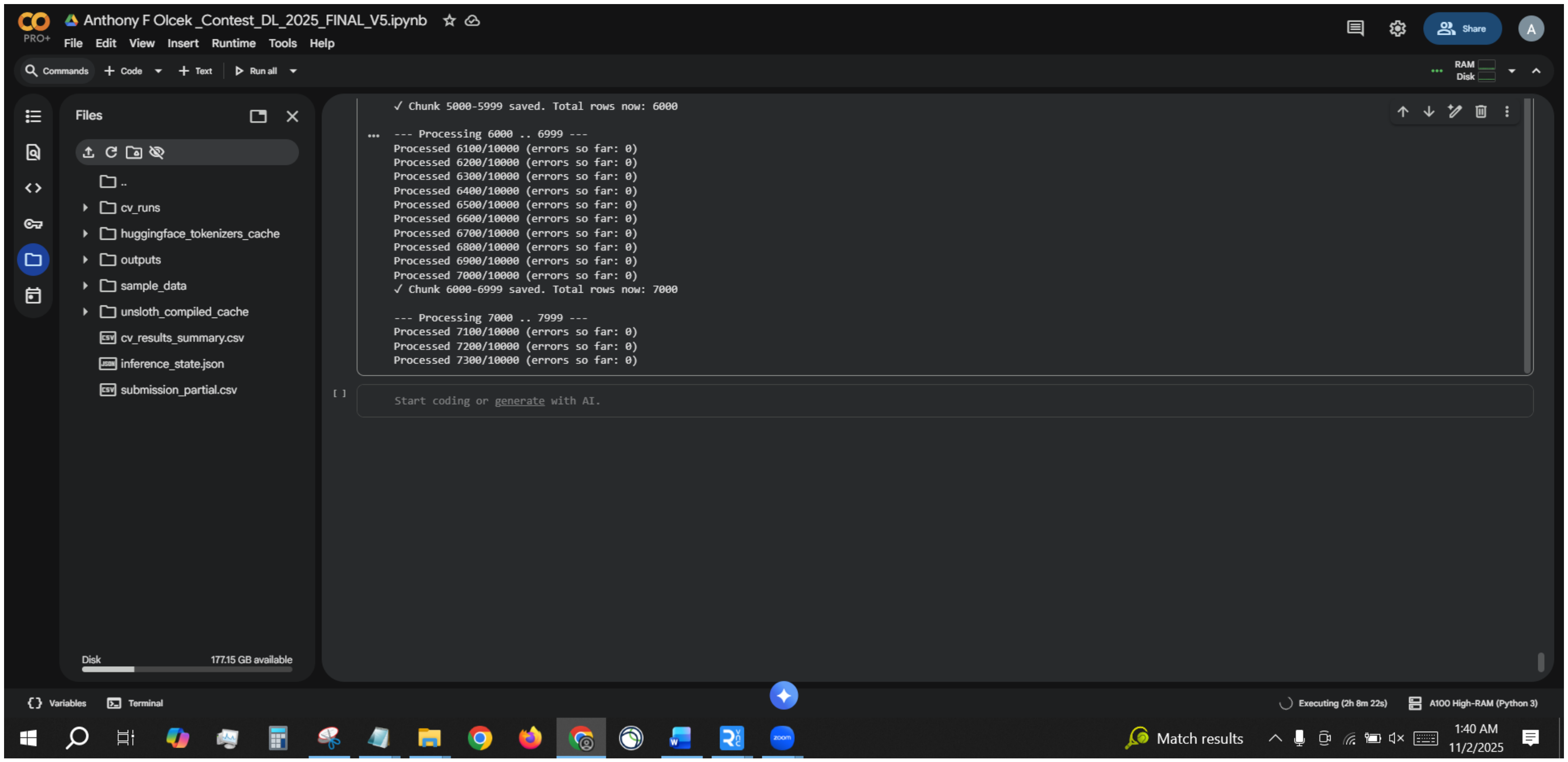
```
Processed 5800/10000 (errors so far: 0)
Processed 5900/10000 (errors so far: 0)
Processed 6000/10000 (errors so far: 0)
✓ Chunk 5000-5999 saved. Total rows now: 6000

--- Processing 6000 .. 6999 ---
Processed 6100/10000 (errors so far: 0)
Processed 6200/10000 (errors so far: 0)
Processed 6300/10000 (errors so far: 0)
Processed 6400/10000 (errors so far: 0)
Processed 6500/10000 (errors so far: 0)
Processed 6600/10000 (errors so far: 0)
Processed 6700/10000 (errors so far: 0)
Processed 6800/10000 (errors so far: 0)
Processed 6900/10000 (errors so far: 0)
Processed 7000/10000 (errors so far: 0)
✓ Chunk 6000-6999 saved. Total rows now: 7000

--- Processing 7000 .. 7999 ---
```

```
[ ]  Start coding or generate with AI.
```



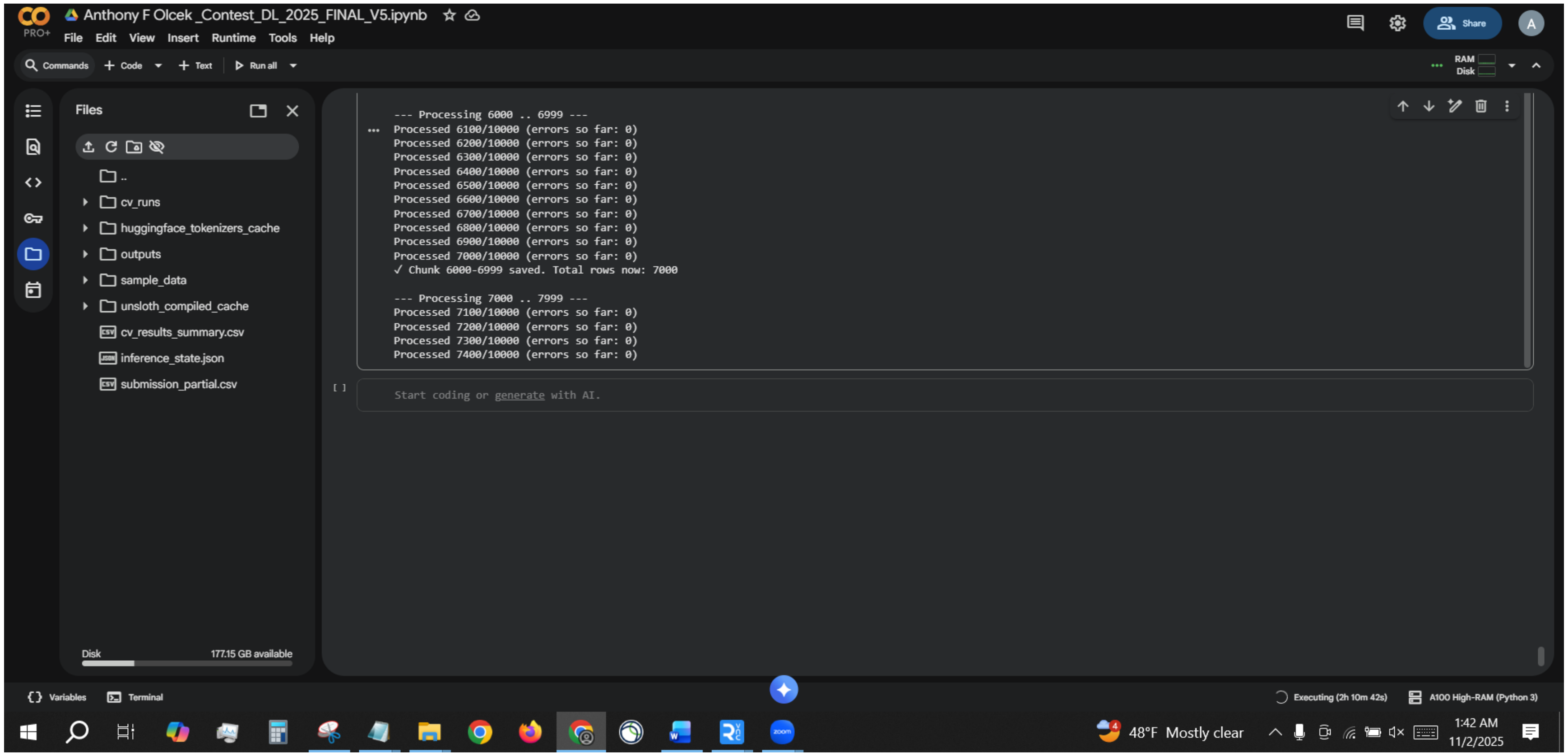
```
Processed 6000/10000 (errors so far: 0)
✓ Chunk 5000-5999 saved. Total rows now: 6000

--- Processing 6000 .. 6999 ---
Processed 6100/10000 (errors so far: 0)
Processed 6200/10000 (errors so far: 0)
Processed 6300/10000 (errors so far: 0)
Processed 6400/10000 (errors so far: 0)
Processed 6500/10000 (errors so far: 0)
Processed 6600/10000 (errors so far: 0)
Processed 6700/10000 (errors so far: 0)
Processed 6800/10000 (errors so far: 0)
Processed 6900/10000 (errors so far: 0)
Processed 7000/10000 (errors so far: 0)
✓ Chunk 6000-6999 saved. Total rows now: 7000

--- Processing 7000 .. 7999 ---
Processed 7100/10000 (errors so far: 0)
Processed 7200/10000 (errors so far: 0)
```

```
Start coding or generate with AI.
```



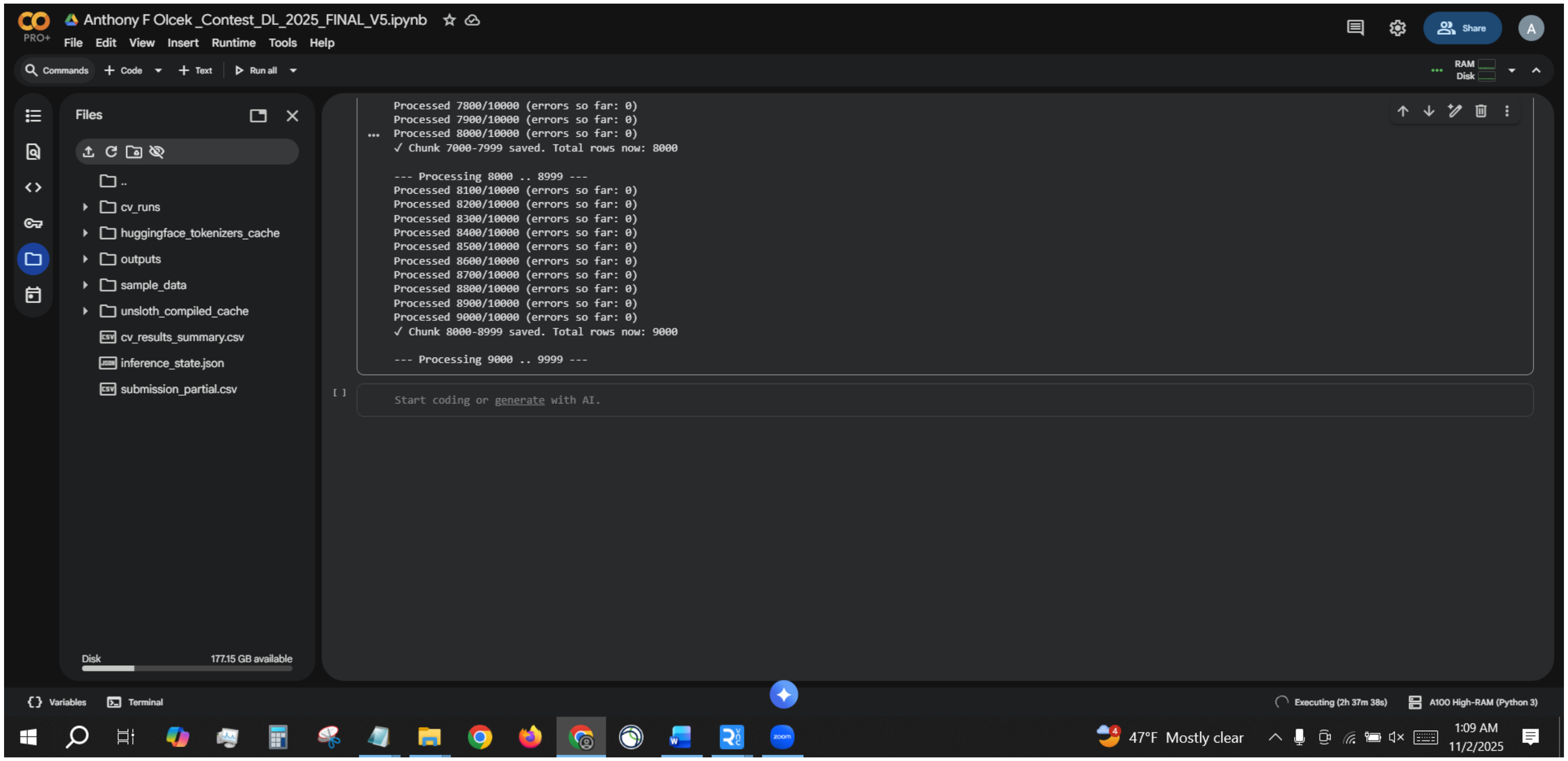
```
✓ Chunk 5000-5999 saved. Total rows now: 6000

--- Processing 6000 .. 6999 ---
Processed 6100/10000 (errors so far: 0)
Processed 6200/10000 (errors so far: 0)
Processed 6300/10000 (errors so far: 0)
Processed 6400/10000 (errors so far: 0)
Processed 6500/10000 (errors so far: 0)
Processed 6600/10000 (errors so far: 0)
Processed 6700/10000 (errors so far: 0)
Processed 6800/10000 (errors so far: 0)
Processed 6900/10000 (errors so far: 0)
Processed 7000/10000 (errors so far: 0)
✓ Chunk 6000-6999 saved. Total rows now: 7000

--- Processing 7000 .. 7999 ---
Processed 7100/10000 (errors so far: 0)
Processed 7200/10000 (errors so far: 0)
Processed 7300/10000 (errors so far: 0)
```

```
Start coding or generate with AI.
```



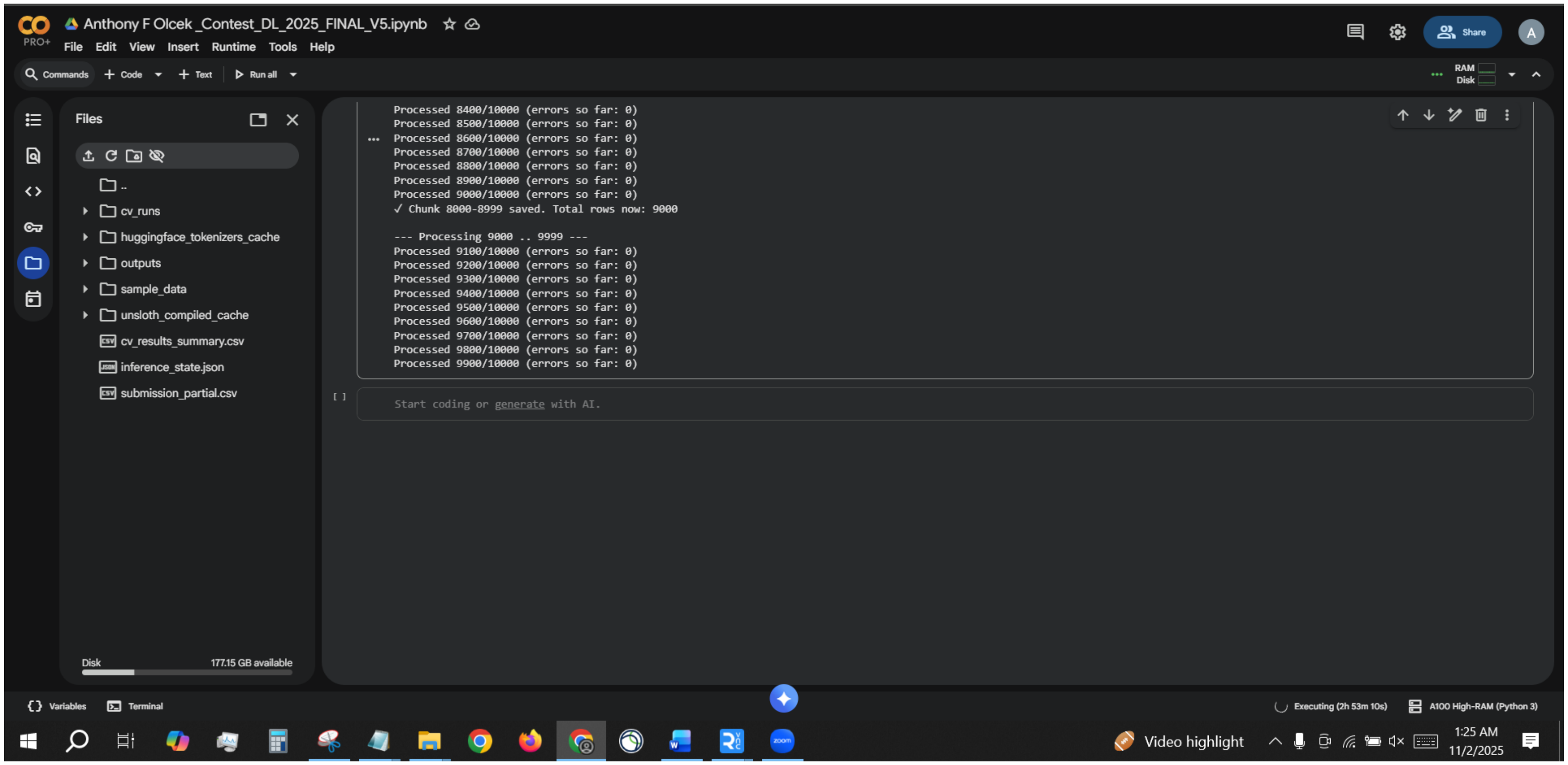
Anthony F Olcek _Contest_DL_2025_FINAL_V5.ipynb

```
--- Processing 6000 .. 6999 ---
Processed 6100/10000 (errors so far: 0)
Processed 6200/10000 (errors so far: 0)
Processed 6300/10000 (errors so far: 0)
Processed 6400/10000 (errors so far: 0)
Processed 6500/10000 (errors so far: 0)
Processed 6600/10000 (errors so far: 0)
Processed 6700/10000 (errors so far: 0)
Processed 6800/10000 (errors so far: 0)
Processed 6900/10000 (errors so far: 0)
Processed 7000/10000 (errors so far: 0)
✓ Chunk 6000-6999 saved. Total rows now: 7000

--- Processing 7000 .. 7999 ---
Processed 7100/10000 (errors so far: 0)
Processed 7200/10000 (errors so far: 0)
Processed 7300/10000 (errors so far: 0)
Processed 7400/10000 (errors so far: 0)
```



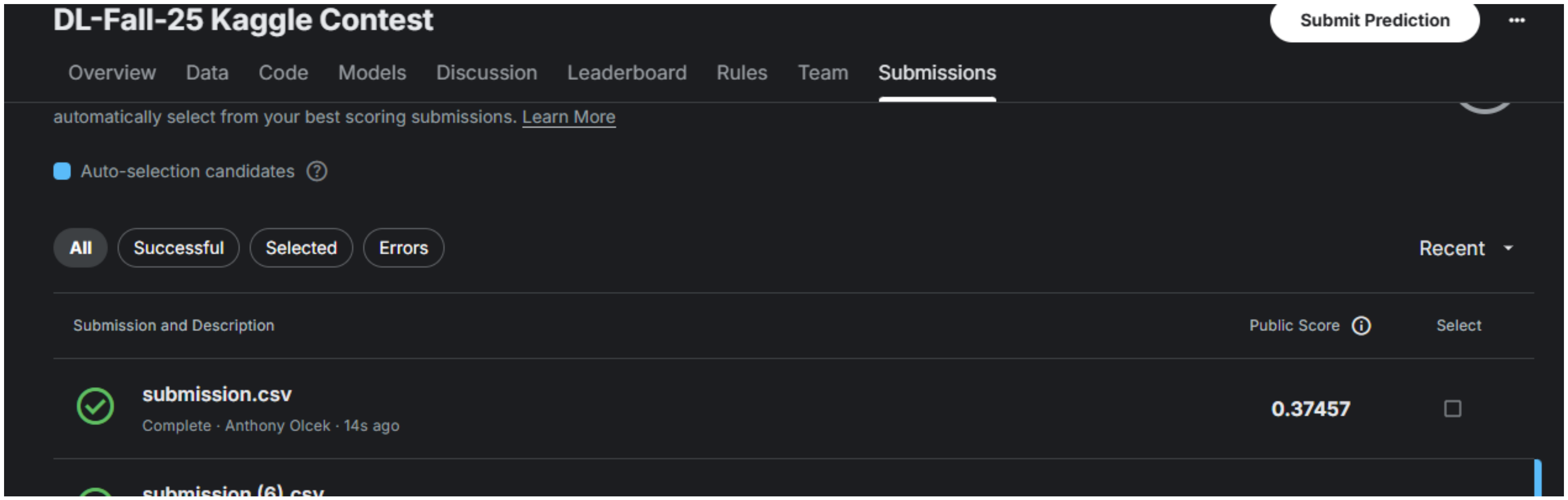
```
Processed 7800/10000 (errors so far: 0)
Processed 7900/10000 (errors so far: 0)
Processed 8000/10000 (errors so far: 0)
✓ Chunk 7000-7999 saved. Total rows now: 8000

--- Processing 8000 .. 8999 ---
Processed 8100/10000 (errors so far: 0)
Processed 8200/10000 (errors so far: 0)
Processed 8300/10000 (errors so far: 0)
Processed 8400/10000 (errors so far: 0)
Processed 8500/10000 (errors so far: 0)
Processed 8600/10000 (errors so far: 0)
Processed 8700/10000 (errors so far: 0)
Processed 8800/10000 (errors so far: 0)
Processed 8900/10000 (errors so far: 0)
Processed 9000/10000 (errors so far: 0)
✓ Chunk 8000-8999 saved. Total rows now: 9000

--- Processing 9000 .. 9999 ---
```

```
[ ] Start coding or generate with AI.
```



```
Processed 8400/10000 (errors so far: 0)
Processed 8500/10000 (errors so far: 0)
Processed 8600/10000 (errors so far: 0)
Processed 8700/10000 (errors so far: 0)
Processed 8800/10000 (errors so far: 0)
Processed 8900/10000 (errors so far: 0)
Processed 9000/10000 (errors so far: 0)
✓ Chunk 8000-8999 saved. Total rows now: 9000

--- Processing 9000 .. 9999 ---
Processed 9100/10000 (errors so far: 0)
Processed 9200/10000 (errors so far: 0)
Processed 9300/10000 (errors so far: 0)
Processed 9400/10000 (errors so far: 0)
Processed 9500/10000 (errors so far: 0)
Processed 9600/10000 (errors so far: 0)
Processed 9700/10000 (errors so far: 0)
Processed 9800/10000 (errors so far: 0)
Processed 9900/10000 (errors so far: 0)
```

```
[ ]  Start coding or generate with AI.
```



DL-Fall-25 Kaggle Contest

automatically select from your best scoring submissions. Learn More

🟦 Auto-selection candidates

Submission and Description	Public Score	Select
✓ **submission.csv** Complete · Anthony Olcek · 14s ago	**0.37457**	☐
submission (6).csv		