FINO PODEŠAVANJE MANJIH JEZIČNIH MODELA ZA DOMENSKI SPECIJALIZIRANU PRIMJENU

Franjo Čotić, Lea Masnec i Zlatko Stapić

SADRŽAJ

1. UVOD

- Umjetna inteligencija postaje standardna komponenta modernih aplikacija
- Korisnici očekuju:
  - Pametne asistente
  - Automatizirane analize
  - Inteligentnu podršku u donošenju odluka
- Razvojni timovi sve češće moraju integrirati AI u postojeće sustave

Cilj rada je **usporediti različite načine integracije AI funkcionalnosti u softverski proizvod**.

2. PREGLED PODRUČJA
2.1. KAKO INTEGRIRATI AI U APLIKACIJU

U praksi postoji više pristupa:

- API modeli
  - brza implementacija
  - model se izvršava na udaljenim serverima
- lokalni open-source modeli
  - veća kontrola nad podacima
  - mogućnost prilagodbe modela
- vlastiti ML modeli
  - specifični prediktivni zadaci
  - potpuno prilagođeni domeni

Svaki pristup ima različite kompromise u trošku, performansama i kontroli

2.2. MALI DOMENSKI MODELI

Zašto mali domenski modeli?

- LLM-ovi nisu uvijek optimalni za poslovne aplikacije.
- Problemi:
  - privatnost podataka
  - latencija
  - troškovi API poziva
  - ograničena prilagodba domeni
- Alternativa:

  fine-tuning manjih open-source modela
- Prednosti:
  - lokalno izvođenje
  - uži kontekst
  - manji računalni zahtjevi
  - veća kontrola nad modelom

3. METODOLOGIJA I PLAN

- Sustav chatbota koji odgovara na specifična pitanja (solarne elektrane)
- Podaci iz raznih izvora - QA parovi za treniranje
- Fino podešeni manji javno dostupni model
- Nije računalno „pretežak”
- Predikcija kwh uz pomoć drugog modela strojnog učenja

3.1 PLAN PRAKTIČNE IMPLEMENTACIJE

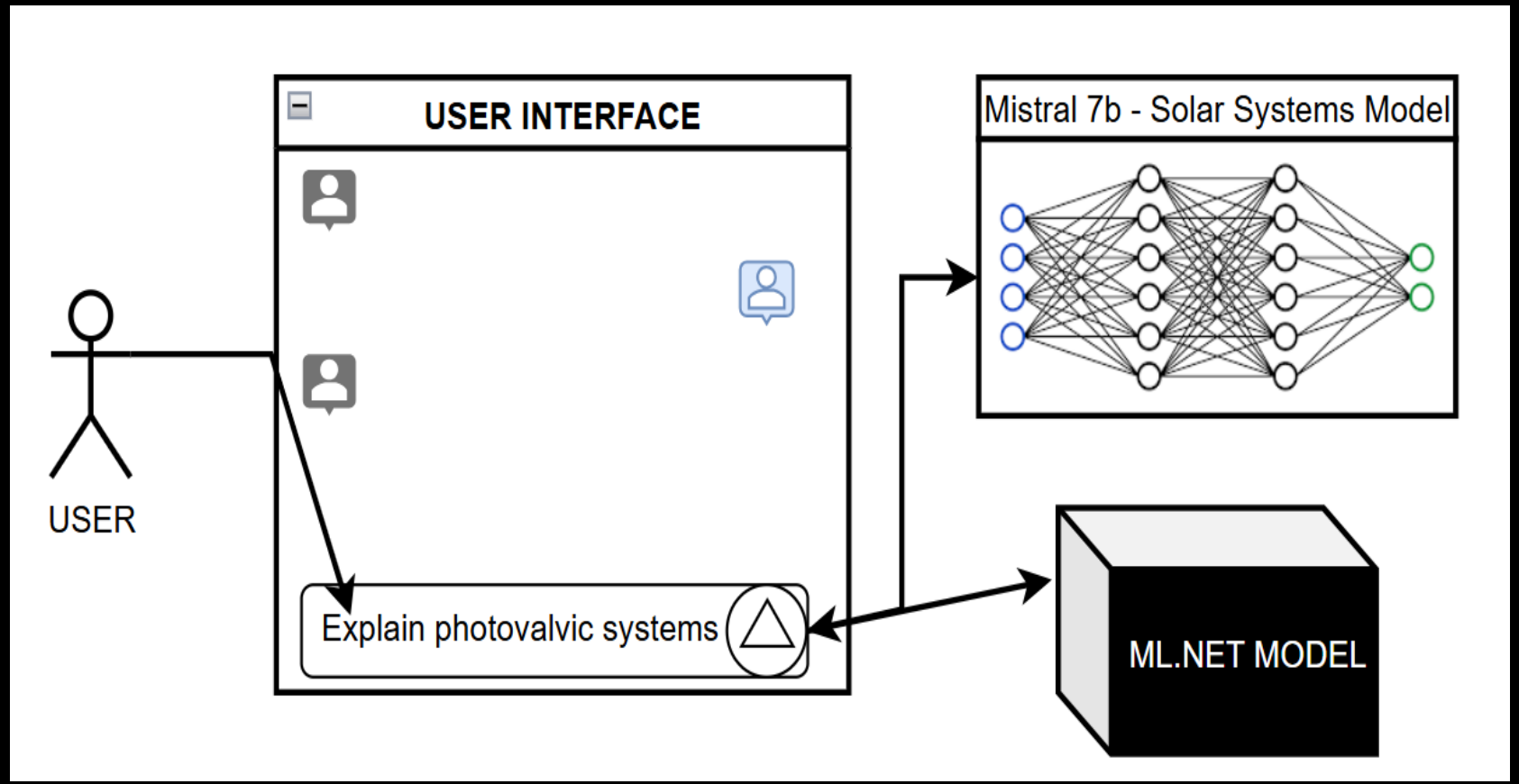
- Programski jezici: C#, Python
- Softverski okviri: .NET, WPF, ML.NET
- Softverske biblioteke: Llama Sharp
- Platforma za obuku modela: Google Colab, Jupyter
- Platforma za preuzimanje modela i setova podataka: Hugging Face
- Korišteni alati umjetne inteligencije: ChatGPT model 5.2

3.1 PLAN PRAKTIČNE IMPLEMENTACIJE

- Programski jezici: C#, Python
- Softverski okviri: .NET, WPF, ML.NET
- Softverske biblioteke: Llama Sharp
- Platforma za obuku modela: Google Colab, Jupyter
- Platforma za preuzimanje modela i setova podataka: Hugging Face
- Korišteni alati umjetne inteligencije: ChatGPT model 5.2

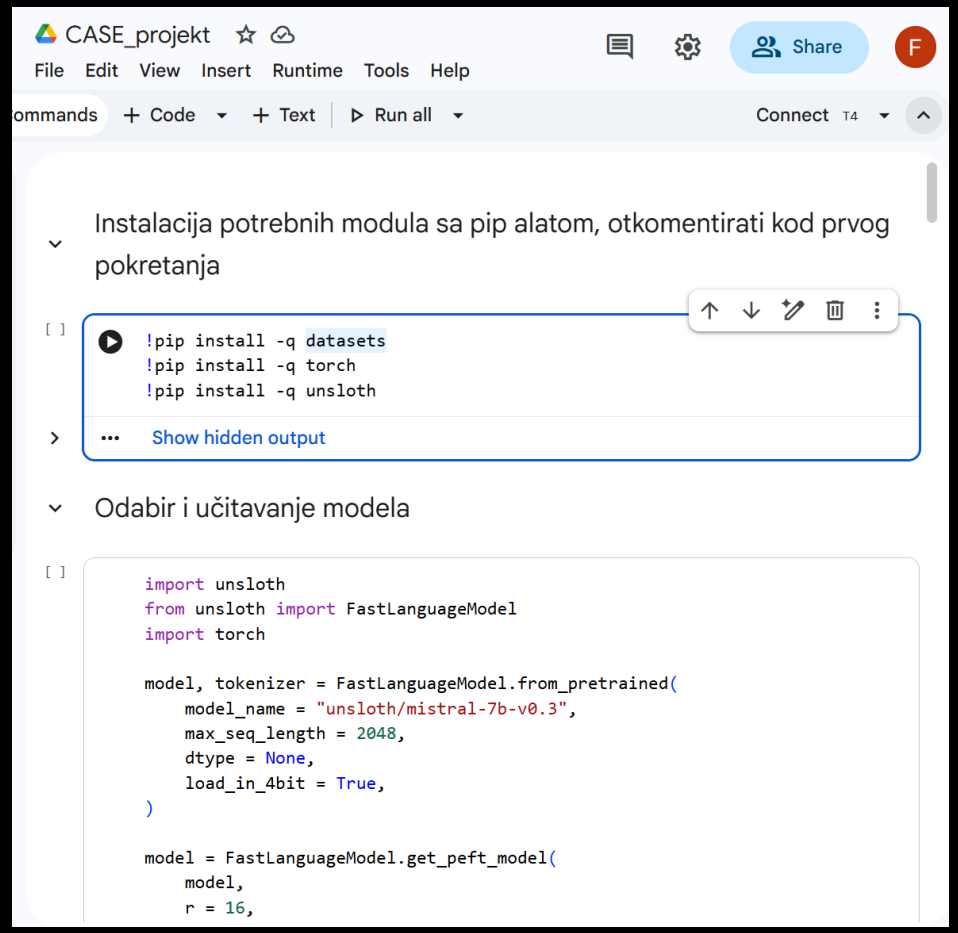
3.2 KORIŠTENE TEHNOLOGIJE

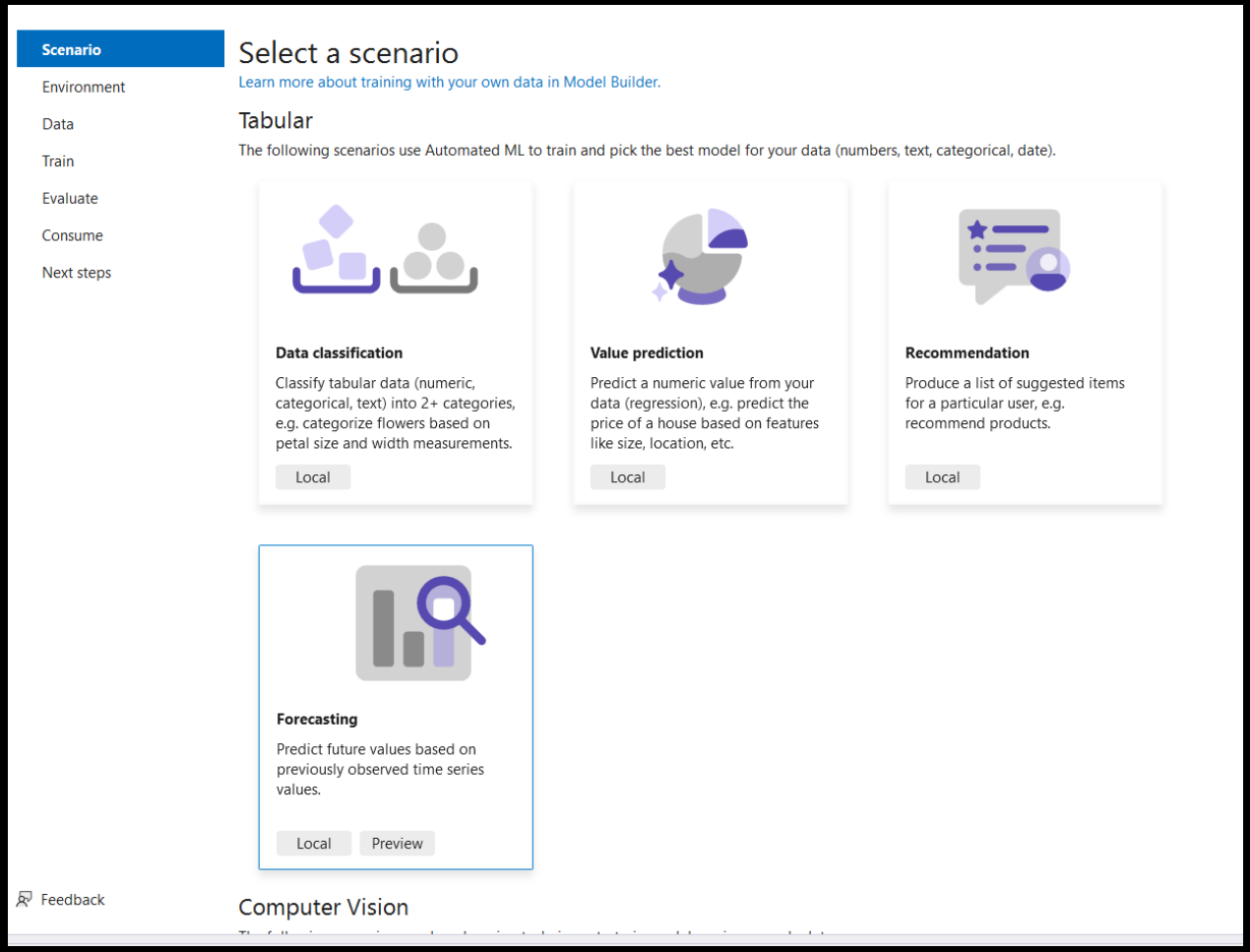
- Mistral 7b i ML.NET model



3.2 KORIŠTENE TEHNOLOGIJE

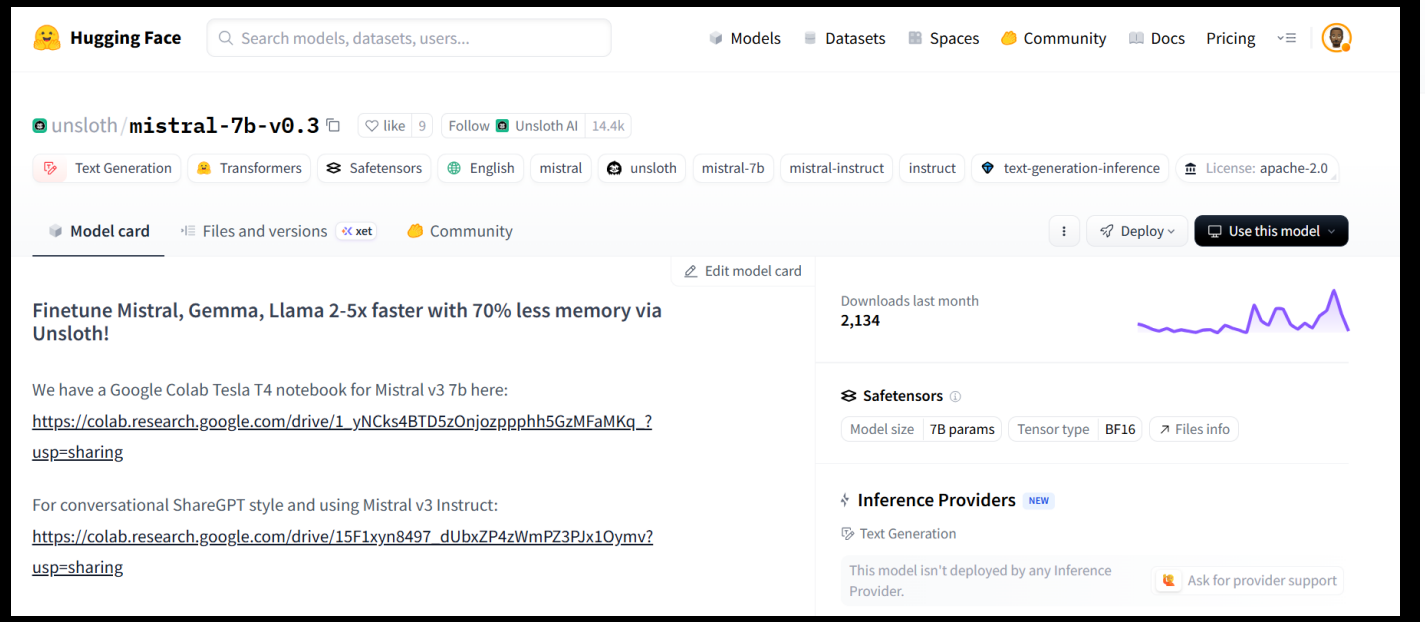
- ML.NET model za predviđanje
- Colab bilježnica za treniranje

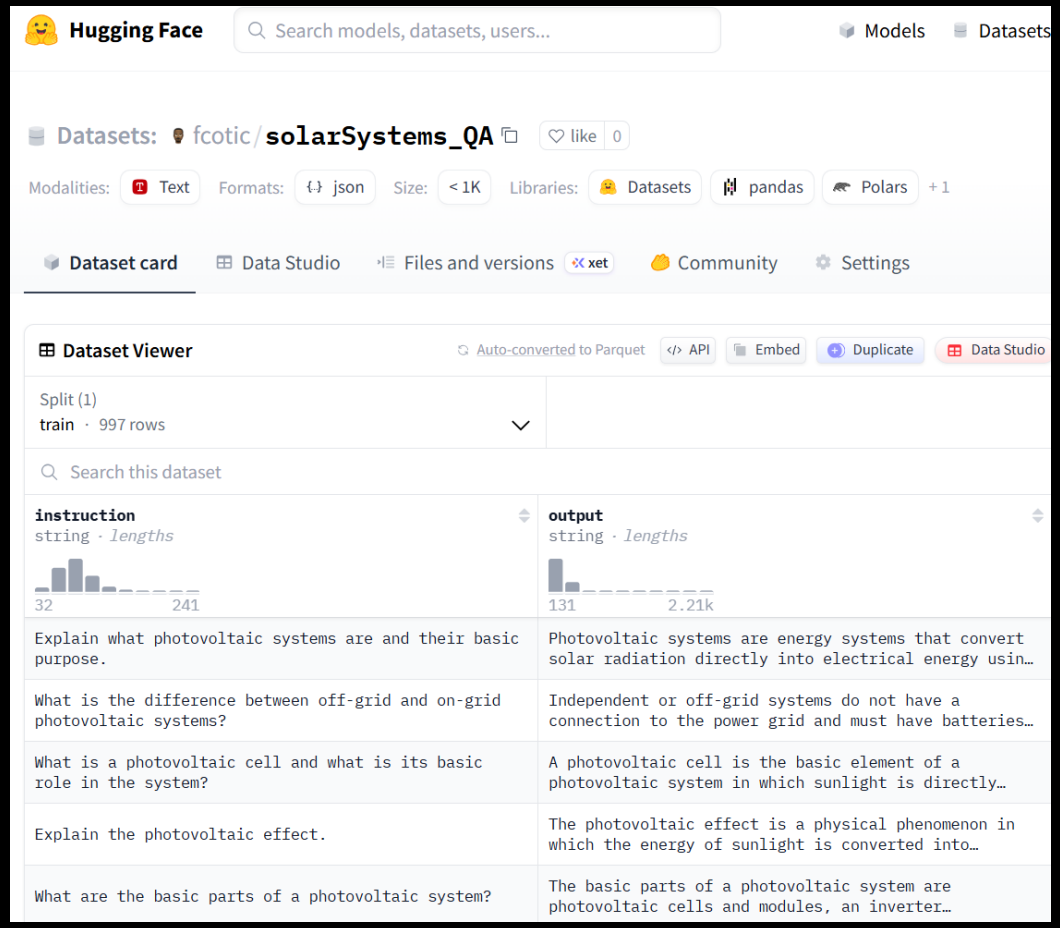




3.2 KORIŠTENE TEHNOLOGIJE

- Hugging Face
  - učitavanje vlastitog QA dataseta
  - preuzimanje osnovnog modela





4. REZULTATI FINO PODEŠAVANJE

Odabir i učitavanje modela

```
import unsloth
from unsloth import FastLanguageModel
import torch

model, tokenizer = FastLanguageModel.from_pretrained(
    model_name = "unsloth/mistral-7b-v0.3",
    max_seq_length = 2048,
    dtype = None,
    load_in_4bit = True,
)

model = FastLanguageModel.get_peft_model(
    model,
    r = 16,
    target_modules = ["q_proj", "k_proj", "v_proj", "o_proj",
                      "gate_proj", "up_proj", "down_proj",],
    lora_alpha = 16,
    lora_dropout = 0,
    bias = "none",
    use_gradient_checkpointing = "unsloth",
    random_state = 3407,
)
```

Učitavanje dataseta i tokeniziranje podataka za davanje modelu za treniranje

```
prompt = """Below is an instruction that describes a task, paired with an input that prov
### Instruction:
{}
### Response:
{}"""

EOS_TOKEN = tokenizer.eos_token
def promptFormatter(prompts):
    instructions = prompts["instruction"]
    outputs = prompts["output"]
    texts = []
    for instruction, output in zip(instructions, outputs):
        text = prompt.format(instruction, output) + EOS_TOKEN
        texts.append(text)
    return { "text" : texts, }
pass

from datasets import load_dataset
dataset = load_dataset("fcotic/solarSystems_QA", split="train")
dataset = dataset.map(promptFormatter, batched = True,)
print(dataset[0])
```

4. REZULTATI
FINO PODEŠAVANJE

- Stvaranje trainer-a
  - Model, tokenizer, dataset
  - Argumenti treniranja
- Pokretanje treniranja

```
trainer = SFTTrainer(
    model = model,
    tokenizer = tokenizer,
    train_dataset = dataset,
    dataset_text_field = "text",
    max_seq_length = 2048,
    dataset_num_proc = 2,
    packing = False,
    args = TrainingArguments(
        per_device_train_batch_size = 1,
        gradient_accumulation_steps = 8,
        warmup_steps = 5,
        max_steps = 60,
        learning_rate = 2e-4,
        fp16 = not is_bfloat16_supported(),
        bf16 = is_bfloat16_supported(),
        logging_steps = 1,
        optim = "adamw_8bit",
        weight_decay = 0.01,
        lr_scheduler_type = "linear",
        seed = 3407,
        output_dir = "outputs",
    ),
)
```

```
Unsloth: Tokenizing ["text"] (num_proc=6): 100% 997/997
```

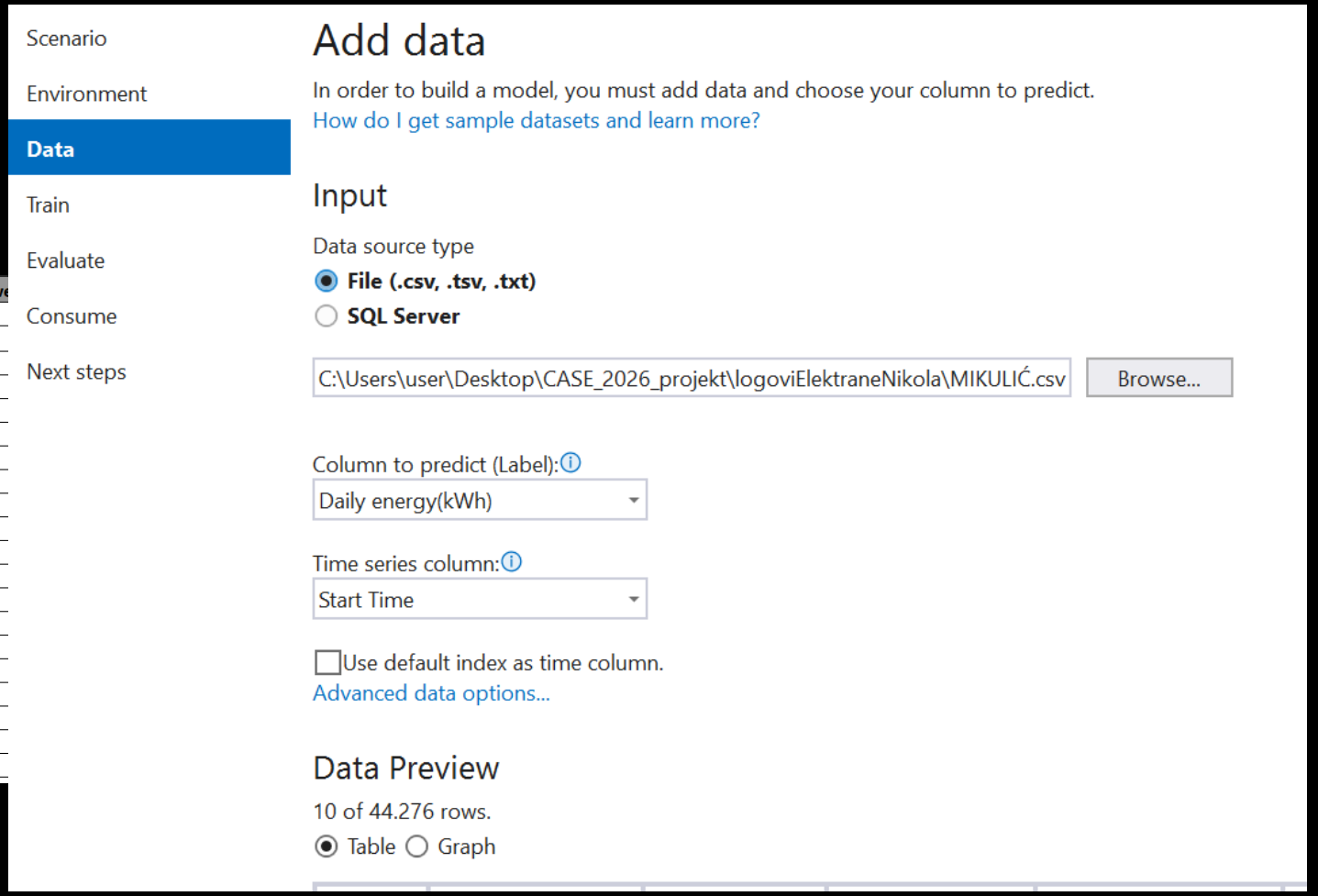
Fino podešavanje na pripremljenim podacima

```
trainer_stats = trainer.train()
```

4. REZULTATI ML.NET MODEL

- Podaci za ML.NET model
- Stupac za predikciju

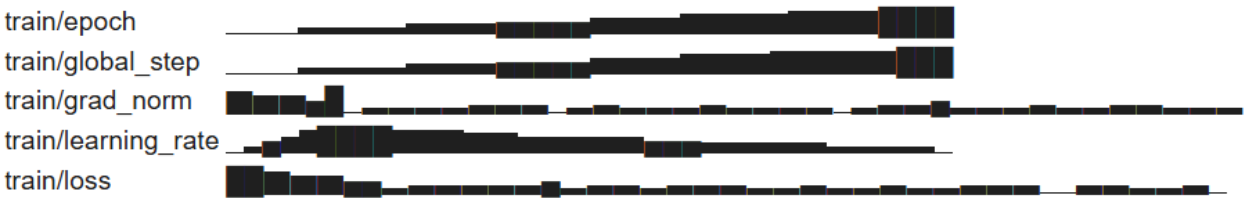
Site Name	Start Time	Active power(kW)	Annual energy(kWh)	Conve
Mikulić	2024-11-10 07:45:00	0,263	0,40	
Mikulić	2024-11-10 08:00:00	0,352	0,54	
Mikulić	2024-11-10 08:15:00	0,286	0,62	
Mikulić	2024-11-10 08:30:00	3,631	1,01	
Mikulić	2024-11-10 08:45:00	4,276	1,91	
Mikulić	2024-11-10 09:00:00	4,512	3,01	
Mikulić	2024-11-10 09:15:00	3,145	4,03	
Mikulić	2024-11-10 09:30:00	2,618	4,86	
Mikulić	2024-11-10 09:45:00	0,627	5,47	
Mikulić	2024-11-10 10:00:00	0,692	5,64	
Mikulić	2024-11-10 10:15:00	3,916	6,42	
Mikulić	2024-11-10 10:30:00	2,243	7,34	
Mikulić	2024-11-10 10:45:00	5,907	8,28	
Mikulić	2024-11-10 11:00:00	4,633	9,49	
Mikulić	2024-11-10 11:15:00	3,857	10,48	
Mikulić	2024-11-10 11:30:00	0,708	10,86	
Mikulić	2024-11-10 11:45:00	2,222	11,12	
Mikulić	2024-11-10 12:00:00	0,969	11,45	
Mikulić	2024-11-10 12:15:00	1,011	11,66	
Mikulić	2024-11-10 12:30:00	1,986	11,91	



4. REZULTATI

Colab bilježnica

Run history:



Run summary:

total_flos	4594978264031232.0
train/epoch	0.48144
train/global_step	60
train/grad_norm	1.25656
train/learning_rate	0.0
train/loss	0.8401
train_loss	1.08627
train_runtime	413.059
train_samples_per_second	1.162
train_steps_per_second	0.145

ML.NET model

Train

Specify a time to train for evaluating various models.
How long should I train for?

Training setup summary

Time to train (seconds): 100

Horizon: 12

Advanced training options...

Train again ✓ Training complete

Training results

Best RMSE:	1,6831
Best model:	ForecastBySsa
Training time:	33,66 seconds
Models explored (total):	5
Generated code-behind:	DailyEnergyPredictionModel.consumption.cs, DailyEnergyPredictionModel.training.cs

Next step

4. REZULTATI - IMPLEMENTACIJA

```
1 reference
private void InitializeModel()
{
    try
    {
        string modelPath = System.IO.Path.Combine(KnownFolders.GetPath(KnownFolder.Downloads), "mistral_7B_solarSystems.q4_k_m.gguf");

        var parameters = new ModelParams(modelPath)
        {
            ContextSize =4096,
            GpuLayerCount =80
        };
        var weights = LLamaWeights.LoadFromFile(parameters);
        var context = weights.CreateContext(parameters);
        executor = new InteractiveExecutor(context);


        var chatHistory = new ChatHistory();
        chatHistory.AddMessage(AuthorRole.System,
        "You are a strictly domain-limited assistant specialized ONLY in solar systems and residential solar power plants installation
        "CRITICAL RULES:\n" +
        "1. You MUST answer ONLY questions related to solar panels, photovoltaic systems, inverters, batteries, mounting systems, insta
        "2. If the question is NOT related to solar systems in any way, you MUST refuse.\n" +
        "3. When refusing, respond EXACTLY with:\n" +
        "'I'm sorry, but I can only answer questions related to solar systems and photovoltaic power plants.'\n" +
        "4. Do NOT provide any additional information when refusing.\n" +
        "5. Do NOT attempt to reinterpret unrelated questions as solar-related.\n" +
        "6. If a question seems like it isn't related to the topic DO NOT ANSWEAR AND REFUSE AS INSTRUCTED");
        chatSession = new ChatSession(executor, chatHistory);
        inferenceParams = new InferenceParams() { MaxTokens =4096, AntiPrompts = new List<string> { "User:" } };
    }
    catch (System.Exception ex)
    {
        AppendText($"Error loading model: {ex.ToString()}\n");
    }
```

- Inicijalizacija modela
- Postavljanje sistemske poruke

4. REZULTATI - IMPLEMENTACIJA

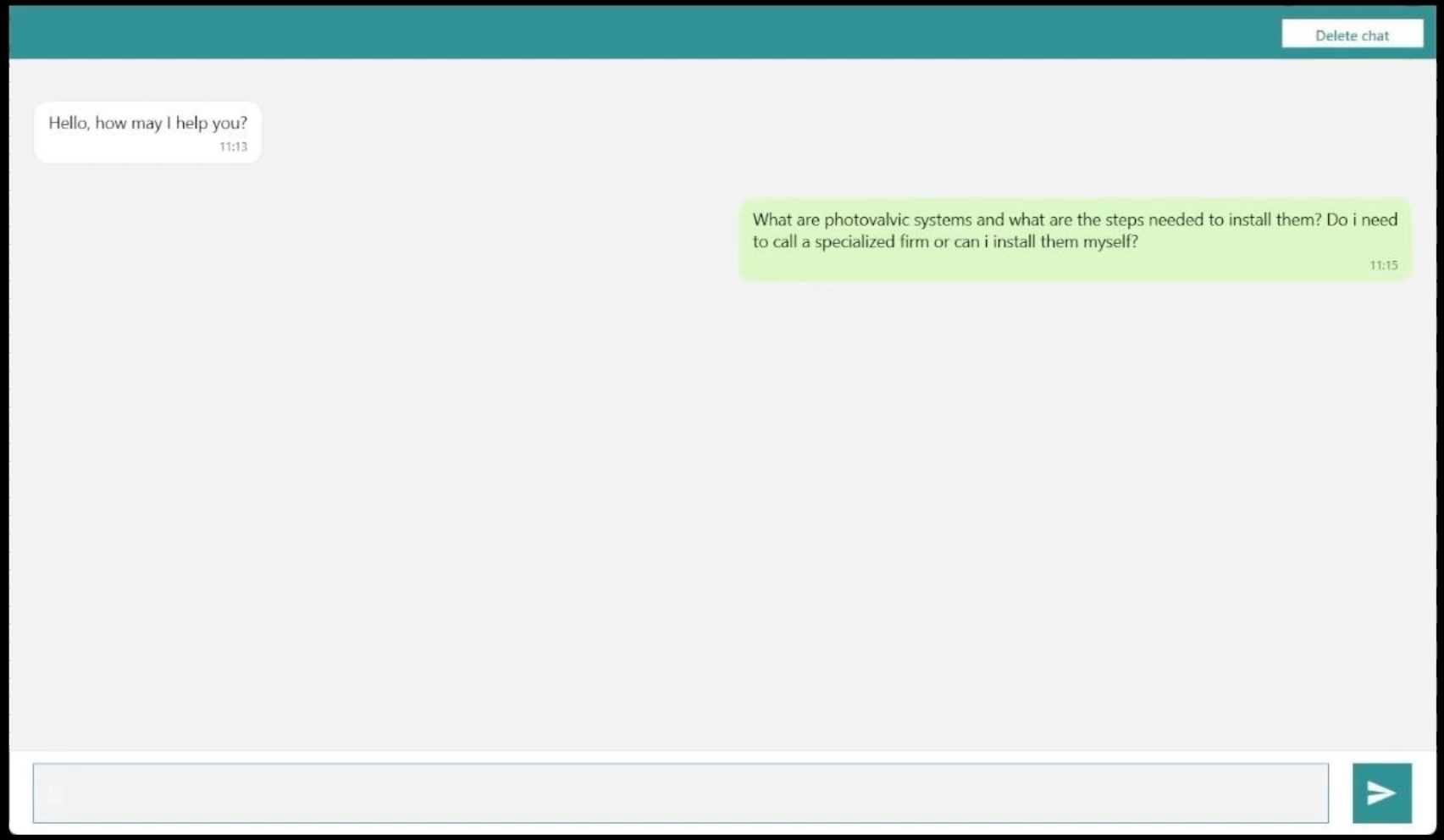
- Slanje upita modelu
- Prikupljanje i prikaz odgovora

```
if (chatSession != null && inferenceParams != null)
{
    var responseBuilder = new System.Text.StringBuilder();
    await foreach (var token in chatSession.ChatAsync(new ChatHistory.Message(AuthorRole.User, message), inferenceParams))
    {
        responseBuilder.Append(token);
    }
    var response = responseBuilder.ToString();
    response = Regex.Replace(response, @"\s*User:\s*$", string.Empty, RegexOptions.IgnoreCase);
    Messages.Add(new Message { Role = MessageRoleType.Assistant, Content = response, Timestamp = System.DateTime.Now });
}
else
{
    Messages.Add(new Message { Role = MessageRoleType.Assistant, Content = "Model not loaded.", Timestamp = System.DateTime.Now });
}
```

- Slanje poruke chatbotu
- Provjera riječi za predikciju
- Prikaz predikcije generirane energije

```
private async void btnSend_Click(object sender, RoutedEventArgs e)
{
    var message = txtMessage.Text.Trim();
    if (string.IsNullOrEmpty(message))
    {
        MessageBox.Show("Please enter a message.", "CHATBOT", MessageBoxButton.OK, MessageBoxImage.Warning);
        return;
    }

    var userMsg = new Message { Role = MessageRoleType.User, Content = message, Timestamp = System.DateTime.Now };
    Messages.Add(userMsg);

    txtMessage.Text = string.Empty;
    txtMessage.Focus();

    if (ContainsPredictionKeyword(message))
    {
        var span = ParseTimeSpanFromQuery(message);
        if (span == null)
        {
            Messages.Add(new Message { Role = MessageRoleType.Assistant, Content = "I couldn't parse the time period from your request. P
            return;
        }

        int intervals = Get15MinIntervalCount(span.Value);
        try
        {
            var output = DailyEnergyPredictionModel.Predict(null, intervals);
            if (output != null && output.Daily_energy_kWh_ != null && output.Daily_energy_kWh_.Length >0)
            {
                var preds = output.Daily_energy_kWh_;
                var sb = new System.Text.StringBuilder();
                var over15MinAverage = new double[preds.Length];

                sb.AppendLine($"Prediction for the next {span.Value}: \n");
                for (int i = 0; i < preds.Length; i++)
                {
                    sb.AppendLine($"Interval {i +1}: {preds[i].ToString()} kWh");
                    if (i != 0) over15MinAverage[i] = preds[i] - preds[i - 1];
                    else over15MinAverage[i] = preds[i];
                }

                var avg = over15MinAverage.Average();
                sb.AppendLine($"\nAverage over 15 minute period: {avg.ToString("F2")} kWh");

                Messages.Add(new Message { Role = MessageRoleType.Assistant, Content = sb.ToString(), Timestamp = System.DateTime.Now });
                return;
```

4. REZULTATI


Delete chat
Hello, how may I help you?
11:13
What are photovalvic systems and what are the steps needed to install them? Do i need to call a specialized firm or can i install them myself?
11:15

5. DISKUSIJA

- Izvlačenje i revidiranje pitanja – ChatGPT – 1000 QA parova
- PEFT – LORA - 0.58% parametara podešeno
- Mistral 7b – Google Colab – limitiran hardver
- Komercijalna primjena
- Vremenska i novčana isplativost

6. ZAKLJUČAK

- Pregled i usporedba pristupa integraciji AI funkcionalnosti
- Proveden LoRA fine-tuning modela Mistral 7B
- Proveden trening ML.NET modela
- Integracija AI modela u .NET aplikaciju
- Pregled isplativosti

HVALA
NA POZORNOSTI!