Quickstart

GRadio - это пакет Python с открытым исходным кодом, который позволяет вам быстро создать демонстрационное или веб-приложение для вашей модели машинного обучения, API или любой произвольной функции Python.

Затем вы можете поделиться ссылкой на свое демонстрационное или веб-приложение всего за несколько секунд, используя встроенные функции совместного использования GRadio. Никаких навыков работы с JavaScript, CSS или веб-хостингом не требуется!

Для создания вашей собственной демонстрации потребуется всего несколько строк на Python.

Установка

Обязательное условие: для Gradio требуется Python версии 3.10 или выше.

Мы рекомендуем установить Radio с помощью **pip**, который включен в Python по умолчанию. Запустите это в своем терминале или командной строке:

pip install --upgrade gradio

https://www.gradio.app/guides/quickstart

```
# установка Gradio
# pip install --upgrade gradio
```

Как создать Interface

Класс Interface имеет 3 обязательных параметра:

Interface(fn, inputs, outputs, ...)

Это параметры:

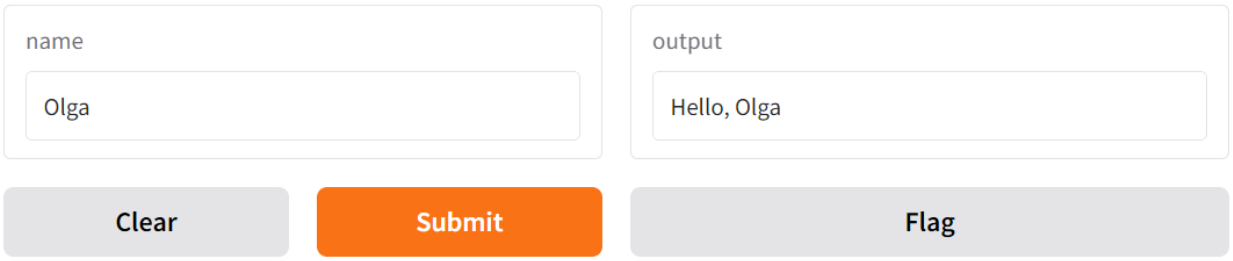
- **fn**: функция прогнозирования, обернутая интерфейсом Gradio. Эта функция может принимать один или несколько параметров и возвращать одно или несколько значений
- **inputs**: тип(ы) компонента(ов) ввода. Gradio предоставляет множество готовых компонентов, таких как "image" или "mic".
- **outputs**: тип(ы) компонента(ов) вывода. Опять же, Gradio предоставляет множество предварительно созданных компонентов, например, " image" или "label".

https://www.gradio.app/docs

Пример 1

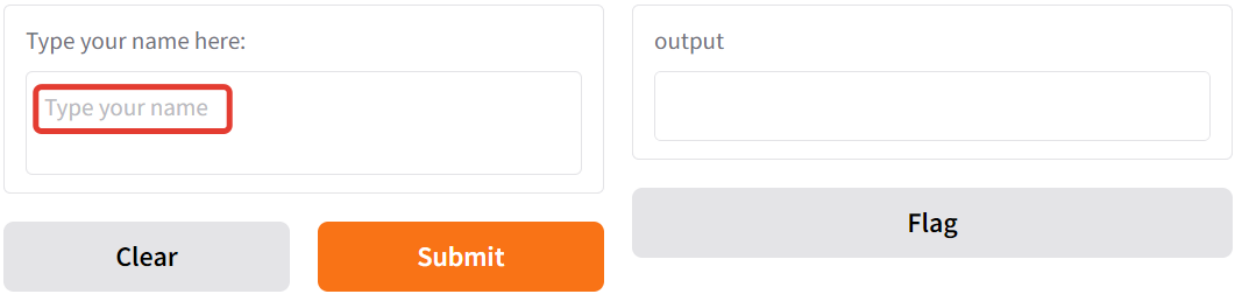
```
# пример "Hello World"
import gradio as gr

#  функция greet()
def greet(name):
    return "Hello, " + name

# интерфейс Gradio Interface с тремя
аргументами, fn, inputs и outputs
demo = gr.Interface(fn=greet, inputs="text",
outputs="text")

# метод launch() для Interface demo
demo.launch()
```

https://www.gradio.app/guides/quickstart https://www.gradio.app/guides/the-interface-class



Пример 2

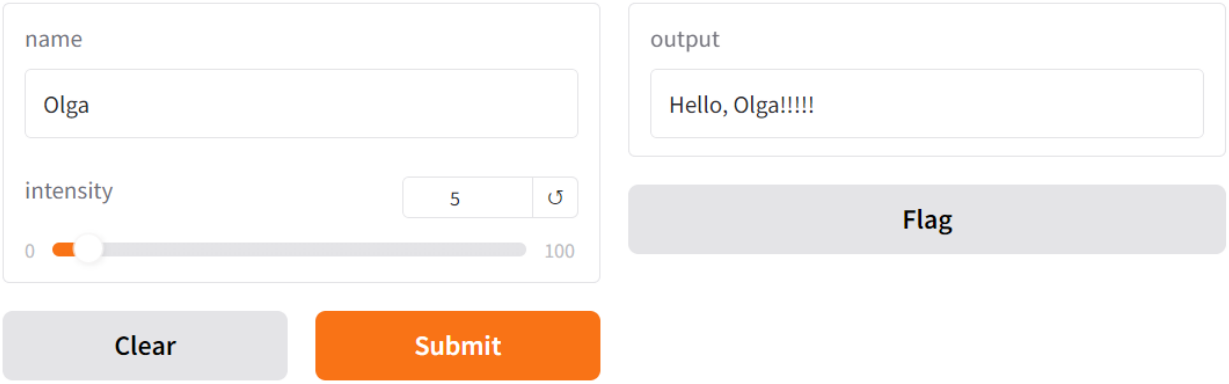
```
import gradio as gr


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


# Инстанцируем класс Textbox
textbox = gr.Textbox(label="Type your name here:", placeholder="Type your name", lines=2)

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

https://www.gradio.app/guides/quickstart https://www.gradio.app/guides/the-interface-class

Пример 3

```
import gradio as gr

def greet(name, intensity):
    return "Hello, " + name + "!" *
int(intensity)

demo = gr.Interface(
    fn=greet,
    inputs=["text", "slider"],
    outputs=["text"],
)

demo.launch()
```



Sharing Your Demo

Что хорошего в красивой демонстрации, если вы не можете поделиться ею?

GRadio позволяет вам легко делиться демонстрацией машинного обучения, не беспокоясь о необходимости размещения на веб-сервере.

Просто установите **share=True в launch()**, и для вашей демонстрации будет создан **общедоступный URL-адрес**.

Давайте вернемся к нашему демонстрационному примеру, но изменим последнюю строку следующим образом:

Пример 4

```
import gradio as gr

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

demo = gr.Interface(fn=greet, inputs="textbox", outputs="textbox")

# Поделитесь своей демонстрацией, указав всего 1 дополнительный параметр
demo.launch(share=True)  # Share your demo with just 1 extra parameter
```


Running on public URL:
https://8b9007abc1629124e1.gradio.live

Gradio Components

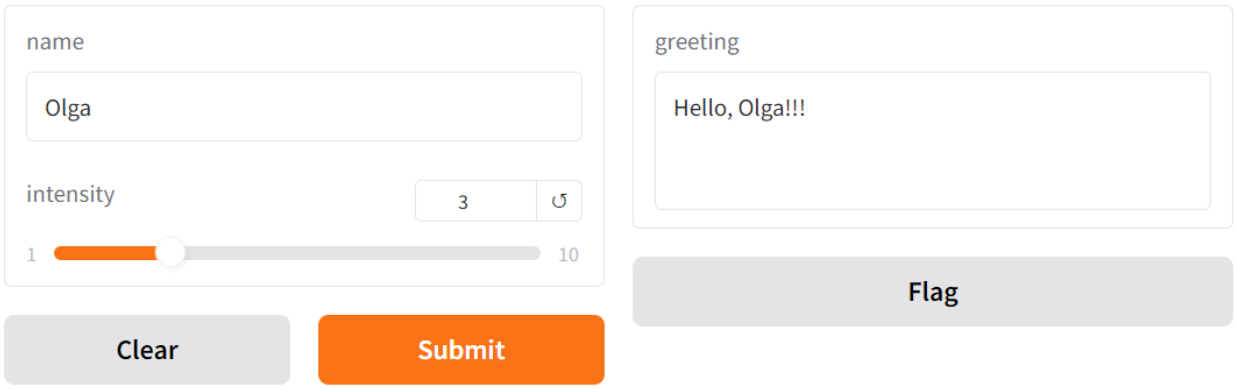
Gradio включает в себя более 30 готовых компонентов (а также множество пользовательских компонентов, созданных сообществом), которые можно использовать в качестве входных или выходных данных в вашей демонстрации.

Эти компоненты соответствуют распространенным типам данных в машинном обучении и науке о данных, например, **gr.Image** предназначен для обработки входных и выходных изображений, компонент **gr.Label** отображает классификационные метки и вероятности, компонент **gr.LinePlot** отображает линейные графики и так далее.

https://www.gradio.app/guides/quickstart https://www.gradio.app/guides/the-interface-class

Components Attributes

Если вы используете реальные классы для gr.Textbox и gr.Slider вместо ярлыков строк, у вас есть доступ к гораздо большей настраиваемости с помощью атрибутов компонента.




```
import gradio as gr

def greet(name, intensity):
    return "Hello, " + name + "!" * intensity

demo = gr.Interface(
    fn=greet,

    inputs=["text", gr.Slider(value=2, minimum=1, maximum=10, step=1)],

    outputs=[gr.Textbox(label="greeting", lines=3)],
)

demo.launch()
```

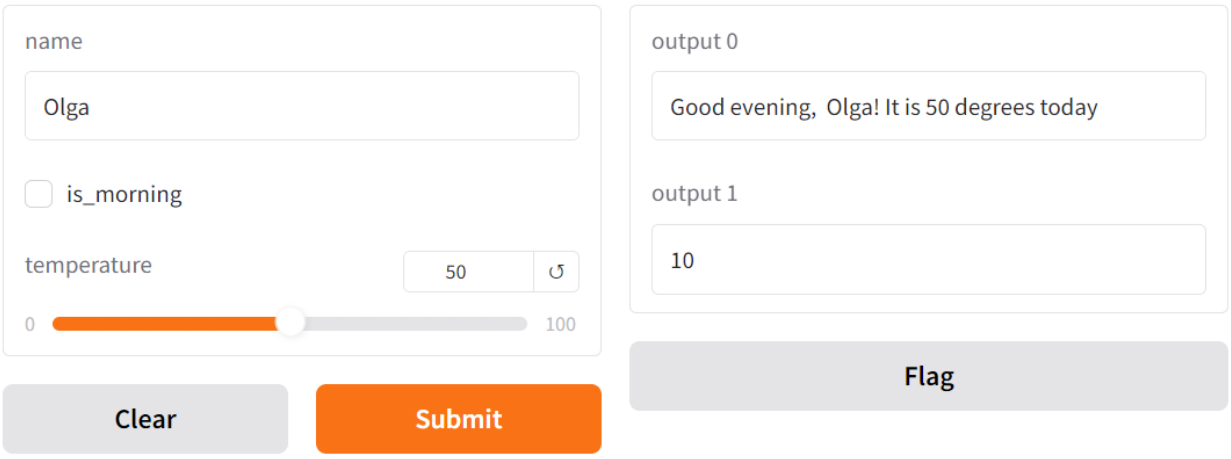
Multiple Input and Output Components

Предположим, у вас есть более сложная функция, также с несколькими выходными данными. В приведенном ниже примере мы определяем функцию, которая принимает строку, логическое значение и число и возвращает строку и число.

Точно так же, как каждый компонент в списке входных данных соответствует одному из параметров функции по порядку, каждый компонент в списке выходных данных соответствует одному из значений, возвращаемых функцией по порядку.

```
import gradio as gr

def greet(name, is_morning, temperature):

    salutation = "Good morning, " if is_morning
else "Good evening, "

    greeting = f"{salutation} {name}! It is
{temperature} degrees today"

    celsius = (temperature - 32) * 5 / 9

    return greeting, round(celsius, 2)

demo = gr.Interface(
    fn=greet,
    inputs=["text", "checkbox", gr.Slider(0,
100)],
    outputs=["text", "number"],
)
demo.launch()
```

https://www.gradio.app/guides/quickstart https://www.gradio.app/guides/the-interface-class



An Image Example

При использовании компоненты Image в качестве входных данных ваша функция получит массив NumPy с формой (высота, ширина, 3), где последнее измерение представляет значения RGB. Мы также вернем изображение в виде массива NumPy.

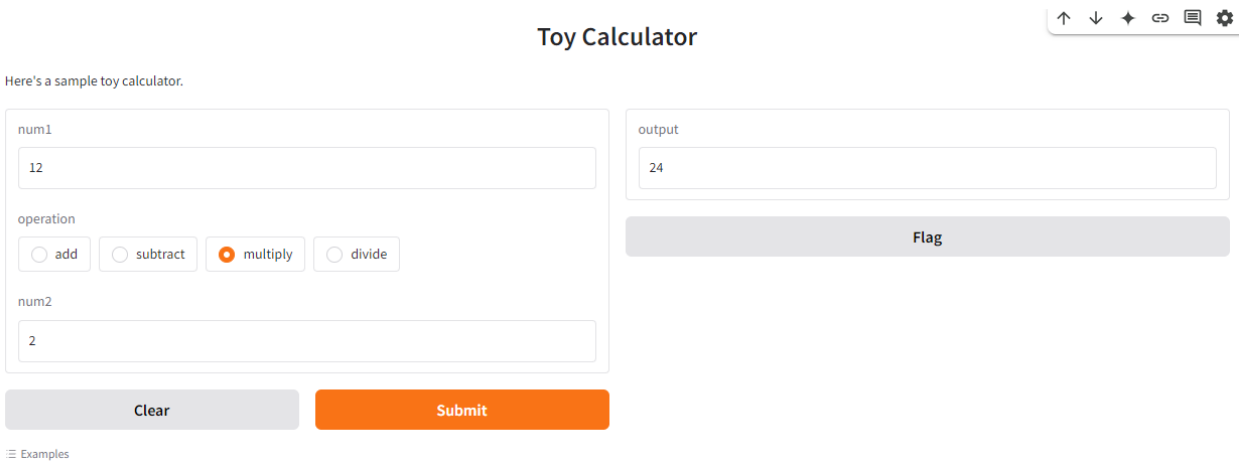

```
import numpy as np
import gradio as gr

def sepia(input_img):
    sepia_filter = np.array([
        [0.393, 0.769, 0.189],
        [0.349, 0.686, 0.168],
        [0.272, 0.534, 0.131]
    ])
    sepia_img = input_img.dot(sepia_filter.T)
    sepia_img /= sepia_img.max()
    return sepia_img

demo = gr.Interface(sepia, gr.Image(), "image")
demo.launch()
```

https://www.gradio.app/guides/quickstart https://www.gradio.app/guides/the-interface-class

Example Inputs

```
import gradio as gr

def calculator(num1, operation, num2):
    if operation == "add":
        return num1 + num2
    elif operation == "subtract":
        return num1 - num2
    elif operation == "multiply":
        return num1 * num2
    elif operation == "divide":
        if num2 == 0:
            raise gr.Error("Cannot divide by
zero!")
        return num1 / num2

demo = gr.Interface(
    calculator,
    [
        "number",
        gr.Radio(["add", "subtract", "multiply",
"divide"]),
        "number"
    ],
    "number",
        examples=[
        [45, "add", 3],
        [3.14, "divide", 2],
```

```
        [144, "multiply", 2.5],
        [0, "subtract", 1.2],
    ],
    title="Toy Calculator",
    description="Here's a sample toy
calculator.",
)

demo.launch()
```



Descriptive Content

В конструкторе интерфейса есть три аргумента, указывающих, куда следует поместить содержимое:

- **title**: который принимает текст и может отображать его в самом верху интерфейса, а также становится заголовком страницы.

- **description**: который принимает текст в формате markdown или HTML и размещает его прямо под заголовком.

- **article** который также принимает текст в формате markdown или HTML и размещает его под интерфейсом.

Другим полезным аргументом ключевого слова является label=, который присутствует в каждом компоненте. Это изменяет текст метки в верхней части каждого компонента.

Вы также можете добавить аргумент ключевого слова info= к таким элементам формы, как Textbox или Radio, чтобы предоставить дополнительную информацию об их использовании.

gr.Number(label='Age', info='In years, must be greater than 0')

Additional Inputs within an Accordion

```
import gradio as gr

def generate_fake_image(prompt, seed,
initial_image=None):
    return f"Used seed: {seed}",
"https://dummyimage.com/300/09f.png"

demo = gr.Interface(
    generate_fake_image,
    inputs=["textbox"],
    outputs=["textbox", "image"],
    additional_inputs=[
        gr.Slider(0, 1000),
        "image"
    ]
)

demo.launch()
```