Funded by
the European Union

ASSISTANT

Evaldas Vaičiukynas (evaldas.vaiciukynas@ktu.lt)

https://www.assistant-erasmus.eu

Dashboard with Python

A diagram of the steps to share a machine learning model using Gradio.

(a) The machine learning researcher defines the input and output interface types, and launches the interface either inline or in a new browser tab.

a **Introducing Gradio**

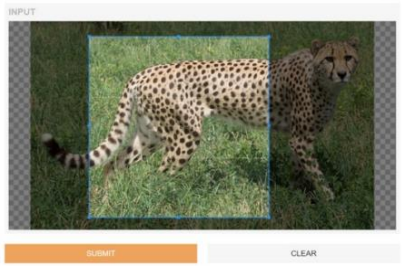
```
import tensorflow as tf
import gradio

mdl = tf.keras.models.Sequential()
# ... define and train the model as you would normally

io = gradio.Interface(inputs="imageupload",
    outputs="label", model_type="keras", model=mdl)
io.launch()
```

b

(b) The interfaces launches, and optionally, a public link is created that allows remote collaborators to input their own data into the model.

(c) The users of the interface can also manipulate the model in natural ways, such as cropping images or obscuring parts of the image.

c



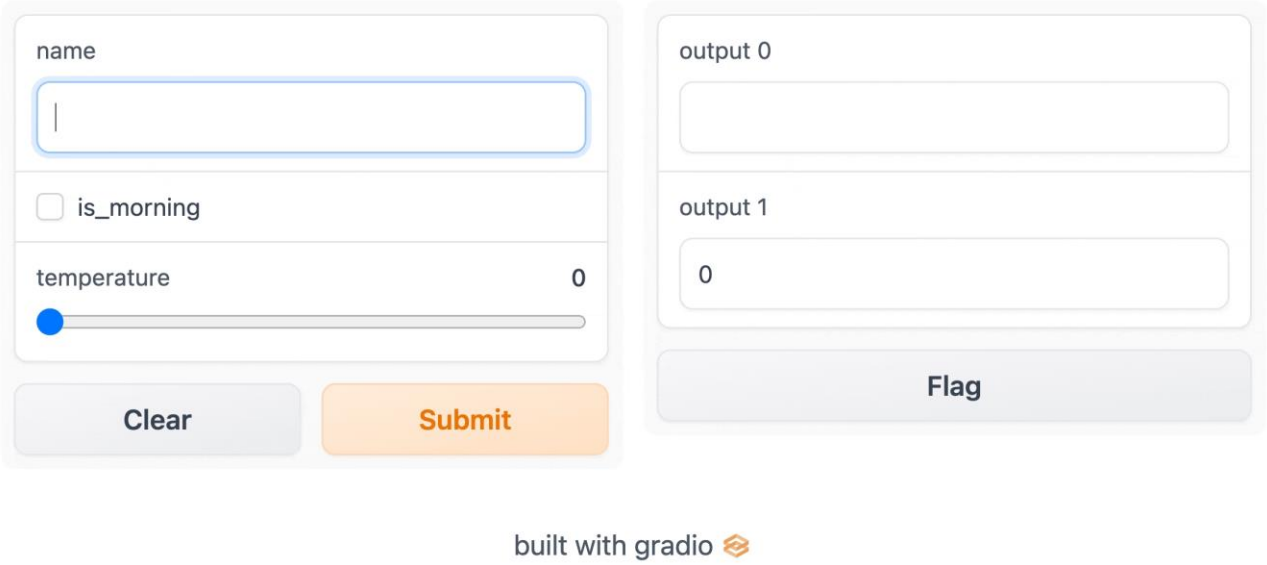
d

(d) All of the model computation is done by the host (i.e. the computer that called Gradio).

Abid, A., Abdalla, A., Abid, A., Khan, D., Alfozan, A., & Zou, J. (2019). Gradio: Hassle-free sharing and testing of ML models in the wild. arXiv preprint arXiv:1906.02569. https://doi.org/10.48550/arXiv.1906.02569

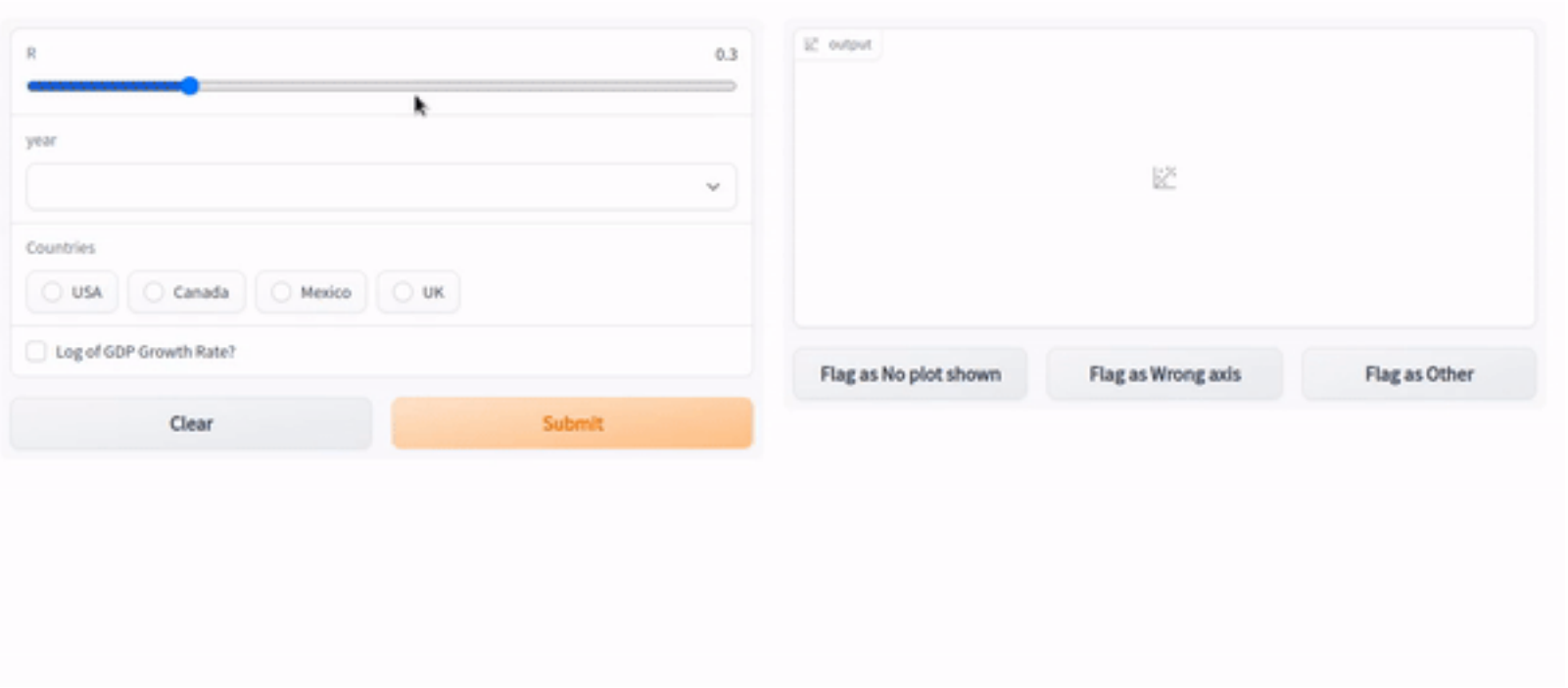
Gradio Interface: example with multiple inputs and outputs

```
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()
```



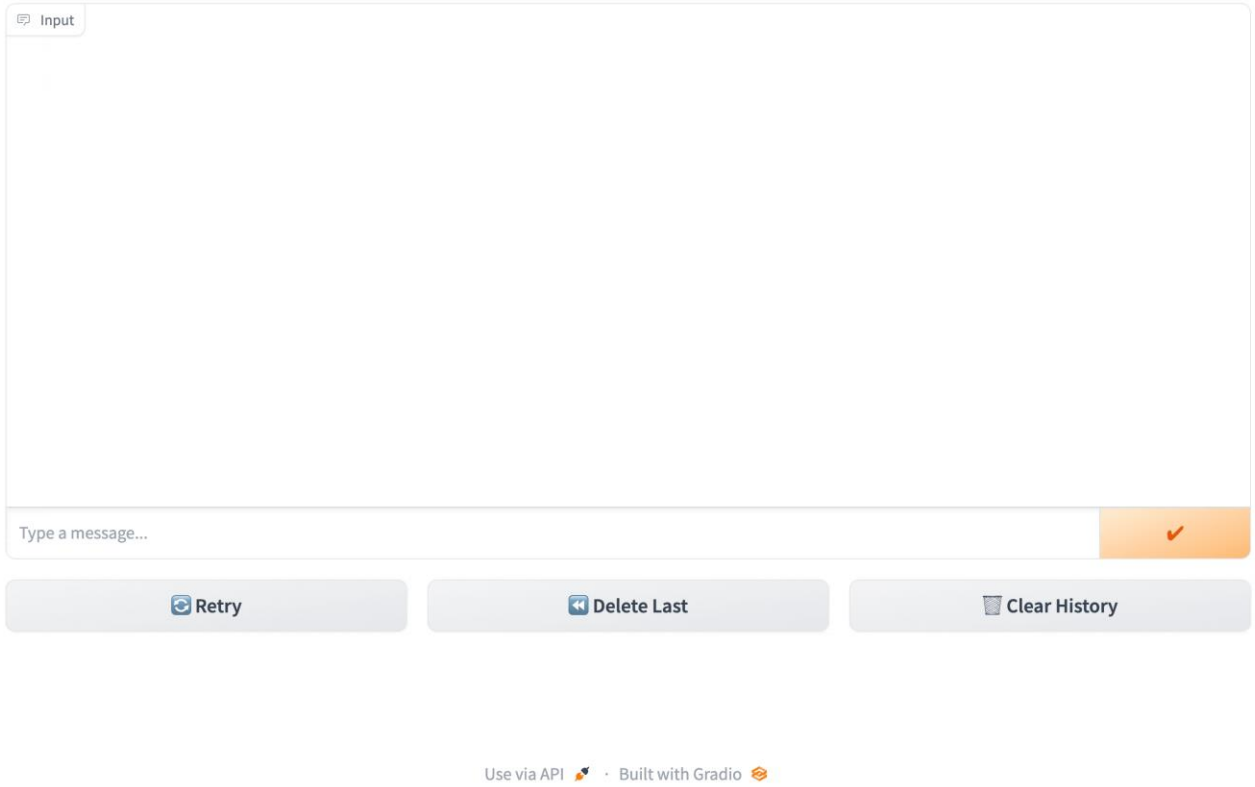
Gradio Flagging: collecting data from users of the dashboard

- Gradio simplifies the collection of data by including a **Flag** button
- Flagging feature can be used with *gradio.Interface* & *gradio.Blocks*



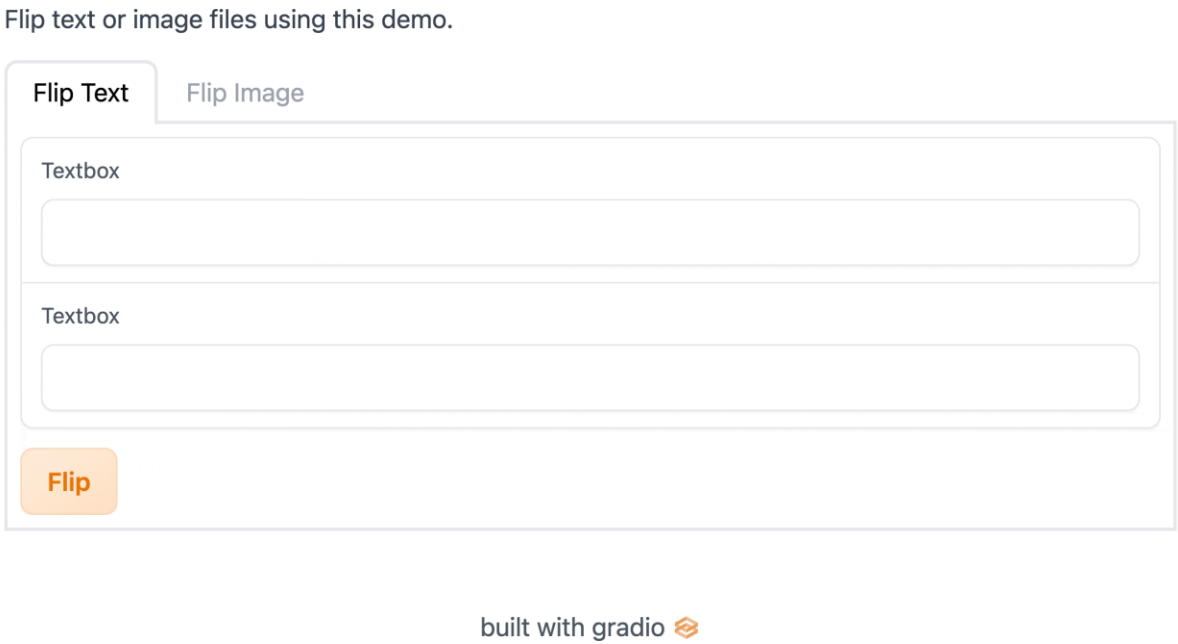
Gradio ChatInterface: wrapping function has message (str) & history (list)

```
import random
import gradio as gr

def random_response(message, history):
    return random.choice(["Yes", "No"])

demo = gr.ChatInterface(random_response)

demo.launch()
```



Gradio Blocks: more flexible layouts and data flows

```
def flip_text(x):
    return x[::-1]

def flip_image(x):
    return np.fliplr(x)

with gr.Blocks() as demo:
    gr.Markdown("Flip text or image files using
    with gr.Tab("Flip Text"):
        text_input = gr.Textbox()
        text_output = gr.Textbox()
        text_button = gr.Button("Flip")
    with gr.Tab("Flip Image"):
        with gr.Row():
            image_input = gr.Image()
            image_output = gr.Image()
        image_button = gr.Button("Flip")

    with gr.Accordion("Open for More!"):
        gr.Markdown("Look at me...")

    text_button.click(flip_text, inputs=text_input, outputs=text_output)
    image_button.click(flip_image, inputs=image_input, outputs=image_output)

demo.launch()
```



Gradio component for diagrams: gradio.Plot(⋯)

Description

Used to display various kinds of plots (matplotlib, plotly, or bokeh are supported).

Behavior

As input: this component does not accept input.

As output: expects either a matplotlib.figure.Figure, a plotly.graph_objects._figure.Figure, or a dict corresponding to a bokeh plot (json_item format)

ASSIST Air Data: function, returning a specific plot for a selected city/years

```
import gradio as gr
import plotly.express as px

# Function for Gradio output plots
def make_plot(myCity, myYear, myPlot):
    if len(myYear)==0:
      myYear = myYears
    myTable = dataTableTHI[dataTableTHI['City']==myCity][["Date", "THI"]]
    myTable = myTable[pd.to_datetime(myTable['Date']).dt.year.isin(myYear)]
    if myPlot == "calplot":
      pdTimeSeries = pd.Series(myTable['THI'].values, index=pd.DatetimeIndex(myTable['Date']))
      logging.getLogger('matplotlib.font_manager').disabled = True
      cp = calplot.calplot(pdTimeSeries, dropzero=True, cmap='coolwarm', yearlabel_kws={'color': 'black', 'fontsize':9})
      plot_result = cp[0]
    elif myPlot == "by month":
      myTable['Month'] = pd.to_datetime(myTable['Date']).dt.month
      plot_result = px.box(myTable, x="Month", y="THI")
    elif myPlot == "by weekday":
      myTable['DayOfWeek'] = pd.to_datetime(myTable['Date']).dt.dayofweek + 1
      plot_result = px.box(myTable, x="DayOfWeek", y="THI")
    else:
      plot_result = px.line(myTable, x='Date', y='THI')
      plot_result.update_xaxes(rangeslider_visible=True, rangeselector=dict(buttons=list(tsDrill)))
    return plot_result
```

ASSIST Air Data: using Gradio Blocks, Dropdown, Checkbox, Radio, Plot

```
# Get unique cities and years
myCities = dataTableTHI.City.unique().tolist()
myYears = pd.to_datetime(dataTableTHI['Date']).dt.year.unique().tolist()

# Set time series plot buttons
tsDrill = [dict(count=1, label="1m", step="month", stepmode="backward"),
           dict(count=6, label="6m", step="month", stepmode="backward"),
           dict(count=1, label="YTD", step="year", stepmode="todate"),
           dict(count=1, label="1y", step="year", stepmode="backward"),
           dict(step="all")]

# Design of Gradio dashboard
with gr.Blocks() as demo:
    citySelect = gr.Dropdown(myCities, label="City:", value="Bucharest")
    yearSelect = gr.CheckboxGroup(myYears, label="Years:", value=myYears)
    plotSelect = gr.Radio(label="Plot type:", choices=['calplot', 'by month', 'by weekday', 'time series'], value='calplot')
    plotVisual = gr.Plot(show_label=False, container=False)
    citySelect.change(make_plot, inputs=[citySelect, yearSelect, plotSelect], outputs=[plotVisual])
    yearSelect.change(make_plot, inputs=[citySelect, yearSelect, plotSelect], outputs=[plotVisual])
    plotSelect.change(make_plot, inputs=[citySelect, yearSelect, plotSelect], outputs=[plotVisual])
    demo.load(make_plot, inputs=[citySelect, yearSelect, plotSelect], outputs=[plotVisual])

if __name__ == "__main__":
    demo.launch(debug=True)
```

ASSIST Air Data: THI dashboard for Bulgaria

Temperature-Humidity Index in Bulgaria

THI = 0.8 * Temperature + Humidity * (Temperature - 14.4) + 46.4

City:

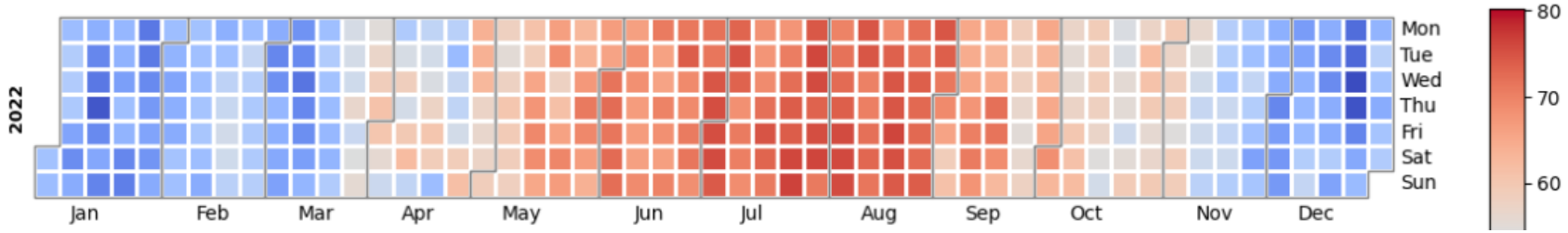
Bucharest

Years:

2022 2023

Plot type:

calplot by month by weekday time series



Thank you for attention

Funded by
the European Union