Python Tools to Deploy Your Machine Learning Models Faster 🛩️

Jeff Hale

gradio

Streamlit

FastAPI

Preflight check

See questions in Slack 🙂

Planes

gradio

Streamlit

FastAPI

Flight plans

- Test flights for each plane
    - Hello world
    - Show data
    - Plot
    - Predict
    - Cruising altitude & turbulence (pros & cons)
- Grab your luggage (takeaways)
- Post-flight data (Q&A and questionnaire)
- Disembark

Flight plan materials

- GitHub repo: https://github.com/discdiver/dsdc-deploy-models

Ultralight

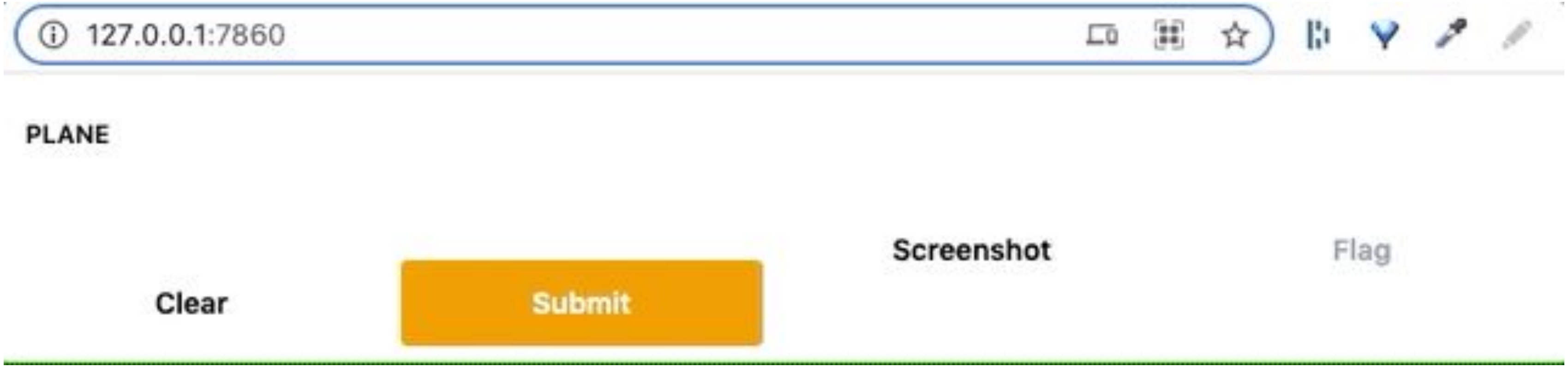
New, quick to fly, experimental

gradio

Gradio Demo 1: Hello world

Gradio #1: Hello world

- *pip install gradio*
- *python gradio_hello.py*



Gradio #1: Hello world

```python
import gradio as gr

def hello(plane):
    return f"I'm an ultralight {plane} 🛩"

iface = gr.Interface(
    fn=hello,
    inputs=['text'],
    outputs=['text']
).launch()
```

gradio

Gradio Demo 2: Show data

Gradio #2: Show me the data!

```python
def show_pens(alpha):
    return pd.read_csv(
     'https://raw.githubusercontent…/penguins.csv')

iface = gr.Interface(
    fn=show_pens,
    inputs=['text'],
    outputs=[gr.outputs.Dataframe()],
    description="Table of Palmer Penguins"
).launch(share=True)
```

Gradio #2: Show me the data!

- *pip install gradio pandas seaborn*
- *python gradio_pandas.py*

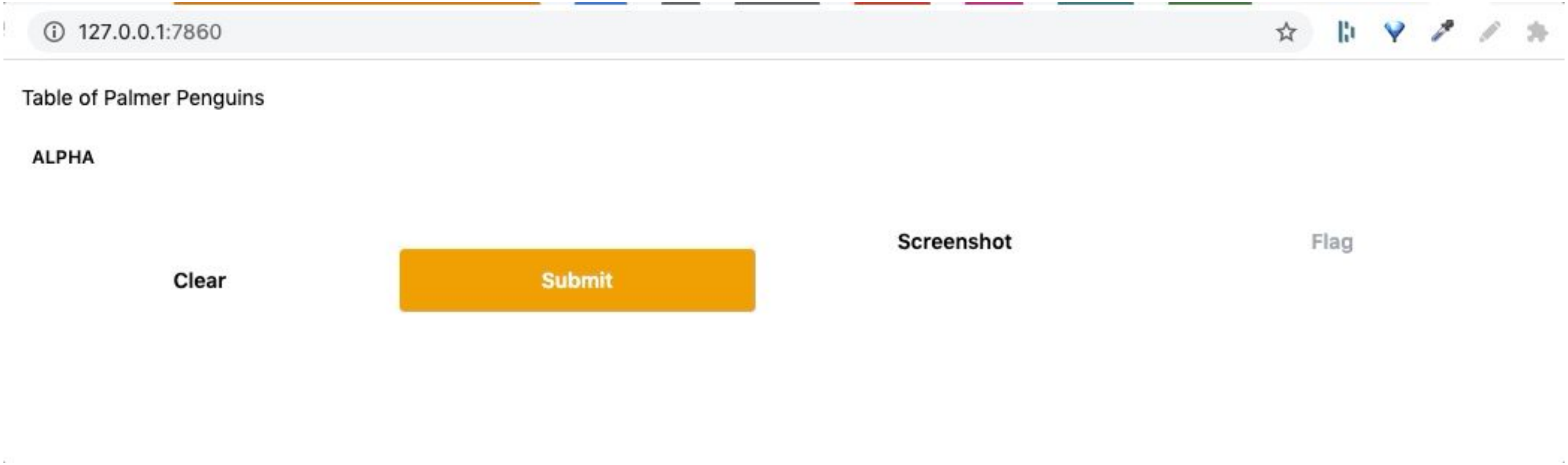


Table of Palmer Penguins

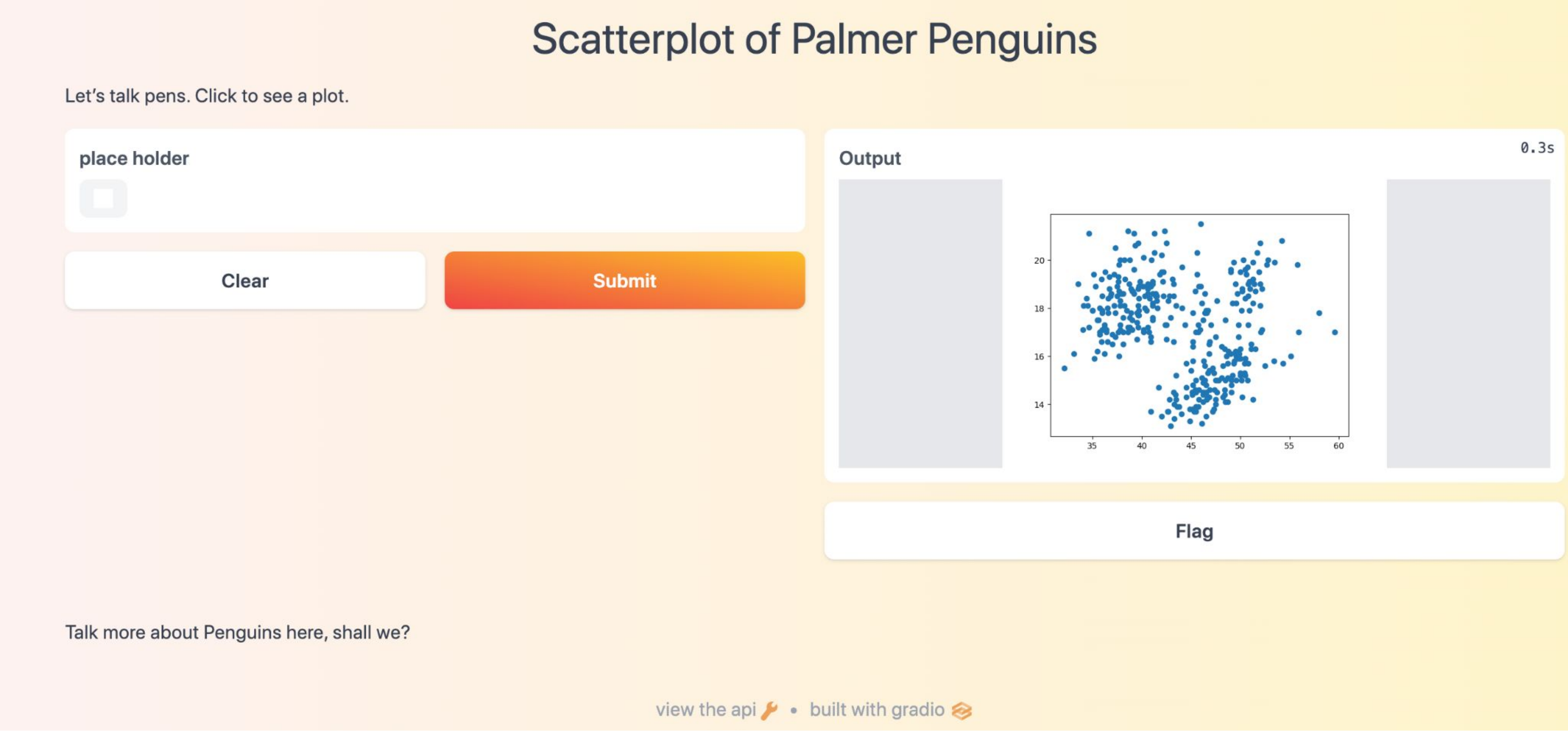
ALPHA

gradio

Gradio Demo 3: Plotting

Gradio #3: Plot it

- Plotly doesn’t work as of 2.8.7 (targeted for 2.9) ☹️
- You can use Matplotlib as of Gradio 2.8.2



Scatterplot of Palmer Penguins

Let’s talk pens. Click to see a plot.

Talk more about Penguins here, shall we?

Gradio #3: Plot it

```python
def plot_pens(place_holder):
    """scatter plot penguin chars using matplotlib"""

    df_pens = pd.read_csv("https://raw.githubuser…/penguins.csv")

    fig = plt.figure()
    plt.scatter(
        x=df_pens["bill_length_mm"], y=df_pens["bill_depth_mm"])

    return fig
```

Gradio #3: Plot it

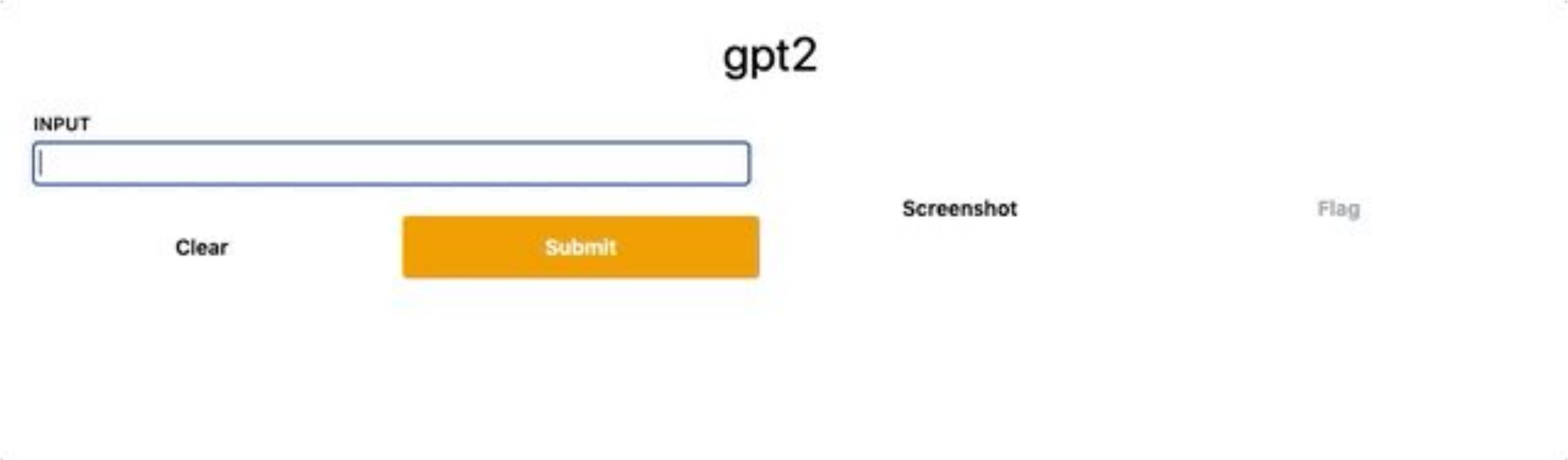
```
iface = gr.Interface(
    fn=plot_pens,
    layout="vertical",
    inputs=["checkbox"],
    outputs=["plot"],
    title="Scatterplot of Palmer Penguins",
    description="Let's talk pens. Click to see a plot.",
    article="Talk more about Penguins here, shall we?",
    theme="peach",
    live=True,
).launch()
```

gradio

Gradio Demo 4: Predict

Gradio #4: Model inference

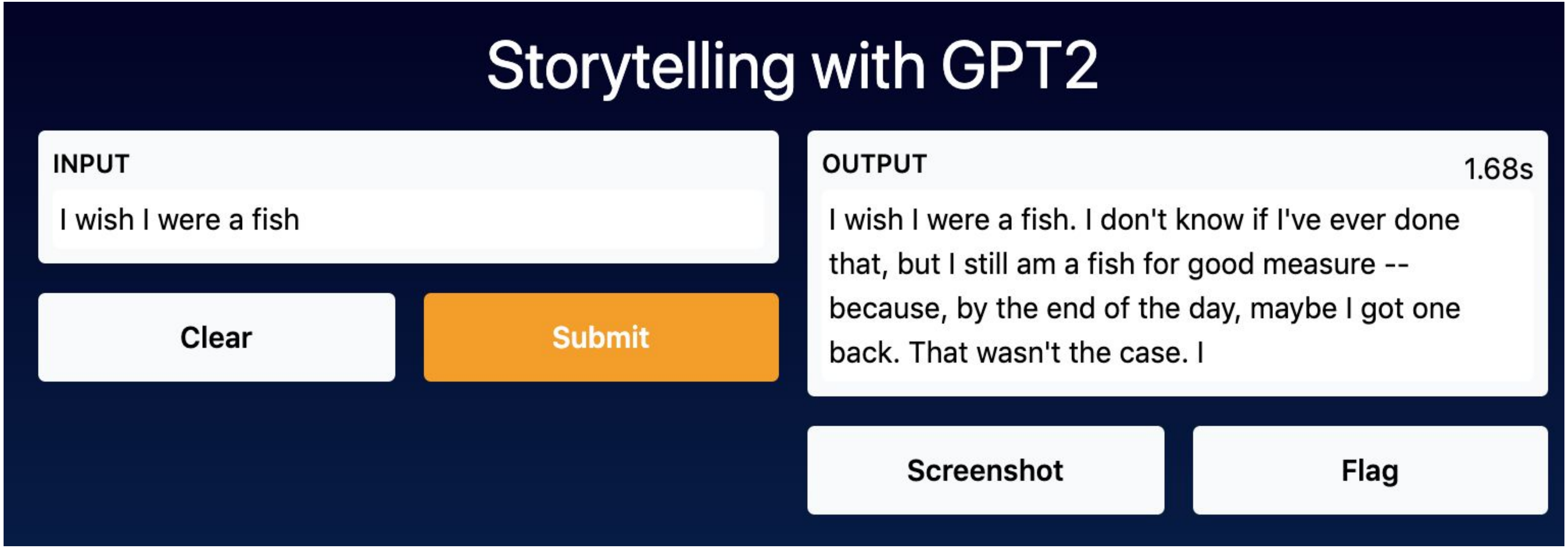
```
import gradio as gr

gr.Interface.load('huggingface/gpt2').launch()
```



Gradio #4: Predict - prettier

```
gr.Interface.load(
   "huggingface/gpt2",
   title="Storytelling with GPT2",
   css="""
       body {background: rgb(2,0,36);
             background: linear-gradient(
             180deg,
             rgba(2,0,36,1) 0%,
             rgba(7,51,99,1) 70%,
             rgba(6,3,17,1) 100%);}
       .title {color: white !important;}
       """,
).launch()
```

Gradio #4: Predict - prettier



Storytelling with GPT2

INPUT

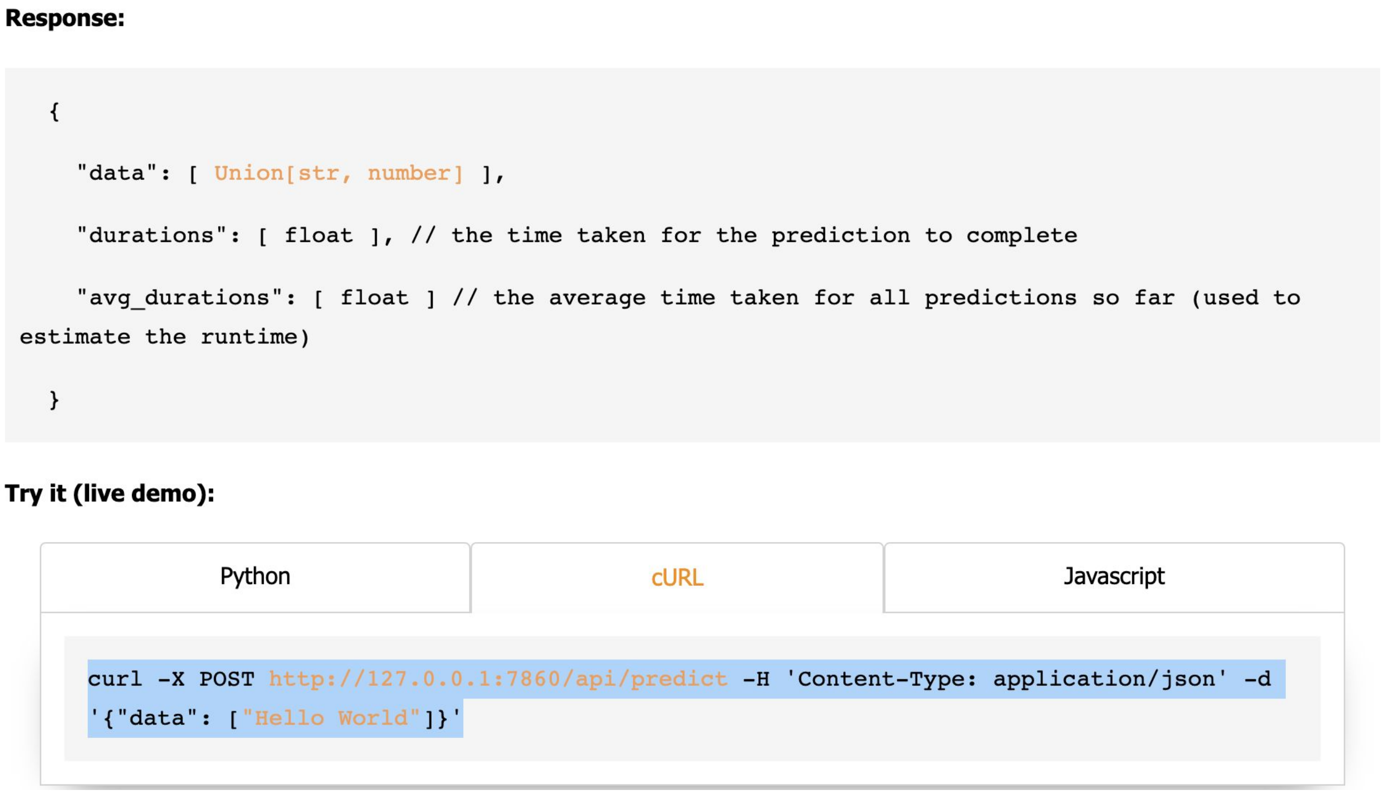
I wish I were a fish

OUTPUT

1.68s

I wish I were a fish. I don't know if I've ever done that, but I still am a fish for good measure -- because, by the end of the day, maybe I got one back. That wasn't the case. I

Gradio Data API - One Click!



Response:

```
{
  "data": [ Union[str, number] ],
  "durations": [ float ], // the time taken for the prediction to complete
  "avg_durations": [ float ] // the average time taken for all predictions so far (used to estimate the runtime)
}
```

Try it (live demo):

```
curl -X POST http://127.0.0.1:7860/api/predict -H 'Content-Type: application/json' -d '{"data": ["Hello World"]}'
```

Gradio Pros

- Quick demos for ML 🚀
- Built-in interpretability (sometimes) 🔎
- Auto-docs 📃
- Nice Hugging Face model integration 🤗
- Bright future 😎
- Easy hosting at Hugging Face spaces

  https://huggingface.co/spaces/discdiver/gpt2

Gradio Cons

- Rough around the edges - early stage 🔪
- Not easy to customize elements ✨
- Single page only 1️⃣

Gradio - Hugging Face Acquisition

Abubakar Abid + 🤗

Streamlit

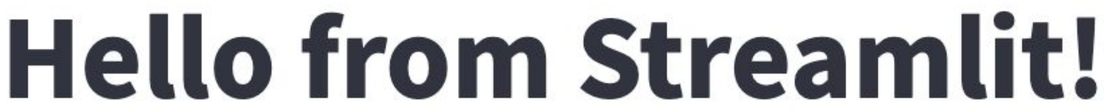
Cessna Citation Longitude

Light, quick to takeoff, easy flying

Streamlit

Streamlit Demo 1: Hello world

Streamlit #1: Hello world



Hello from Streamlit!

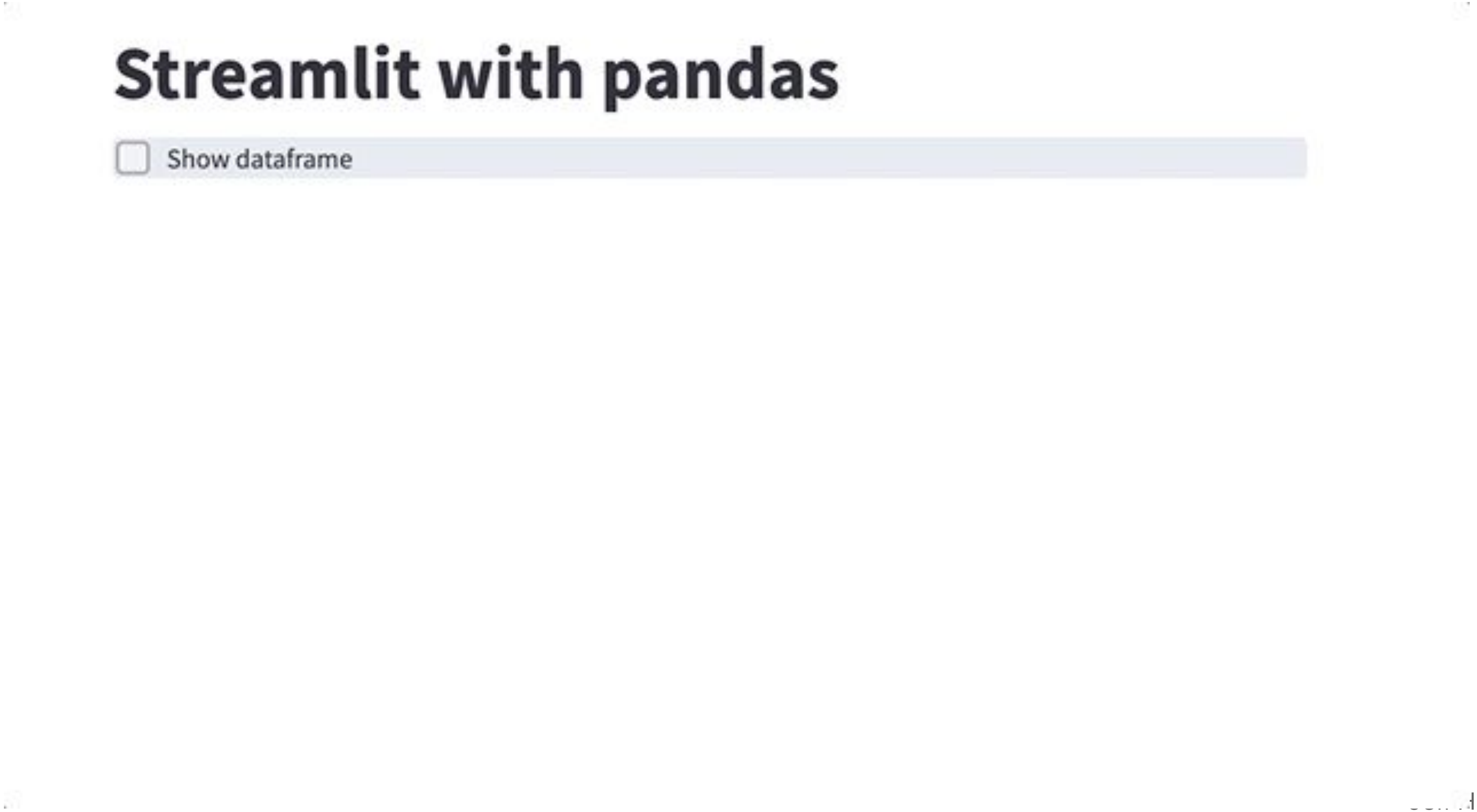
```python
import streamlit as st

name = "Jeff"

st.title(f"Hello from Streamlit, {name}!")
```

- *pip install streamlit*
- *streamlit run streamlit_hello.py*

Streamlit

Streamlit Demo 2: Show data

Streamlit #2: Show data



Streamlit with pandas

☐ Show dataframe

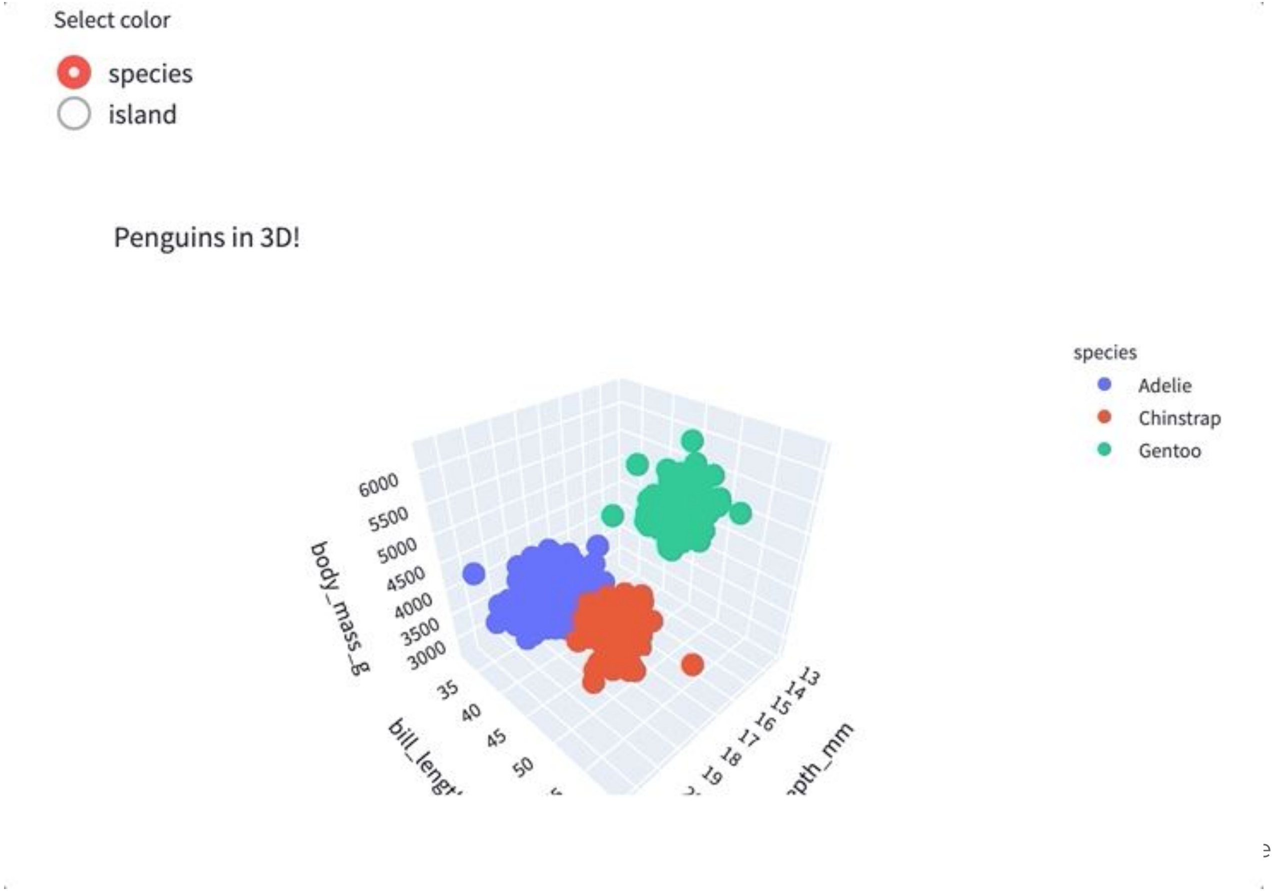
Streamlit #2: Show data

```
import streamlit as st
import pandas as pd

st.title("Streamlit with pandas")
show = st.checkbox("Show dataframe")
df_pens = pd.read_csv( "https://raw.githubusercontent…/penguins.csv")

if show:
    df_pens
```

Streamlit

Streamlit Demo 3: Plotting

Streamlit #3: Plotting



Select color

● species
○ island

Penguins in 3D!

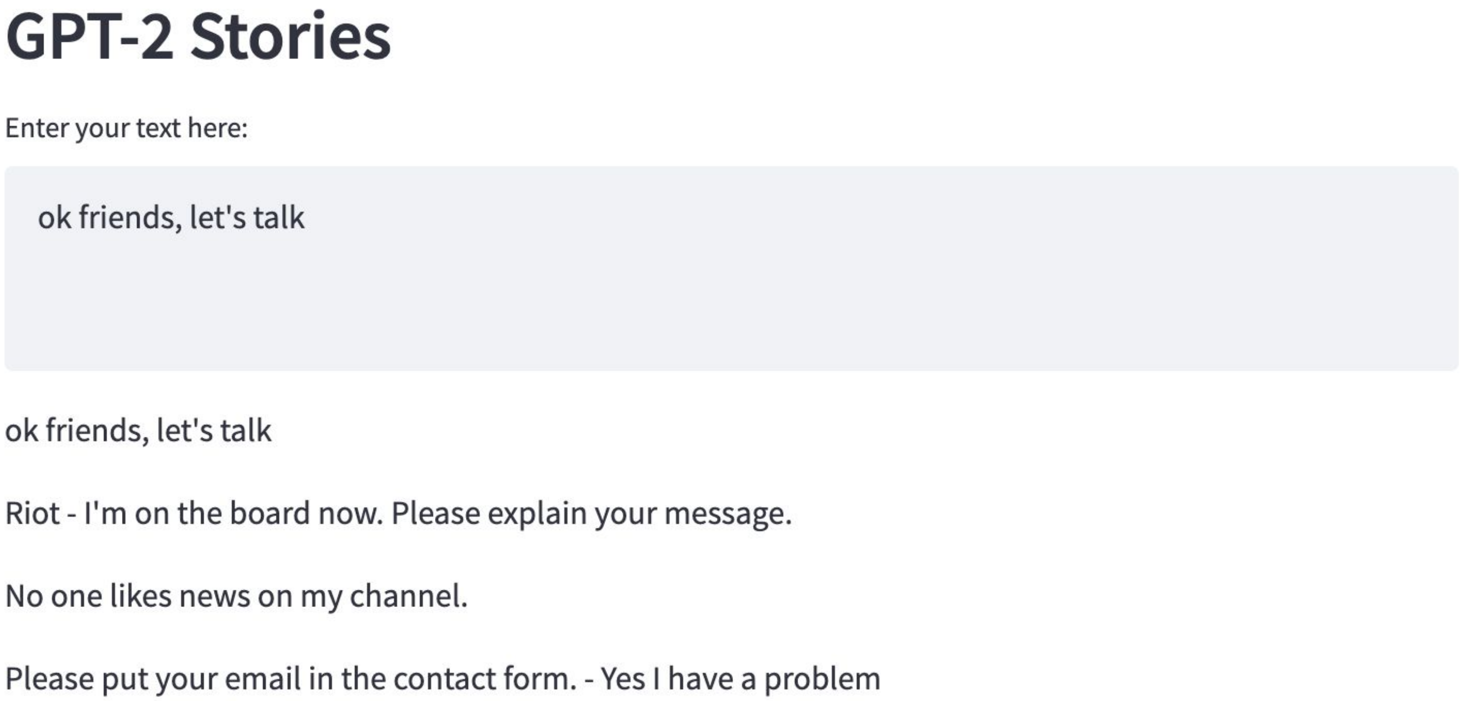
Streamlit #3: Plotting

```
choice = st.radio("Select color", ["species", "island"])

fig = px.scatter_3d(
    data_frame=df_pens,
    x="bill_depth_mm",
    y="bill_length_mm",
    z="body_mass_g",
    color=choice,
    title="Penguins in 3D!",
)
fig
```

Streamlit

Streamlit Demo 4: Predict

Streamlit #4: Predict



GPT-2 Stories

Enter your text here:

ok friends, let's talk

ok friends, let's talk

Riot - I'm on the board now. Please explain your message.

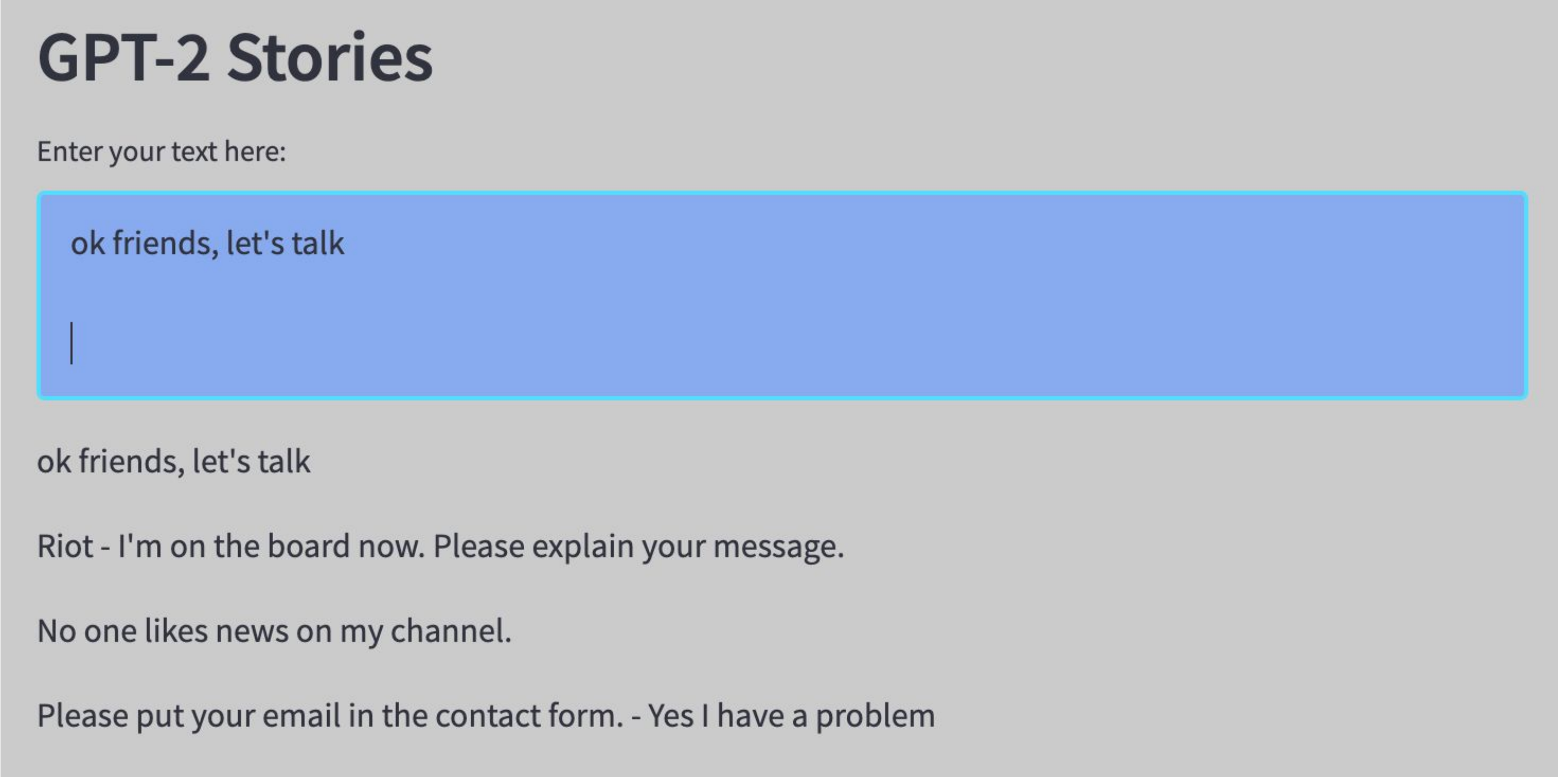
No one likes news on my channel.

Please put your email in the contact form. - Yes I have a problem

Streamlit #4: Predict

```python
import streamlit as st
from transformers import pipeline


st.header("GPT-2 Stories")
input_text = st.text_area("Enter your text here:")


generator = pipeline("text-generation", model="gpt2")
output = generator(input_text, max_length=100)
output[0]["generated_text"]
```

Streamlit #4: Predict



GPT-2 Stories

Enter your text here:

ok friends, let's talk

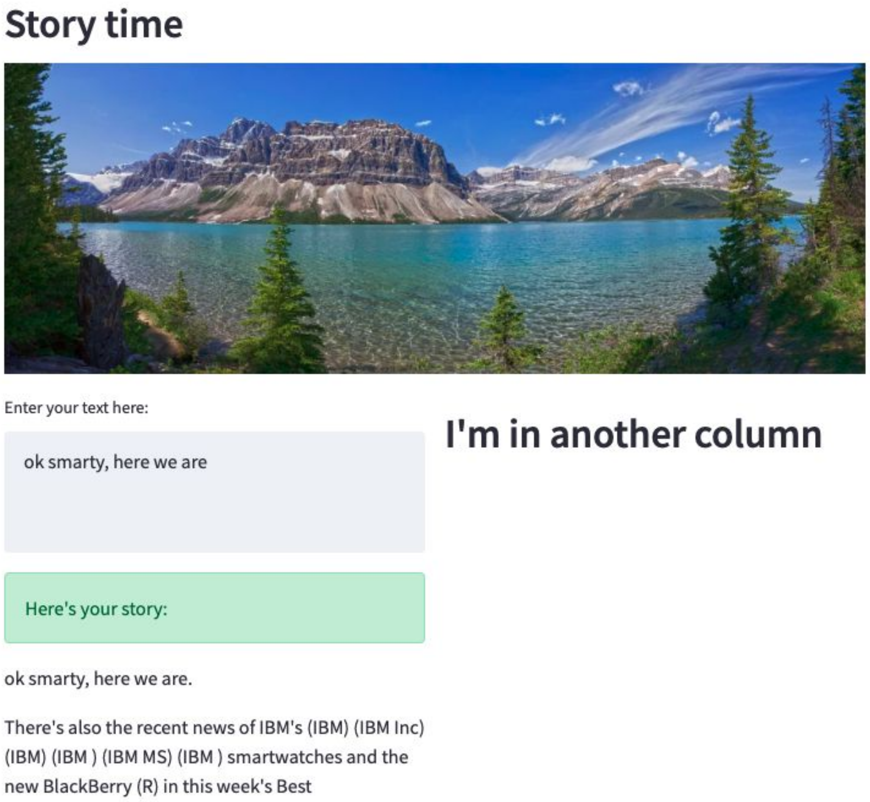
ok friends, let's talk

Riot - I'm on the board now. Please explain your message.

No one likes news on my channel.

Please put your email in the contact form. - Yes I have a problem

Streamlit #4: Predict - prettier



Story time

Enter your text here:

ok smarty, here we are

Here's your story:

ok smarty, here we are.

There's also the recent news of IBM's (IBM) (IBM Inc) (IBM) (IBM) (IBM MS) (IBM) smartwatches and the new BlackBerry (R) in this week's Best

I'm in another column

Streamlit #4: Predict - prettier

```
st.header("Story time")
st.image("https://cdn.pixabay.com/photo/2017/07/12/19/03/highway-2497900_960_720.jpg")
col1, col2 = st.columns(2)

with col1:
    input_text = st.text_area("Enter your text here:")
    with st.spinner("Generating story..."):
        generator = pipeline("text-generation", model="gpt2")
        if input_text:
            generated_text = generator(input_text, max_length=60)
            st.success("Here's your story:")
            generated_text[0]["generated_text"]

with col2:
    st.header("I'm in another column")
```

Streamlit Serving Options

- Serve from Streamlit's servers for free: bit.ly/st-6plots
- Serve from Hugging Face Spaces or Heroku for free 🤗
- Pay Streamlit for more/better hosting 💰
- Host elsewhere

Streamlit Pros

- Quick websites for many Python use cases 🐍
- Many intuitive interactive widgets ✅
- Caching ⏱️
- Nice hosting options 🎁
- Thoughtful docs 📄
- Strong development cadence & team 💪

Streamlit Cons

- Some customizations cumbersome ✨
- Single page only (without hacks)1️⃣

Streamlit Snowflake Acquisition

Recent acquisition by Snowflake

Image Source: Snowflake, Public domain, via Wikimedia Commons

FastAPI

Boeing 737

Commercial grade, fast, smart!

Jeff Hitchcock, CC BY 2.0 <https://creativecommons.org/licenses/by/2.0>, via Wikimedia Commons

FastAPI

FastAPI Demo 1: Hello world

FastAPI #1: Hello world

```
import uvicorn
from fastapi import FastAPI


app = FastAPI()


@app.get("/")
def home():
    return {"Hello world": "How's it going?"}


if __name__ == "__main__":
    uvicorn.run("fastapi_hello:app")
```

FastAPI #1: Hello world

pip install fastapi uvicorn

python fastapi_hello.py

Returns json

{"Hello world":"How's it going ?"}

FastAPI

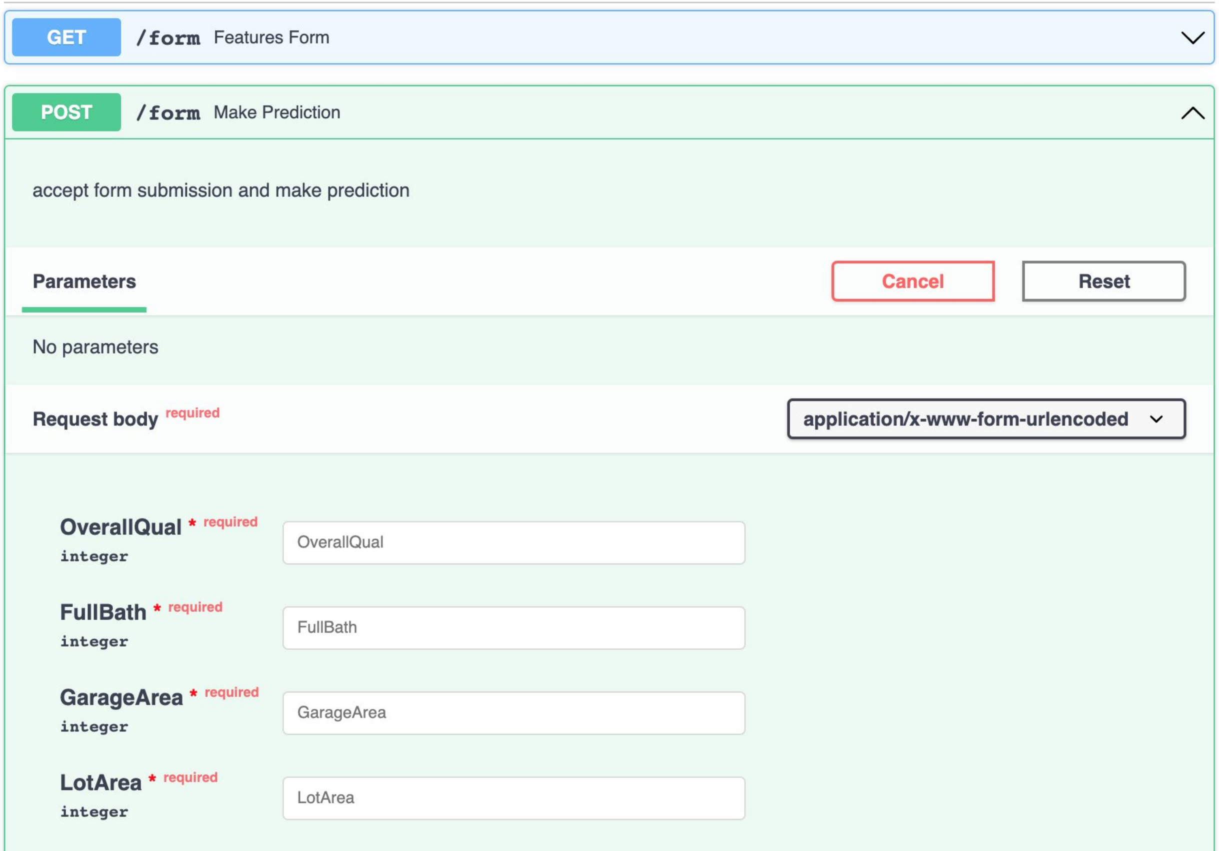
https://giphy.com/gifs/maverick-the-need-for-speed-i-feel-26vUzl3ufQxx9dims

FastAPI Speed

- Uses Starlette - ASGI (*Asynchronous Server Gateway Interface*)
- Fastest Python framework - Techempower benchmark

FastAPI Demo 2: Show data

Automatic docs



/form Features Form

/form Make Prediction

accept form submission and make prediction

No parameters

Request body required

application/x-www-form-urlencoded

OverallQual * required
integer
OverallQual

FullBath * required
integer
FullBath

GarageArea * required
integer
GarageArea

LotArea * required
integer
LotArea

FastAPI #2: Show me the data!

```python
@app.get("/df")
async def pens_data():
    df_pens = pd.read_csv(
    "https://raw.githubuser…/penguins.csv")
    df_no_nans = df_pens.fillna(-1.01)
    return df_no_nans
```

FastAPI #2: Show me the data!

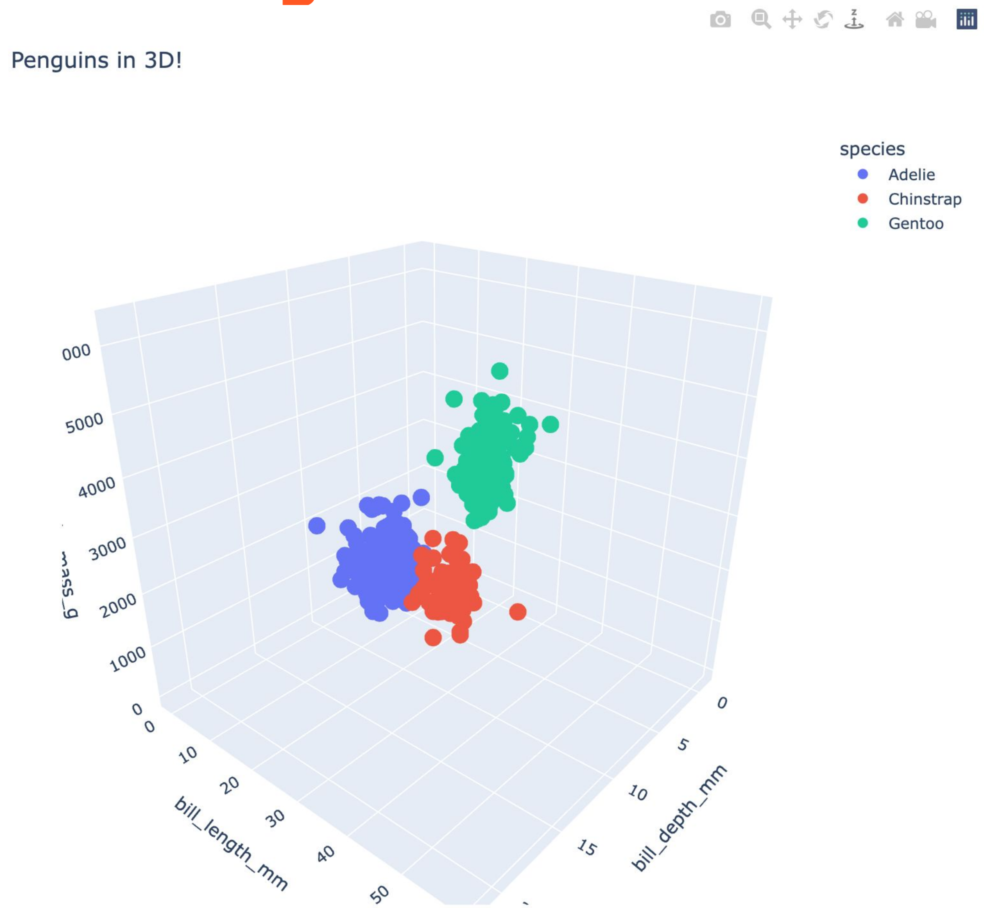
Return DataFrame, get back JSON 😎

{"species":{"0":"Adelie","1":"Adelie"...

FastAPI

FastAPI Demo 3: Plotting

FastAPI #3: Plotting



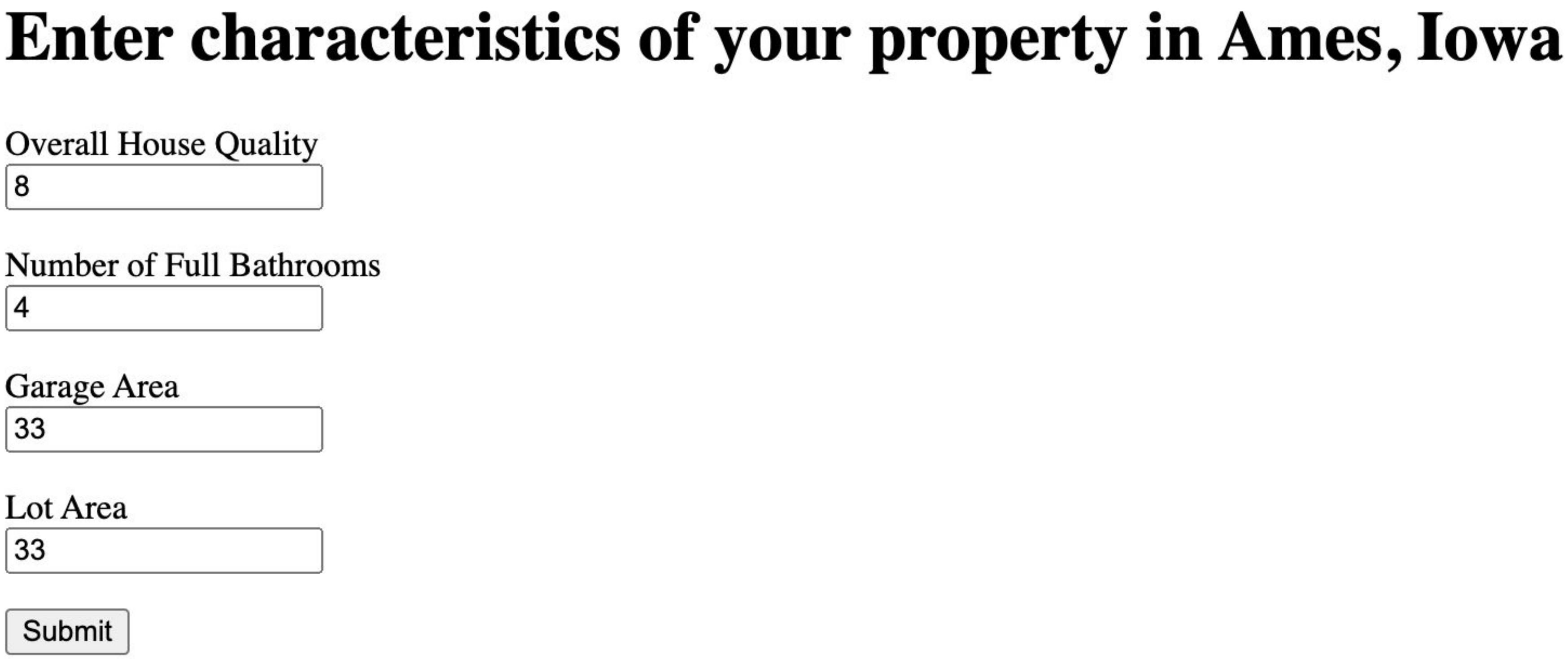
FastAPI #3: Plotting

```python
@app.get("/plot")
async def plot() -> HTMLResponse:
    """return a plotly plot"""
    fig = px.scatter_3d(
        data_frame=df,
        x="bill_depth_mm",
        y="bill_length_mm",
        z="body_mass_g",
        color="species",
        title="Penguins in 3D!",
    )
    return HTMLResponse(fig.to_html())
```

FastAPI

FastAPI Demo 4: Predict

FastAPI #4: Predict: Form



Enter characteristics of your property in Ames, Iowa

Overall House Quality
8

Number of Full Bathrooms
4

Garage Area
33

Lot Area
33

FastAPI #4: Predict: Form

```
app = FastAPI()


templates = Jinja2Templates(directory="templates")


@app.get("/form", response_class=HTMLResponse)
async def features_form(request: Request):
   """form for getting data"""
   return templates.TemplateResponse(
      "form.html",
       {"request": request})
```

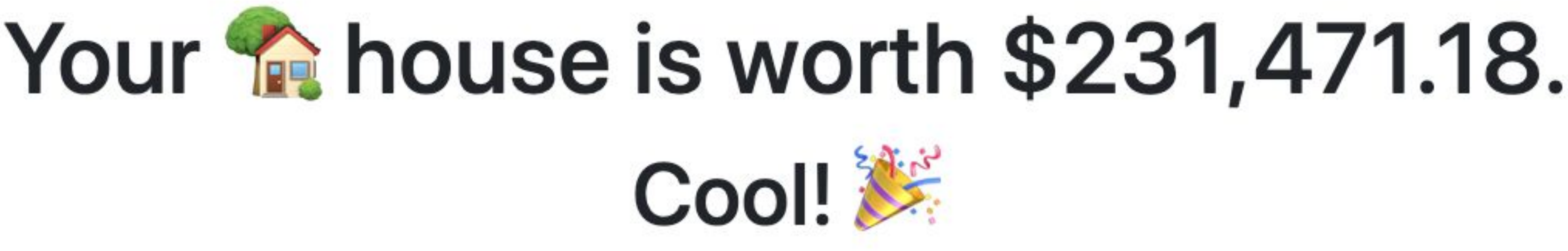
FastAPI #4: Predict: HTML Template

```
<body>
   <h1>Enter characteristics of your property in Ames, Iowa</h1>
   <form method='POST' enctype="multipart/form-data">
           <label for="OverallQual">Overall House Quality</label><br>
           <input type="number" name="OverallQual">
           <label for="FullBath">Number of Full Bathrooms</label><br>
           <input type="number" name="FullBath">
           <label for="GarageArea">Garage Area</label><br>
           <input type="number" name="GarageArea">
           <label for="LotArea">Lot Area</label><br>
           <input type="number" name="LotArea">
       <p><button type="submit" value="Submit">Submit</button></p>
   </form>
</body>
```

FastAPI #4: Predict: pydantic schema for form

```python
class FeaturesForm(BaseModel):
   """pydantic model to get the form input we want"""
   OverallQual: int
   FullBath: int
   GarageArea: int
   LotArea: int

   @classmethod
   def as_form(cls, OverallQual: int = Form(...), FullBath: int = Form(...),
       GarageArea: int = Form(...), LotArea: int = Form(...)):
       return cls(OverallQual=OverallQual, FullBath=FullBath,
           GarageArea=GarageArea, LotArea=LotArea)
```

FastAPI #4: Display prediction



Your 🏡 house is worth $231,471.18.

Cool! 🎉

FastAPI #4: Predict: process

```
@app.post("/form", response_class=HTMLResponse)
async def make_prediction(
    request: Request,
    user_input: FeaturesForm=Depends(FeaturesForm.as_form)
    ):
    """accept form submission and make prediction"""
   ...load model and make prediction

    return templates.TemplateResponse(
        "results.html", {"request": request, "prediction": pred}
    )
```

FastAPI #4: Display prediction

```
<body>
   <div class='container'>
       <div class='row'>
           <div class='col-lg text-center'>
           <h2> Your 🏡 house is worth
                {{"${:,.2f}".format(prediction) }}.<h2>
           <h3> Cool! 🎉 </h3 </div>
           </div>
       </div>
</body>
```

FastAPI Pros

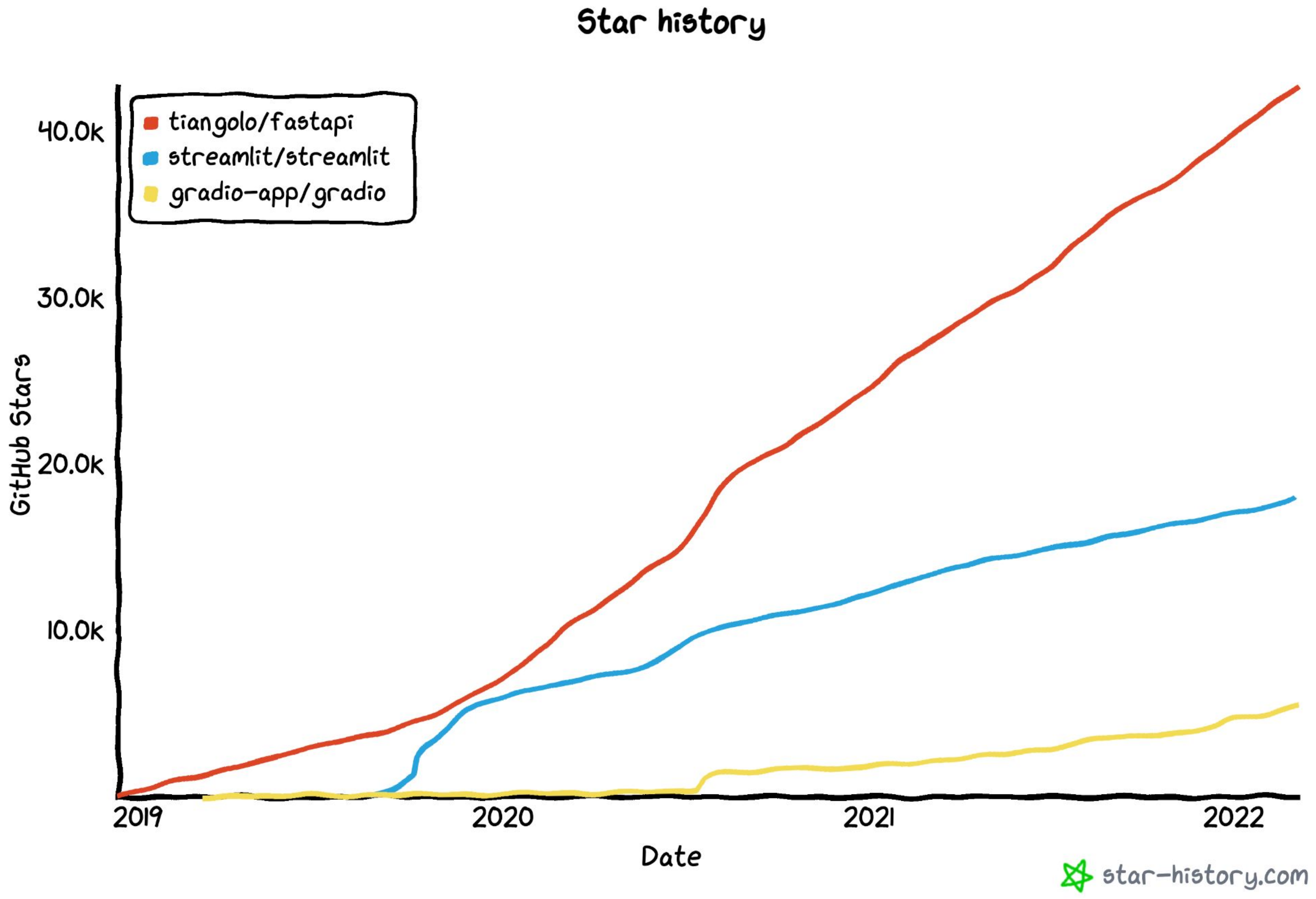
- Fastest Python API framework - async ASGI 🏎️
- Automatic API documentation ⭐
- Extensive docs 📃
- Nice test client 👌
- Nice dependency injection 💉
- Data validation with pydantic / SQL Model integration ↔️
- Security / authentication support 🔐

FastAPI Cons

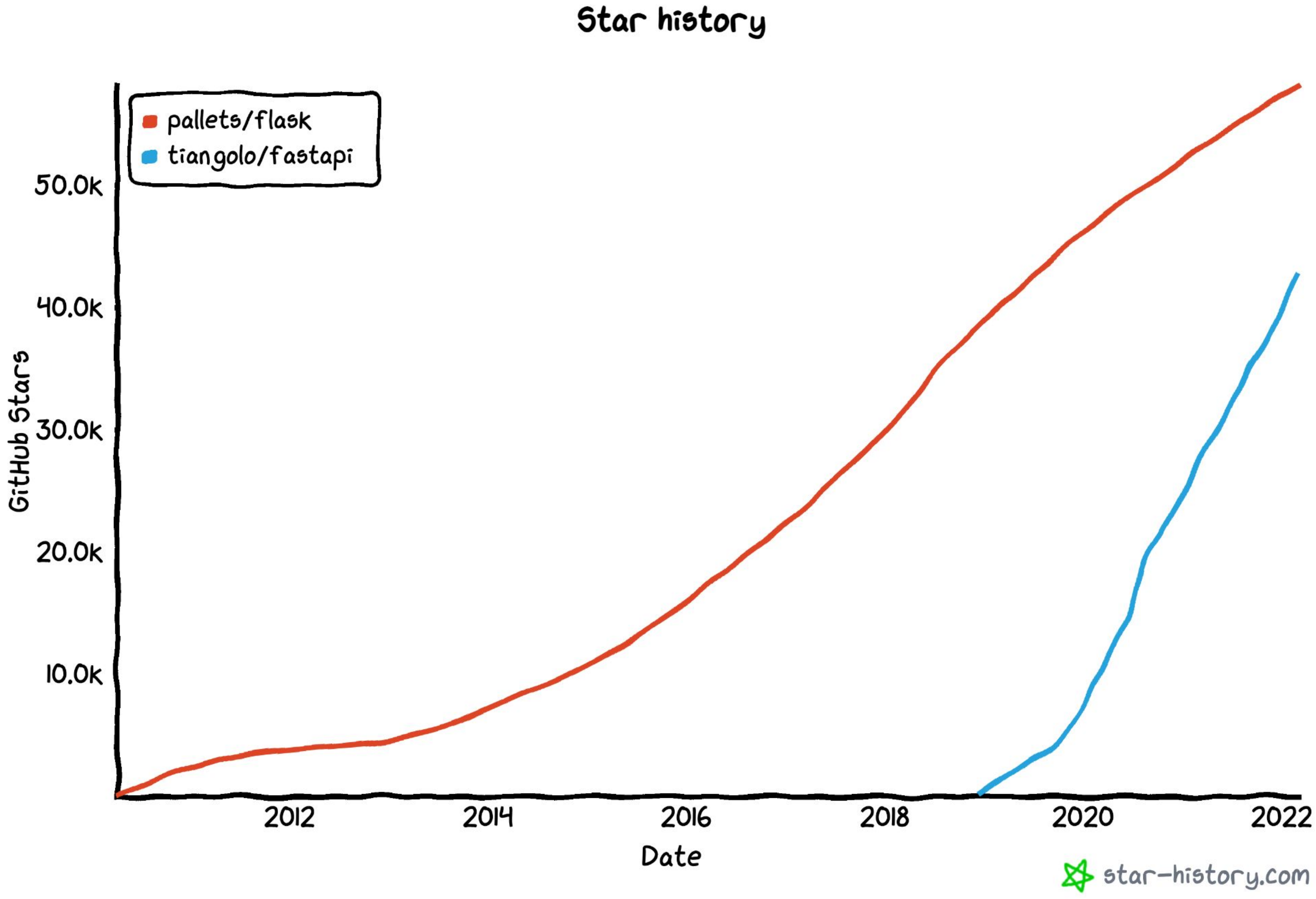
- Reliant on a single maintainer 😬
- Overriding uvicorn logging is a bit of a pain 🌳
- HTML templating more painful than Flask ⌨️

Grab your luggage (take aways)

	Web App	**Data API**
gradio	Yes	Yes
Streamlit	Yes	No
FastAPI	Yes (Jinja templates)	Yes



Source: https://star-history.t9t.io/#streamlit/streamlit&tiangolo/fastapi&gradio-app/gradio



Source: https://star-history.com/#tiangolo/fastapi&pallets/flask&Date

What about flask?

- Huge inspiration for FastAPI
- Now has more async support, but not easily full async
- FastAPI is faster
- FastAPI leverages typing
- FastAPI winning mindshare
- Gradio just switched to FastAPI backend
- Flask is nicer for quick web app

Grab Your Luggage (takeaways)

Use what you know, unless it doesn’t meet your needs

Blank slate?

Learn what’s popular, growing, and quick to get off the ground

Streamlit

For single-page app that doesn't need custom styling

gradio

Gradio for quick 🤗 models for fun

FastAPI

FastAPI for serving data