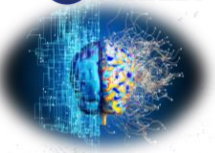
Scope and opportunities in the areas of

Artificial Intelligence - Gradio

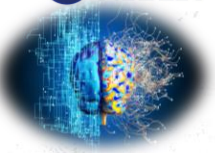
Dr. Sarwan Singh

Scientist – D, NIELIT Chandigarh



AI-Roadmap to Computer Vision

Step 1 :	Step 2 :	Step 3 :	Step 4 :	Step 5 :	Step 6 :	Step 7 :	Step 8 :	AI-Roadmap to Computer Vision
• Python & Statistics : Basics of computer vision	• Solving Image Classification : Basic Machine Learning Image pre-processing Techniques	• Introduction to Keras & Neural Networks: Fundamentals of Neural Networks (keras)	• Understanding CNNs, Transfer Learning : Convolutional Neural Networks (CNNs) Transfer learning	• Object Detection Algorithms	• Image Segmentation & Attention Models	• Deep Learning Tools : PyTorch & TensorFlow	• Generative Adversial Networks (GANs)	



Introduction

- Gradio is an open-source Python package that allows you to quickly **build a demo** or web application for your machine learning model, API, or any arbitary Python function.
- You can then **share your demo** with a a public link in seconds using Gradio's built-in sharing features.
- *No JavaScript, CSS, or web hosting experience needed!*

Gradio allows you to build, customize, and share web-based demos for any machine learning model, entirely in Python.

Gradio developed by Abubakar

- Abubakar has been building machine learning models for over a decade.
- He did his PhD at Stanford in deep learning applied to medical images and videos.
- During his PhD, he developed Gradio (www.gradio.dev), an open-source Python library for creating GUIs for machine learning models.
- Since Gradio's acquisition by Hugging Face, Abubakar continues to lead the Gradio team and also teaches machine learning at Hugging Face and beyond!
- Gradio's founding team, Abubakar "Abu" Abid, Ali Abdalla, Ali Abid, and Dawood Khan

Published @2021

abidlabs

Abubakar Abid

Gradio is joining Hugging Face! By acquiring Gradio, a machine learning startup, Hugging Face will be able to offer users, developers, and data scientists the tools needed to get to high level results and create better models and tools...

As one of the founders of Gradio, I couldn't be more excited about the next step in our journey. I still remember clearly how we started in 2019: as a PhD student at Stanford, I struggled to share a medical computer vision model with one of my collaborators, who was a doctor. I needed him to test my machine learning model, but he didn't know Python and couldn't easily run the model on his own images. I envisioned a tool that could make it super simple for machine learning engineers to build and share demos of computer vision models, which in turn would lead to better feedback and more reliable models

I recruited my talented housemates Ali Abdalla, Ali Abid, and Dawood Khan to release the first version of Gradio in 2019. We steadily expanded to cover more areas of machine learning including text, speech, and video. ……

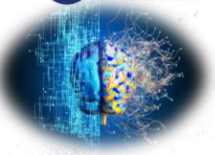
Since we first released the library, more than 300,000 demos have been built with Gradio. ……

Gradio – Helloworld

```
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()
```

Flask app – Haar cascade face detection

- requirements.txt
- opencv-python

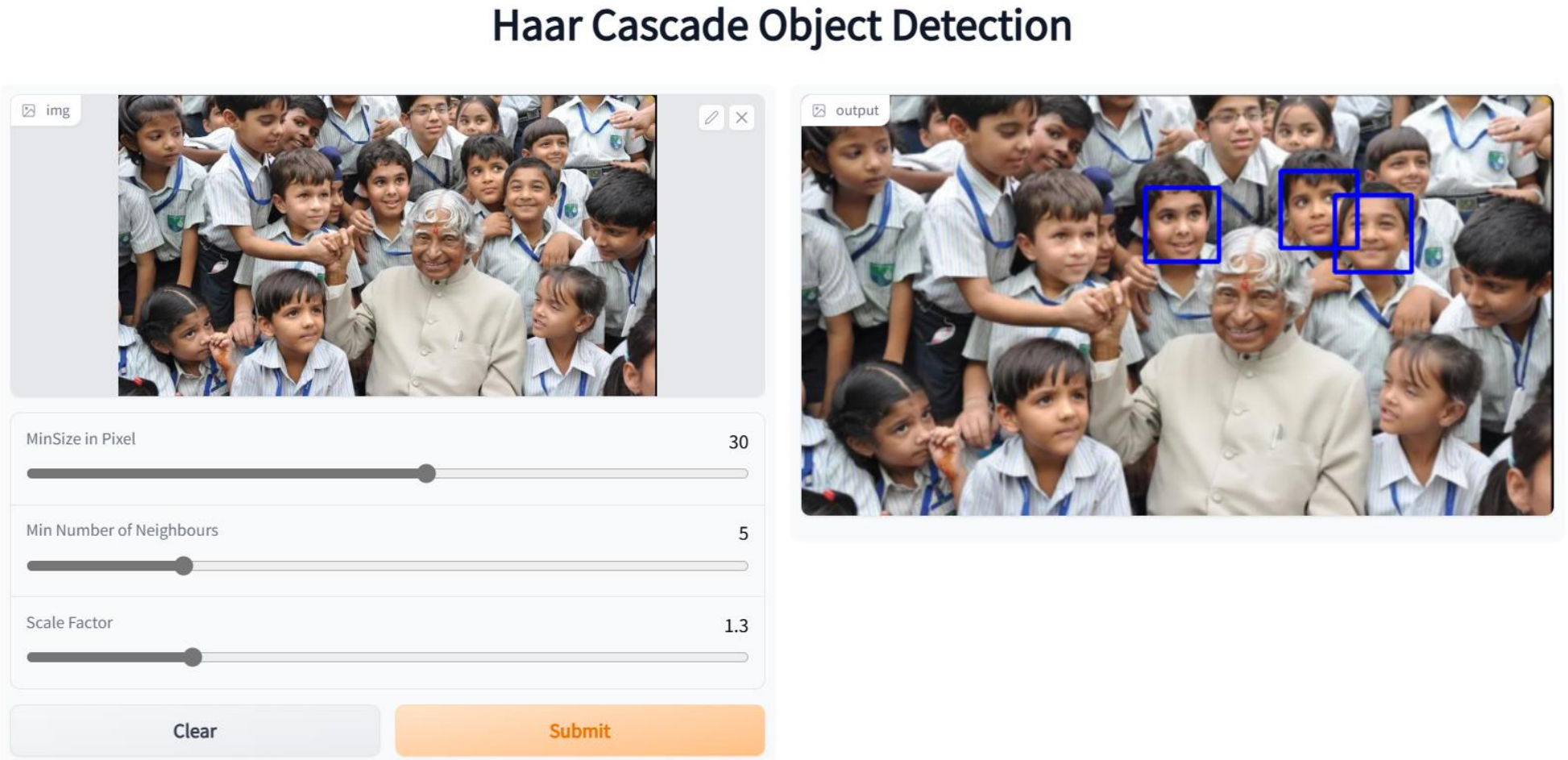


App.py

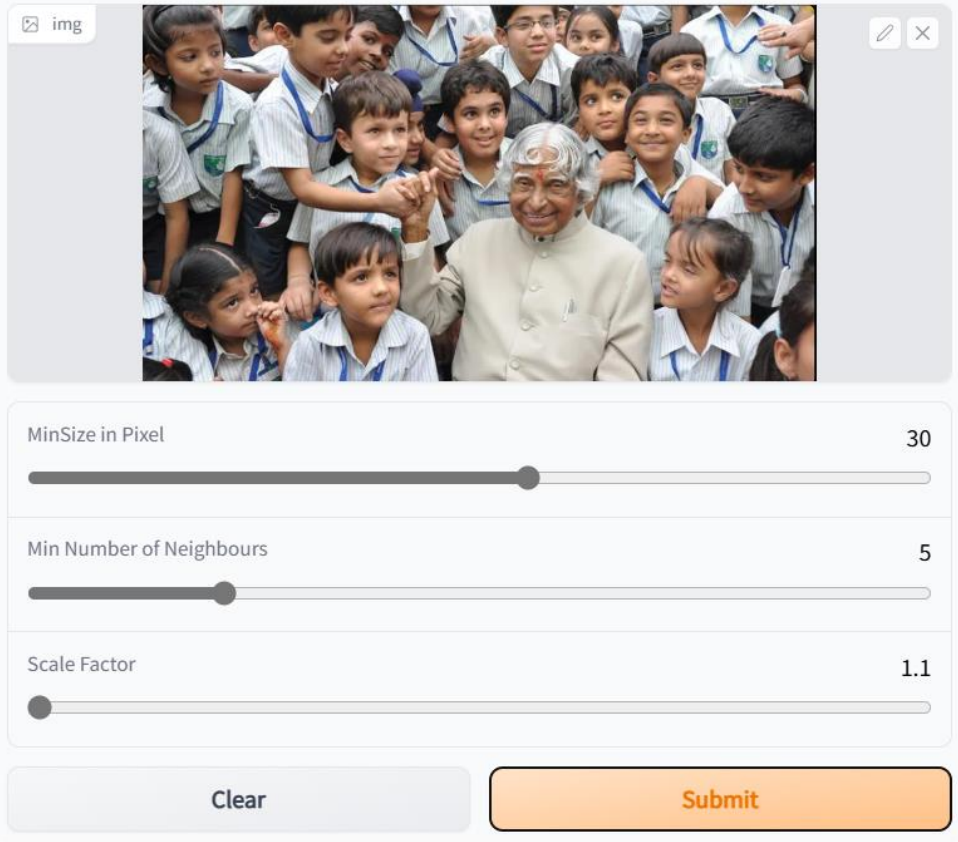
```
import numpy as np

import cv2

import gradio as gr

from PIL import Image

def detect_faces(image):

    image_np = np.array(image)

    gray_image = cv2.cvtColor(image_np, cv2.COLOR_RGB2GRAY)

    face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + "haarcascade_frontalface_default.xml")

    faces = face_cascade.detectMultiScale(gray_image, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30))

    for (x, y, w, h) in faces:

        cv2.rectangle(image_np, (x, y), (x+w, y+h), (0, 255, 0), 2)

    return image_np
```

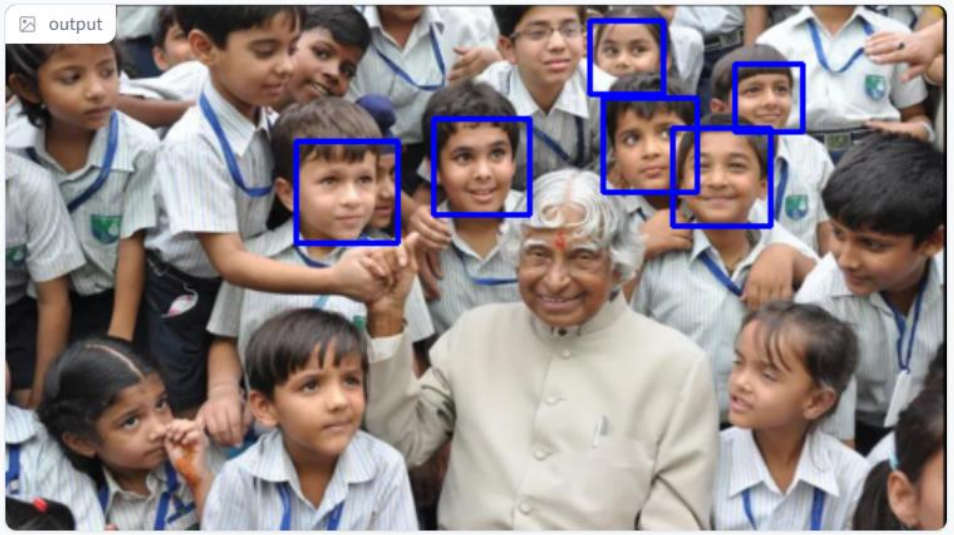
```
iface = gr.Interface(
    fn=detect_faces,
    inputs="image",
    outputs="image",
    title="Face Detection",
    description="Upload an image, and the model will detect faces and draw bounding boxes around them.",
)
iface.launch()
```

Adding parameters to the Haar Cascade Face Detection



app.py · johko/haar-cascade-frontal-face-detection at main (huggingface.co)


img
MinSize in Pixel
30
Min Number of Neighbours
5
Scale Factor
1.1
Clear
Submit


output

Understanding Interface class

- The Interface class is designed to create demos for machine learning models which accept one or more inputs, and return one or more outputs.

The Interface class has three core arguments:

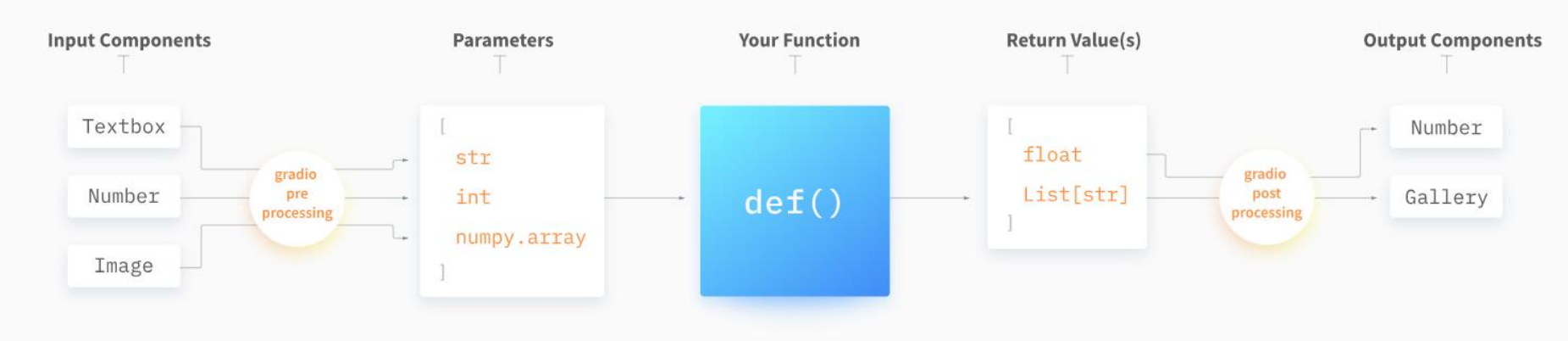
- fn: the function to wrap a user interface (UI) around
- inputs: the Gradio component(s) to use for the input. The number of components should match the number of arguments in your function.
- outputs: the Gradio component(s) to use for the output. The number of components should match the number of return values from your function.

Source : https://www.gradio.app/

- The inputs and outputs arguments take one or more Gradio components.
- Gradio includes more than 30 built-in components (such as the gr.Textbox(), gr.Image(), and gr.HTML() components) that are designed for machine learning applications.

Gradio Built-in Components

- Gradio includes pre-built components that can be used as inputs or outputs in your Interface or Blocks with a single line of code.
- Components include preprocessing steps that convert user data submitted through browser to something that be can used by a Python function, and postprocessing steps to convert values returned by a Python function into something that can be displayed in a browser.
- Consider an example with three inputs (Textbox, Number, and Image) and two outputs (Number and Gallery),



Hugging Face

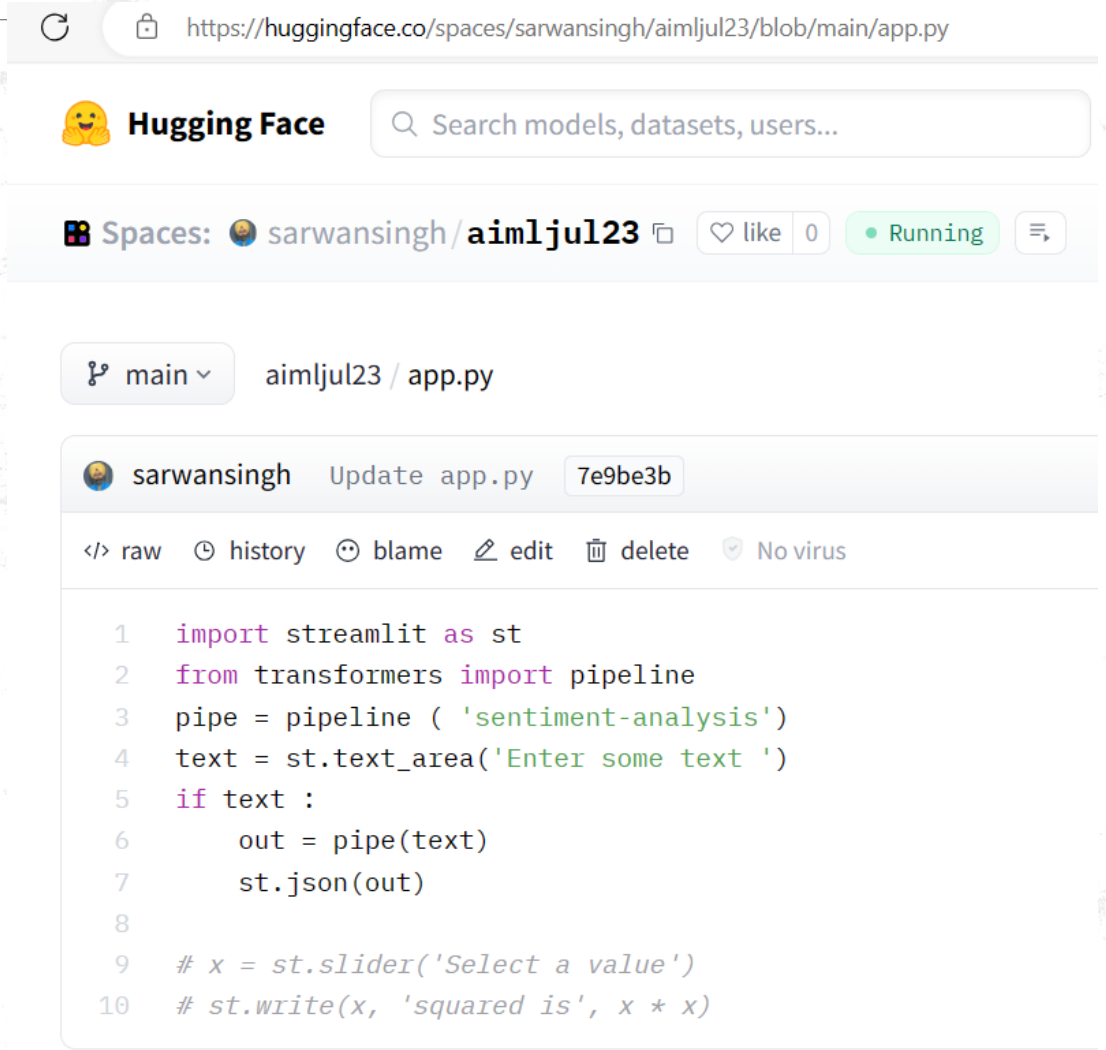
GETTING STARTED

getting started with Huggingface.co

- Hugging Face Spaces make it easy for you to create and deploy ML-powered demos in minutes.
- Each Spaces environment is limited to 16GB RAM, 2 CPU cores and 50GB of (not persistent) disk space by default, which you can use free of charge

First app with Spaces

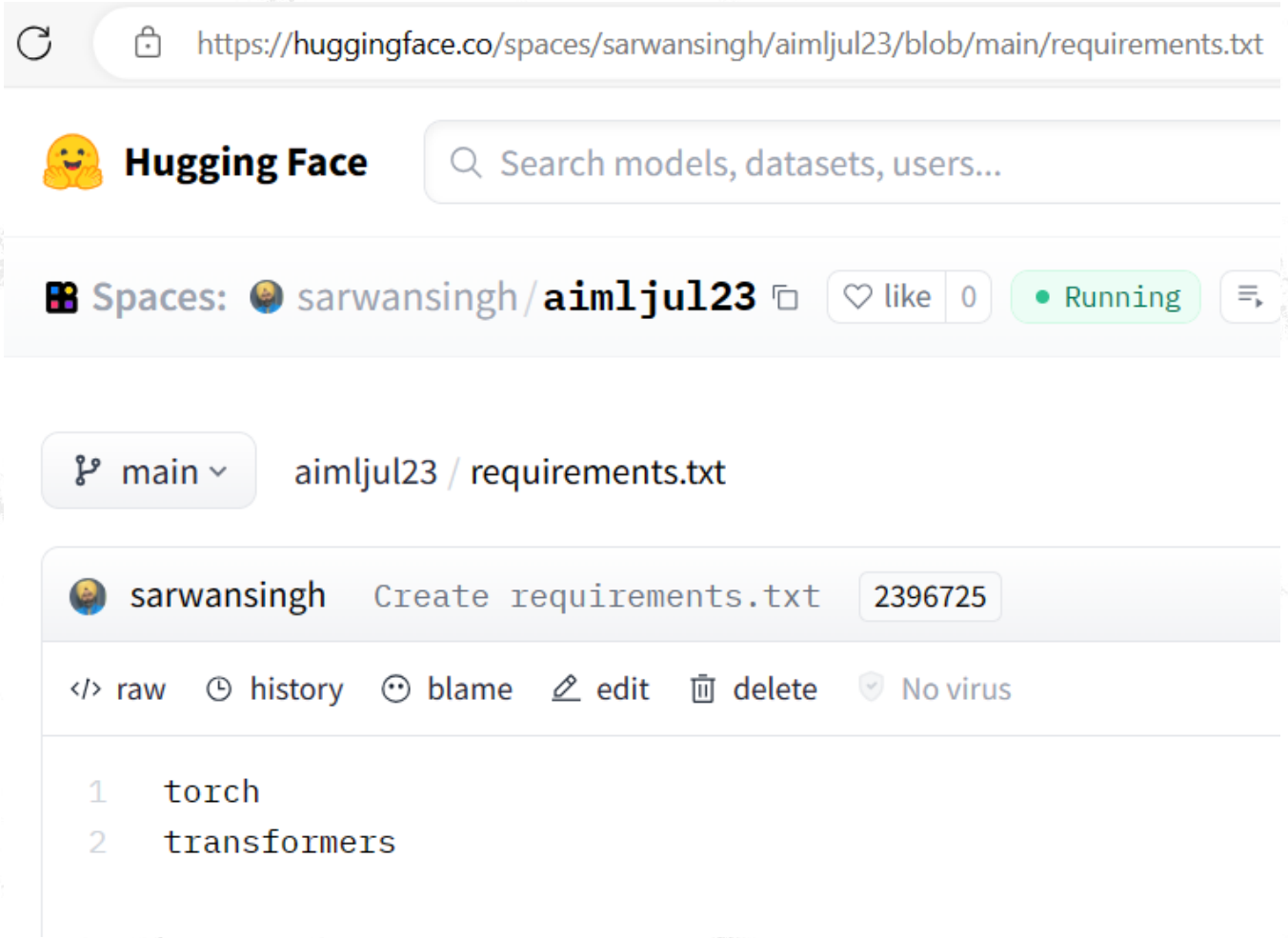
```
1. import streamlit as st
2. from transformers import pipeline
3. pipe = pipeline ( 'sentiment-analysis')
4. text = st.text_area('Enter some text ')
5. if text :
6.     out = pipe(text)
7.     st.json(out)
```



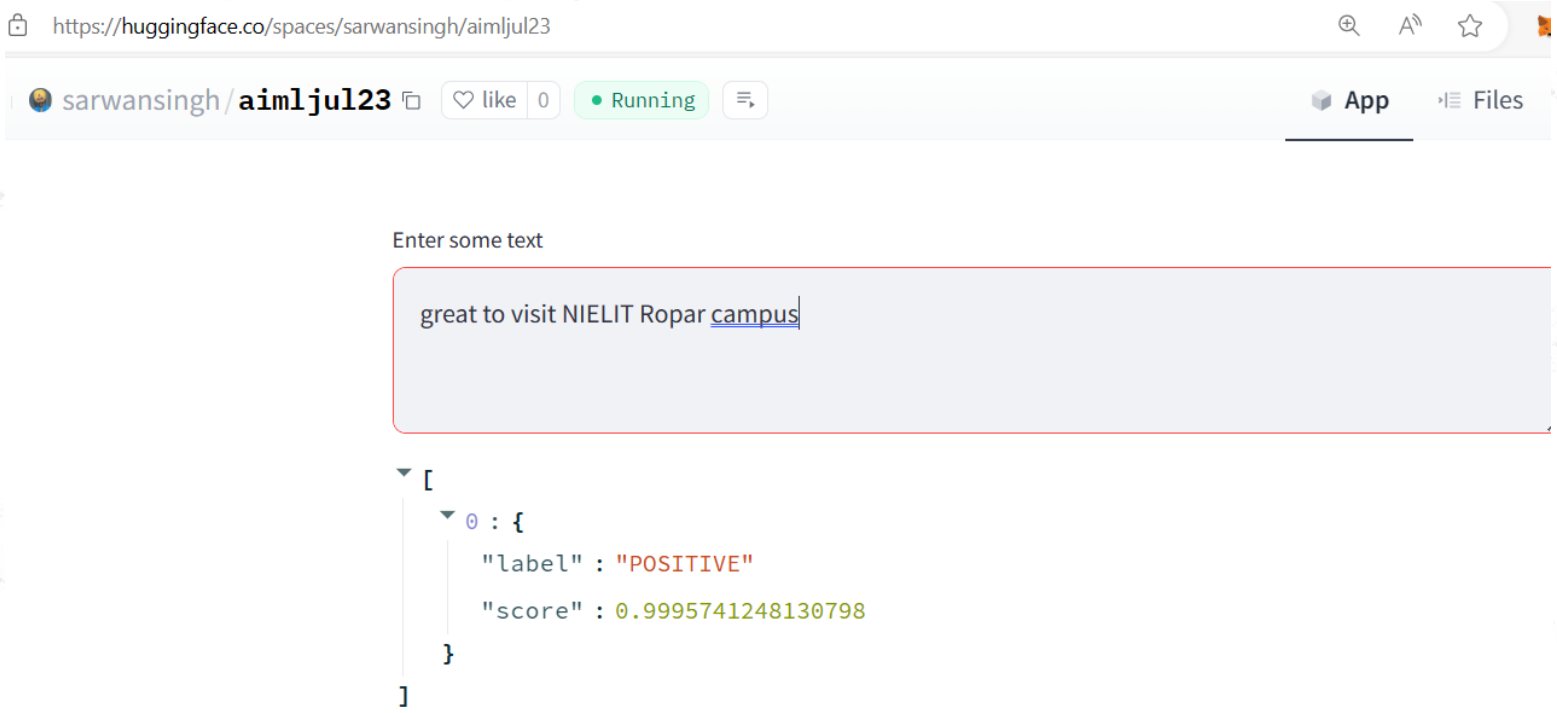
Aimljuly24 - a Hugging Face Space by sarwansingh

requirements.txt

1. torch
2. transformers



Output - *running app*



Introduction

- Streamlit is a software company offering an open-source platform for machine learning and data science teams to create data applications with python that is headquartered in San Francisco, California and was founded in 2018 by Adrien Treuille, Amanda Kelly, and Thiago Teixeira.

Adrien Treuille

CEO and Co-founder of Streamlit

(Acquired by Snowflake)

Source : https://golden.com/wiki/Streamlit-5KMKYAB

About Streamlit

- Streamlit is a modern, easy-to-use, open-source python library that allows developers to build beautiful and interactive data applications.
- Easy to create beautiful visualizations and interactive dashboards.
- Streamlit is built using the Python programming language, making it handy for data scientists and machine learning engineers already familiar with Python.
- Streamlit is designed to be easy to learn and use, with a simple and intuitive API that makes it easy to get started.
- Streamlit works by using Flask to provide a server-side environment for Python code and React to provide a client-side environment for rendering and interacting with the results of that code.

About Streamlit

- Streamlit platform uses python scripting, APIs, widgets, instant deployment, team collaboration tools, and application management solutions to help data scientists and machine learning engineers create python-based applications.
- Applications created using Streamlit range from applications capable of real time object detection, geographic data browsers, deep dream network debuggers, to face-GAN explorers.
- Frameworks compatible with Streamlit include: Scikit Learn, Altair, Bokeh, latex, Keras, Plotly, OpenCV, Vega-Lite, PyTorch, NumPy, Seaborn, Deck.GL, TensorFlow, Python, Matplotlib, and Pandas.

Aspect	Streamlit	Gradio
Ease of Use	More advanced with a steeper learning curve; offers extensive customization options.	Intuitive and straightforward; quick to set up and easy for beginners.
Learning Curve	Requires time to master its advanced features and customization capabilities.	Easier to pick up due to its simplified interface and focus on rapid development.
Deployment	Supports deployment with community support and extensive documentation; real-time updates.	Flexible deployment options; real-time updates with simpler setup.
Visual Appeal	Focuses on creating visually engaging and interactive dashboards with extensive customization.	Emphasizes simplicity; visual customization is more limited compared to Streamlit.
Integration Options	Suitable for projects requiring advanced customization and complex integrations.	Ideal for simpler, interactive applications with quicker development cycles.
Available Resources	Large support base, detailed documentation, and extensive examples available.	Good documentation but a smaller support base compared to Streamlit.
User Base	Larger community presence with a diverse pool of expertise and active collaboration.	Smaller user base; growing but not as extensive as Streamlit's community.

Sample code

```
import streamlit as st
import pandas as pd
import numpy as np
# Title of the dashboard
st.title('Simple Data Dashboard')
# Create a sample dataframe
df = pd.DataFrame({
    'Column A': np.random.randn(100),
    'Column B': np.random.randn(100),
    'Column C': np.random.randn(100)
})
# Display the dataframe
st.write("Here's a sample dataframe:", df)
# Add a line chart
st.line_chart(df)
```

First Streamlit Space: Hot Dog Classifier

Streamlit Spaces

- **Streamlit** gives users freedom to build a full-featured web app with Python in a *reactive* way.
- Your code is rerun each time the state of the app changes.
- Streamlit is also great for data visualization and supports several charting libraries such as Bokeh, Plotly, and Altair.
- Selecting Streamlit as the SDK when creating a new Space will initialize your Space with the latest version of Streamlit by setting the sdk property to streamlit in your README.md file's YAML block. If you'd like to change the Streamlit version, you can edit the sdk_version property

Step 1

- create a **Hot Dog Classifier** Space with Streamlit that'll be used to demo the julien-c/hotdog-not-hotdog model, which can detect whether a given picture contains a hot dog

- Python script uses a Transformers pipeline to load the julien-c/hotdog-not-hotdog model, which is used by the Streamlit interface. The Streamlit app will expect you to upload an image, which it'll then classify as *hot dog* or *not hot dog*

```
1.  import streamlit as st
2.  from transformers import pipeline
3.  from PIL import Image
4.  pipeline = pipeline(task="image-classification", model="julien-c/hotdog-not-hotdog")
5.  st.title("Hot Dog? Or Not?")
6.  file_name = st.file_uploader("Upload a hot dog candidate image")
7.  if file_name is not None:
8.      col1, col2 = st.columns(2)
9.      image = Image.open(file_name)
10.     col1.image(image, use_column_width=True)
11.     predictions = pipeline(image)
12.     col2.header("Probabilities")
13.     for p in predictions:
14.         col2.subheader(f"{ p['label'] }: { round(p['score'] * 100, 1)}%")
```

Embed Streamlit Spaces on other webpages

- HTML <iframe> tag is used to embed a Streamlit Space as an inline frame on other webpages. Simply include the URL of your Space, ending with the .hf.space suffix. To find the URL of your Space, you can use the “Embed this Space” button from the Spaces options.

1. <iframe
2. src="https://NimaBoscarino-hotdog-streamlit.hf.space? embed=true"
3. title="My awesome Streamlit Space"
4. ></iframe>

```
<iframe
    src="https://sarwansingh-aimljul23.hf.space"
    frameborder="0"
    width="850"
    height="450"
></iframe>
```

Embeded in [AIML (aimljul23f.glitch.me)]