Gradio and Streamlit

Shweta Ajay Shinde
Masters in Data Analytics, San Jose State University
Data 230: Data Visualization
Instructor: Venkata Duvvuri
October 29, 2024

Gradio and Streamlit

Simplified Report on Gradio vs. Streamlit for Flower Prediction and Simple Calculator Apps

a) Introduction:

This report compares two popular frameworks, Gradio and Streamlit, for building interactive web applications. We will look at how both frameworks can be used to create a flower prediction app and a simple calculator app.

b) Installation:

To get started, install the frameworks using:

pip install streamlit
pip install gradio

c) Overview of Applications

a) Flower Prediction App

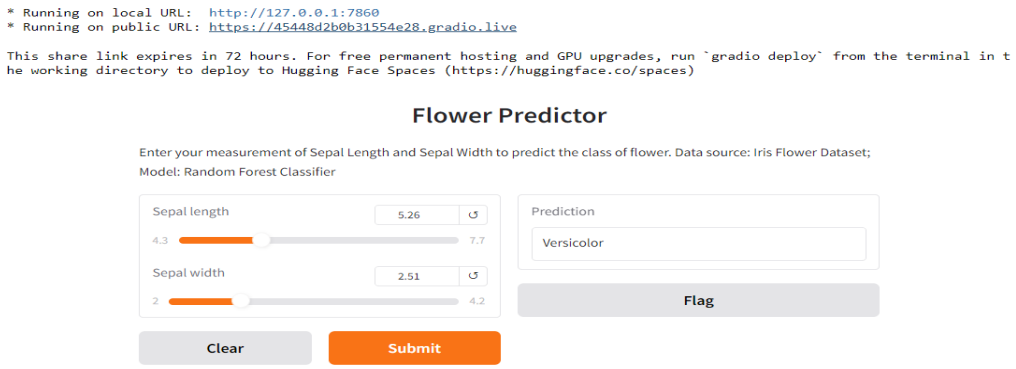
Purpose: To predict the type of Iris flower based on sepal length and width.

Gradio Implementation:

- **User Input:**
  - Users provide sepal length and width using sliders.
  - Command: *gr.Slider* is used to create input sliders.
- **Output:**
  - The predicted flower type is displayed as text in a textbox.
  - Command: *outputs="text"* specifies the output format.
- **Features:**
  - Quick Setup: Easy to implement with minimal code.
  - Built-in Sharing: Offers a link to share the app with others immediately.
  - Real-time Feedback: Predictions are displayed instantly as users adjust slides and click “Submits”.
  - Refer Figure 1 for your reference.
  - For code reference, click here.

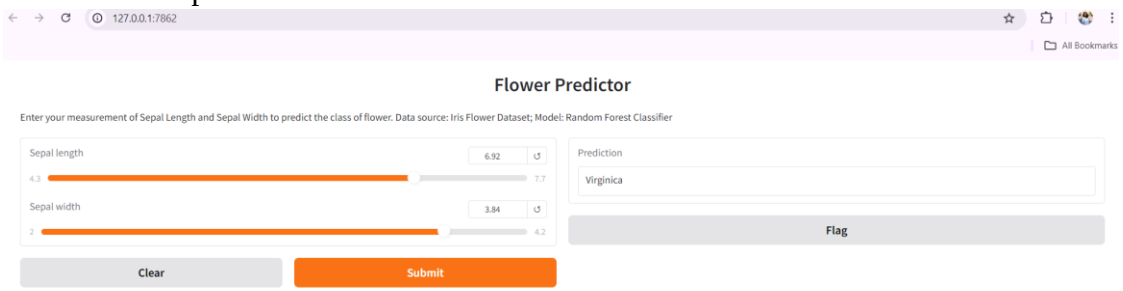
Figure 1:

Gradio flower prediction Jupyter notebook



Gradio and Streamlit

Figure 2:
Gradio flower prediction launch



Streamlit Implementation:

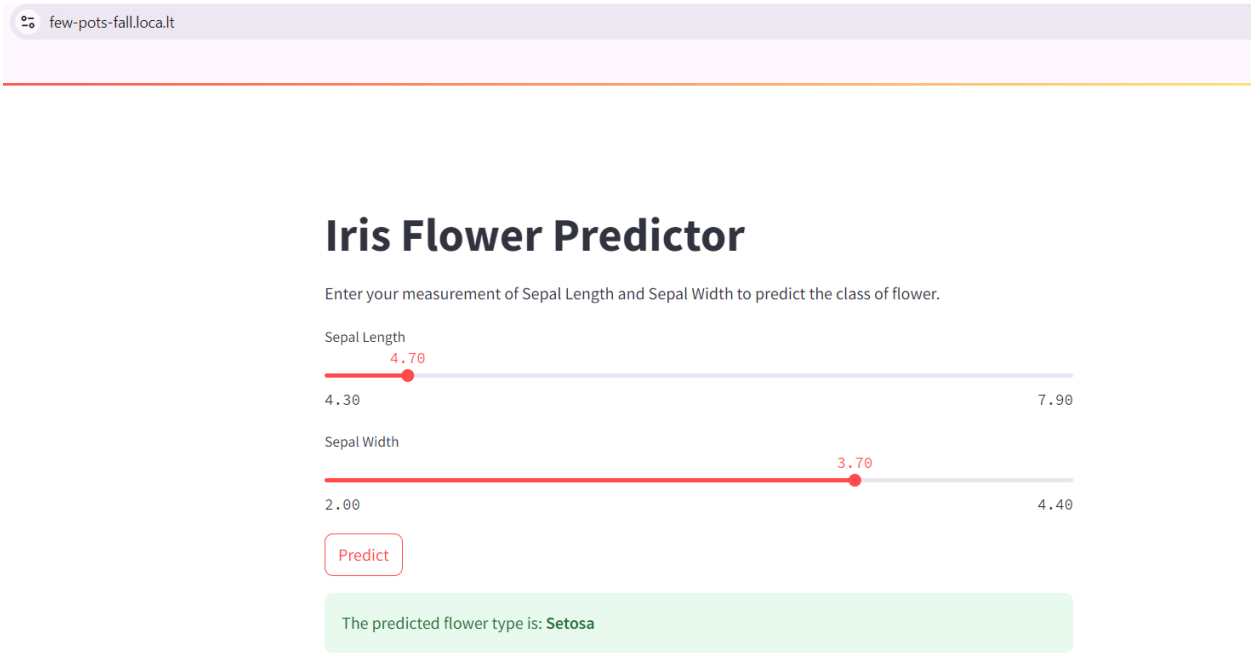
- **User Input:**
  - Similar to Gradio, users input sepal measurements through sliders.
  - Command: *st.slider* creates sliders for input.
- **Output:**
  - The result is shown when a button is clicked.
  - Command: *st.success* displays the prediction result in a visually distinct format.
- **Features:**
  - Layout Options: Offers more flexibility in arranging elements on the page.
  - Interactivity: Users can see updates based on inputs in real-time, enhancing engagement.
  - Conditional Display: Streamlit allows for dynamic content changes based on user interaction, providing a richer user experience.
  - Launching a Streamlit app from Google Colab requires additional steps. Users need to set up specific commands and configurations to ensure the app functions correctly in the Colab environment.
  - Please refer Figure 3 and 4 for your reference.
  - For code reference, click here.

Figure 3:
Streamlit flower prediction code output.

```
!streamlit run flower2.py &>/content/logs.txt & npx localtunnel --port 8501 & curl ipv4.icanhazip.com

34.106.168.56
your url is: https://few-pots-fall.loca.lt
```

Figure 4:
Streamlit flower prediction model



b) Simple Calculator App

Purpose: To perform basic arithmetic operations like addition, subtraction, multiplication, and division.

Gradio Implementation:

- **User Input:**
  - Users enter numbers and select operations using dropdown menus or buttons.
  - Command: *gr.Dropdown* or *gr.Button* for selecting operations.
- **Output:**
  - The result of the calculation is displayed as text.
  - Command: outputs="text" specifies the output format.
- **Features:**
  - Simplicity: The interface is straightforward, making it easy for users to perform calculations.
  - Quick Setup: Allows for rapid development with minimal coding**.**
  - Please refer to Figure 5 and 6 for your reference.
  - For code reference, click here.

Figure 5:
Gradio Simple calculator jupyter notebook output.

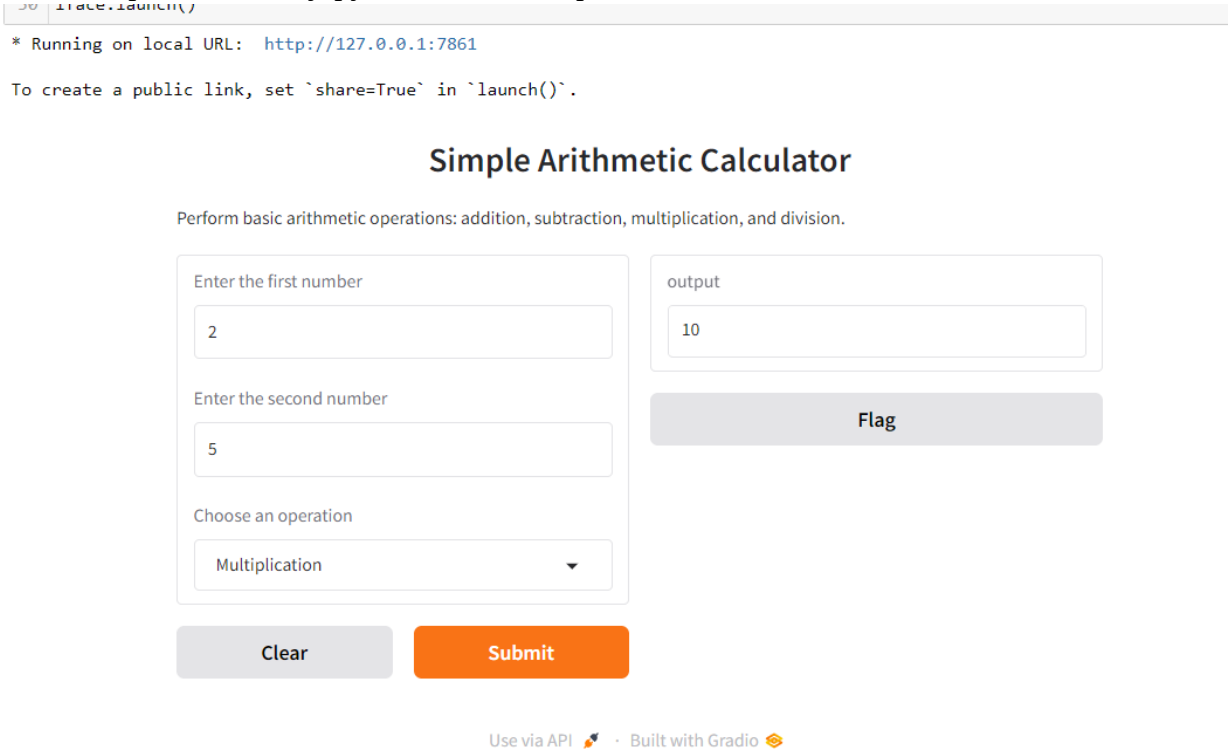
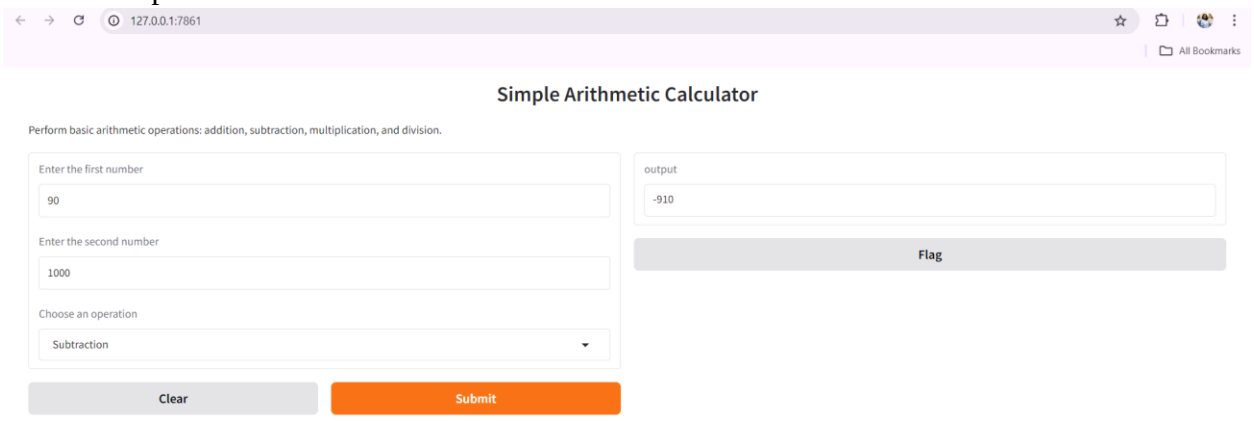


Figure 6:
Gradio Simple calculator launch.



Gradio and Streamlit

Streamlit Implementation:

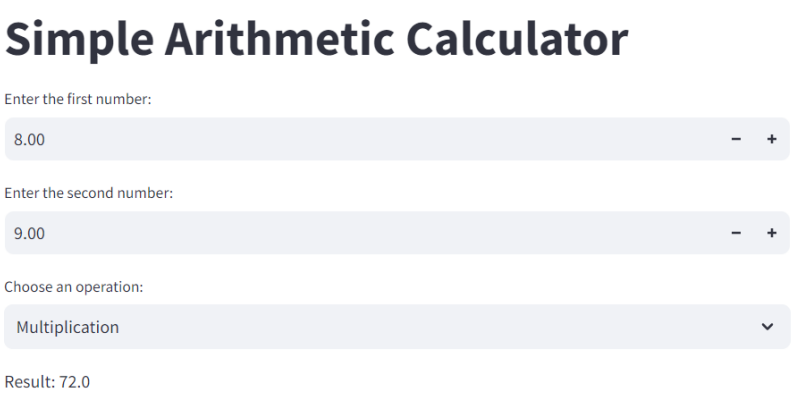
- **User Input**:
  - Similar to Gradio, users provide numbers and select operations.
  - Command: *st.number_input* for entering numbers and *st.selectbox* for selecting operations.
- **Output**:
  - Results are displayed dynamically on the app.
- **Features**:
  - Real-Time Updates: Streamlit provides immediate feedback as users input values and select operations.
  - Interactivity: Offers a more engaging user experience with the ability to see results instantly.
  - Launching a Streamlit app from Google Colab requires additional steps. Users need to set up specific commands and configurations to ensure the app functions correctly in the Colab environment.
  - Please refer to Figure 7 and 8 for your reference.
  - For code reference, click here.

Figure 7:
Streamlit simple calculator code output.

```
!streamlit run app.py &>/content/logs.txt & npx localtunnel --port 8501 & curl ipv4.icanhazip.com

34.106.168.56
your url is: https://wild-emus-flow.loca.lt
```

Figure 8:
Streamlit simple calculator link output.



Gradio and Streamlit

d) Code Migration

Gradio

- **Launch Interface**: Use *iface.launch(share=True, debug=True)* to start your Gradio interface.
  - share=True: Allows you to generate a public link to share the app.
  - debug=True: Enables debugging mode for more detailed error messages.

Streamlit

1. **Save Your Code**: Use *%%writefile flower2.py* in a Google colab cell to save your code into a file named flower2.py.
2. **Run the App**: Use *!streamlit run flower2.py* to execute the Streamlit application from the command line or Jupyter Notebook

Summary

- **Gradio**: Launch an interactive web app easily with sharing and debugging.
- **Streamlit**: Write your code to a file and run the app to see it in action.

Command Differences:

- **Input Creation**:
  - Gradio uses *gr.Dropdown gr.Button*, while Streamlit uses *st.selectbox* and st.number_input.
- **Output Handling**:
  - Gradio utilizes *outputs="text"*, while Streamlit employs *st.write* or *st.success* to format and display results.

Additional Points:

- **Integration:**
  - Both frameworks can easily integrate with existing machine learning models, making deployment straightforward.
- **Documentation and Community Support:**
  - Gradio has concise documentation focused on model deployment, while Streamlit offers extensive documentation with examples for creating complex apps.
- **Use Cases:**
  - Gradio is particularly useful for rapid prototyping and sharing models, while Streamlit is better suited for building full-featured applications with complex layouts and interactive elements.

Comparison of Gradio and Streamlit

Feature	Gradio	Streamlit
Ease of Use	Very simple to set up	Slightly more complex
Input Options	Limited, but straightforward	Rich variety of interactive tools
Output Display	Quick text output	More control over layout
Customization	Basic styling options	Highly customizable layouts
Real-Time Updates	Occurs on interaction	Updates occur based on state changes
Sharing	Easy sharing with a link	Requires additional setup for sharing

e) Conclusion

Both Gradio and Streamlit are excellent choices for building interactive applications. Gradio is great for quick demonstrations and ease of use, while Streamlit offers more flexibility and customization for complex applications. Depending on the project's needs, developers can choose the framework that best fits their goals.

References

Lundervold, A. (2023). *Simple Gradio Kaggle example*. Kaggle. https://www.kaggle.com/code/alexanderlundervold/simple-gradio-kaggle-example

Machine Learning Projects. (n.d.). *Calculator using Streamlit*. https://machinelearningprojects.net/calculator-using-streamlit/#google_vignette

Shahab, H. (2021). *Streamlit vs. Gradio: A comprehensive comparison*. Medium. https://medium.com/@ShahabH/streamlit-vs-gradio-a-comprehensive-comparison-cc2f28b7b832

Streamlit Community. (2022). *How to launch Streamlit app from Google Colab notebook?* Streamlit Discuss. https://discuss.streamlit.io/t/how-to-launch-streamlit-app-from-google-colab-notebook/42399/2

Shinde, S. K. (2023). *DV_Assignments* [GitHub repository]. https://github.com/ShindeShwetaK/DV_Assignments