Part I

Chapter 1

An Introduction to Streamlit

1.1

st.pyplot() is a function that lets us use all the power of the popular matplotlib library and push our matplotlib graph to Streamlit. Once we create a figure in matplotlib, we can explicitly tell Streamlit to write that to our app with the st.pyplot() function.

1.2 Finishing touches – adding text to Streamlit

Other than st.write(), we also can utilize other built-in functions that format our text for us, such as st.title(), st.header(), st.markdown(), and st.subheader(). Using these five functions helps to format text in our Streamlit apps easily and keeps sizing consistent for bigger apps.

More specifically, st.title() will place a large block of text in our app, st.header() uses a slightly smaller font than st.title(), and st.subheader() uses an even smaller one. Other than those three, st.markdown() will allow anyone already familiar with Markdown to use the popular markup language in our Streamlit apps.

Chapter 2

Uploading, Downloading, and Manipulating Data

2.1 An introduction to caching

A good analogy for an app's cache is a human's short-term memory, where we keep bits of information close at hand that we think might be useful. When something is in our short-term memory, we don't have to think very hard to get access to that piece of information. In the same way, when we cache a piece of information in Streamlit, we are making a bet that we'll use that information often.

The way Streamlit caching works more specifically is by storing the results of a function in our app, and if that function is called with the same parameters by another user (or by us if we rerun the app), Streamlit does not run the same function but instead loads the result of the function from memory.

There are two Streamlit caching functions, one for data (`st.cache_data`) and one for resources like database connections or machine learning models (`st.cache_resource`).

2.2 Persistence with Session State

Enter `st.session_state`. Session State is a Streamlit feature that is a global dictionary that persists through a user' s session.

Chapter 3

Data Visualization

3.1

3.2 Streamlit visualization use cases

Some Streamlit users are relatively experienced Python developers with well-tested workflows in visualization libraries of their choice. For these users, the best path forward is the one we've taken so far, which is to create graphs in our library of choice (Seaborn, Matplotlib, Bokeh, and so on) and then use the appropriate Streamlit function to write this to the app.

Other Streamlit users will have less experience in Pythonic graphing, and especially for these users, Streamlit offers a few built-in functions.

3.2.1 Streamlit's built-in graphing functions

There are four built-in functions for graphing – `st.line_chart()`, `st.bar_chart()`, `st.area_chart()`, and `st.map()`.

3.2.2 Bokeh

We can call Bokeh graphs using the same format as Plotly. First, we create the Bokeh graph, and then we use the `st.bokeh_chart()` function to write the app to Streamlit. In Bokeh, we have to first instantiate a Bokeh figure object, and then change aspects of that figure before we can plot it out. The important lesson here is that if we change an aspect of the Bokeh figure object after we call the

`st.bokeh_chart()` function, we will not change the graph shown on the Streamlit app.

Chapter 4

Machine Learning and AI with Streamlit

Chapter 5

Beautifying Streamlit Apps

5.1 Summary

Working with columns in Streamlit Streamlit allows us to format our app into dynamic columns using the `st.columns()` feature. We can divide our Streamlit app into multiple columns of different lengths and then treat each column as its own unique space (called a **container**) in our app to include text, graphs, images, or anything else we would like.

Using Streamlit tabs `st.tabs` works very similarly to `st.columns`, but instead of telling Streamlit the number of tabs we want, we instead pass along the names of the tabs and then use now-familiar 'with' statements to place content into the tab.

Using the Streamlit sidebar We can use the Streamlit `sidebar`, which allows us to place a minimizable sidebar on the left side of the Streamlit app and add any Streamlit component to it.

Picking colors with a color picker Streamlit's approach to this problem is `st.color_picker()`, which lets the user pick a color as a part of their user input, and returns that color in a hex string (which is a unique string that defines very specific color shades used by most graphing libraries as input).

Multi-page apps Streamlit 创建多页面应用程序的方式是在与我们的 Streamlit 应用程序相同的目录中查找名为 `pages` 的文件夹，然后将 `pages` 文件夹内的每个Python 文件作为其自己的 Streamlit 应用程序运行。

Editable DataFrames Streamlit released `st.data_editor`, a way to give users edit ability on top of an `st.dataframe-style` interface.