Streamlit

Team 4

Yudha, Khanif, Fadilah

Contents

- **Introduction**
- **Comparison** with **dash** and **flask**
- **Pros** & **cons**
- **Installation** and Basic **Components**
- How to **deploy**

What is Streamlit?

Intro to **Streamlit**

Streamlit is an open-source Python library that makes it easy to create and share beautiful and **interactive custom web apps** for machine learning and data science.

Streamlit is **compatible with** several major libraries and frameworks such as **Scikit-Learn, Tensorflow, Keras,** OpenCV, PyTorch, **Plotly, Seaborn, NumPy,** and more.

Streamlit vs Dash vs Flask

Streamlit vs other similar **python apps**

	Simplicity	Maturity	Flexibility	Primary Use
Streamlit	**A**	C	B	Dashboards
Dash	B	B	B	Dashboards
Flask	C	A	A	Web Interfaces

As well, Streamlit allows you to build a web UI or a dashboard much faster than Dash or Flask.
e.g.: building a portfolio that have a tight deadline.

Reference: Terence Shin - [Towardsdatascience]

Pros & Cons

Pros

1. Accessible for everyone who understands Python, **no need to understand HTML and CSS.**
2. Has a **wide range and easy** to use **UI** components.
3. **No need** to worry about **routing**.
4. **Super-fast development** to deployment time. Literally minutes.

Cons

1. **Require some time to learn** its own **syntax.**
2. **Not that flexible**: limited set of widgets and doesn't integrate with Python notebooks.
3. **Will not scale**.
4. **Relatively new**, sometimes it's hard to find answers to your questions.

Reference: Terence Shin - Towardsdatascience & Alejandro Colocho - The Startup

Installation
&
Basic **Components**

Installation & Import

Download in seconds and start creating beautiful data and machine learning apps.

```
pip install streamlit
# or conda install streamlit

import streamlit as st
```

You can go to their homepage: www.streamlit.io

Basic **Components**

- **Stateless** Components

  Only send data ***to*** the browser.

  e.g.: `st.markdown`

- **Bidirectional** Components

  Have internal state and send data ***back from*** the browser.

  e.g.: `st.slider`

```
import streamlit as st

x = st.slider('x')
st.markdown(f'`{x}` squared is `{x * x}`')
```

Basic **Components** (cont)

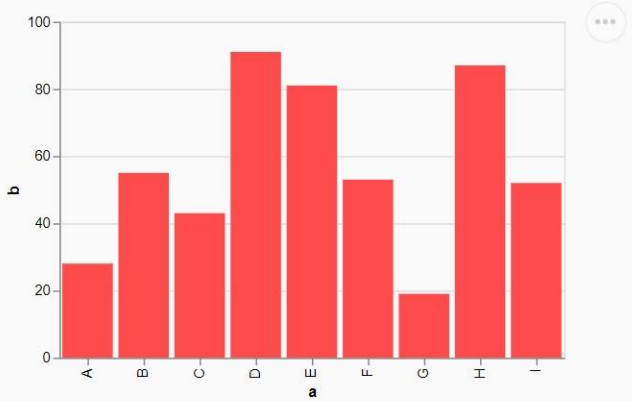
Title, Sidebar and Markdown

```
st.title("Predicting Diabetes Web App")
st.sidebar.title("Model Selection Panel")
st.markdown("Affected by Diabetes or not ?")
st.sidebar.markdown("Choose your model and its parameters")
```

Pick a number

0 100

```
number = st.slider("Pick a number", 0, 100)
```




```
st.altair_chart(my_chart)
```

Streamlit cheat sheet

streamlit.io

This cheat sheet is a summary of the docs

I also recommend streamlitopedia

How to install and import

```
$ pip install streamlit
```

Import convention

```
>>> import streamlit as st
```

Add widgets to sidebar

```
st.sidebar.<widget>
```

```
>>> my_val = st.sidebar.text_input('I:')
```

Command line

```
$ streamlit --help
$ streamlit run your_script.py
$ streamlit hello
$ streamlit config show
$ streamlit cache clear
$ streamlit docs
$ streamlit --version
```

Pre-release features

To access beta and experimental features

```
pip uninstall streamlit
pip install streamlit-nightly --upgrade
```

Magic commands

Magic commands allow you to implicitly `st.write()`

```
''' _This_ is some __Markdown__ '''

a=3
'a', a

'dataframe:', data
```

Display text

```
st.text('Fixed width text')
st.markdown('_Markdown_') # see *
st.latex(r''' e^{i\pi} + 1 = 0 ''')
st.write('Most objects') # df, err, func, keras!
st.write(['st', 'is <', 3]) # see *
st.title('My title')
st.header('My header')
st.subheader('My sub')
st.code('for i in range(8): foo()')
```

* optional kwarg unsafe_allow_html = True

Display data

```
st.dataframe(data)
st.table(data.iloc[0:10]
st.json({'foo':'bar','fu':'ba'})
```

Display charts

```
st.line_chart(data)
st.area_chart(data)
st.bar_chart(data)
st.pyplot(fig)
st.altair_chart(data)
st.vega_lite_chart(data)
st.plotly_chart(data)
st.bokeh_chart(data)
st.pydeck_chart(data)
st.deck_gl_chart(data)
st.graphviz_chart(data)
st.map(data)
```

Display media

```
st.image('./header.png')
st.audio(data)
st.video(data)
```

Display interactive widgets

```
st.button('Hit me')
st.checkbox('Check me out')
st.radio('Radio', [1,2,3])
st.selectbox('Select', [1,2,3])
st.multiselect('Multiselect', [1,2,3])
st.slider('Slide me', min_value=0, max_value=10)
st.text_input('Enter some text')
st.number_input('Enter a number')
st.text_area('Area for textual entry')
st.date_input('Date input')
st.time_input('Time entry')
st.file_uploader('File uploader')
st.beta_color_picker('Pick a color')
```

Use widgets' returned values in variables:

```
>>> for i in range(int(st.number_input('Num:'))): foo()
>>> if st.sidebar.selectbox('I:',['f']) == 'f': b()
>>> my_slider_val = st.slider('Quinn Mallory', 1, 88)
>>> st.write(slider_val)
```

Control flow

```
st.stop()
```

Display code

```
st.echo()

>>> with st.echo():
>>>     # Code below both executed and printed
>>>     foo = 'bar'
>>>     st.write(foo)
```

Display progress and status

```
st.progress(progress__variable_1_to_100)

st.spinner()

>>> with st.spinner(text='In progress'):
>>>     time.sleep(5)
>>>     st.success('Done')

st.balloons()
st.error('Error message')
st.warning('Warning message')
st.info('Info message')
st.success('Success message')
st.exception(e)
```

Placeholders, help, and options

```
st.empty()

>>> my_placeholder = st.empty()
>>> my_placeholder.text('Replaced!')

st.help(pandas.DataFrame)

st.get_option(key)
st.set_option(key)

st.beta_set_page_config(layout='wide')
```

Mutate data

```
DeltaGenerator.add_rows(data)

>>> my_table = st.table(df1)
>>> my_table.add_rows(df2)

>>> my_chart = st.line_chart(df1)
>>> my_chart.add_rows(df2)
```

Optimize performance

```
@st.cache

>>> @st.cache
... def foo(bar):
...     # Mutate bar
...     return data
...
>>> d1 = foo(ref1)
>>> # Executes as first time
>>>
>>> d2 = foo(ref1)
>>> # Does not execute; returns cached value, d1==d2
>>>
>>> d3 = foo(ref2)
>>> # Different arg, so function executes
```

Reference: daniellewisDL

Deployment

Deploying with **Heroku** in 3 Steps

1. Make sure to have **requirements.txt** file and state which version of the package you want Heroku to install.

2. Include **Procfile**, **runtime.txt**, and **setup.sh**.

3. **Push** to Heroku!

Checkout the example from us here: team4-streamlit.herokuapp.com

source code on github: thunder-talk-team4-streamlit

Procfile

```
web: sh setup.sh && streamlit run app.py
```

setup.sh

```
mkdir -p ~/.streamlit/
echo "[general]
email = \"email@com\"
" > ~/.streamlit/credentials.toml
echo "[server]
headless = true
port = $PORT
enableCORS = false
" > ~/.streamlit/config.toml
```

runtime.txt

```
python-3.6.1
```

Only compatible with python 3.6+

Thank You!

Team 4