



如何用 Python 和 Streamlit 做交互式数据分析产品？



玉树芝兰

2020 年 01 月 15 日

「本文参与少数派 2019 年度征文 + 效率有心得」

不用学前端编程，你就能用 Python 简单高效写出漂亮的交互式 Web 应用，将你的数据分析成果立即展示给团队和客户。



痛点

从我开始折腾数据分析工具的那一天，就没有想明白一件事儿——为什么我打算把数据分析的成果做成一个应用，这么难？

其实我需要的核心功能，无非是在网页上接收用户输入，然后做分析处理，把分析结果反馈给用户，完事儿。

可是这谈何容易？

很多人都会笑着告诉你，这得学前端编程，HTML + Javascript 了解一下吧！

什么？你还需要在后台做数据分析？那你就得学 Web 框架了。

你说喜欢 Python？那就学个 Django 或者 Flask 好了。

我也没有看过 Django 和 Flask 的教程，还曾经付费学习过。光是配置环境，就得循序渐进学一堆东西。作为学习的中间成果，我还写了这篇《如何用 Python 做 Web 开发？——Django 环境配置》分享给你。

问题是我在学习中，提不起真正的兴趣。

因为教程里讲的那些功能，我根本不关心。



登录

我为什么要理解那么多的概念？为什么一定要跟那么繁重的数据库操作打交道？为什么几乎所有的样例，都要教我如何做一个 blog ？

我要是想用 blog ，可以直接注册一个免费的啊！难道我要自己开发？

你的教程为什么不干脆教我怎么把数据科学的分析结果，利用这些技术快速变成一个产品？

但是人家写书和做教程的人，就是不疾不徐，坚持一定要教会你，如何做一个 blog 出来.....

我仿佛看见达芬奇的老师教学生画鸡蛋一样。

我相信，这绝不仅仅是我一个人的痛点。

我们都希望尽快把数据分析结果，或是其他的交互功能发布出来，和用户交流。但是因为缺乏这样的简单 Web 包裹，我们不得不每次都给别人展示一个包含了代码的 Jupyter Notebook 。

那些不懂编程的用户，看到代码，就会觉得不适。再看到改变一个输入都需要编程（其实就是改语句中的一个赋值），立刻就决定不玩儿了。

万万没想到，这个痛点，如此容易就解决了。

尝试

我用纯 Python 脚本写了个 Web 应用。

我编写的程序里，没有一丝半毫的 Web 框架，Javascript，甚至是 HTML 。

这玩意儿能用吗？

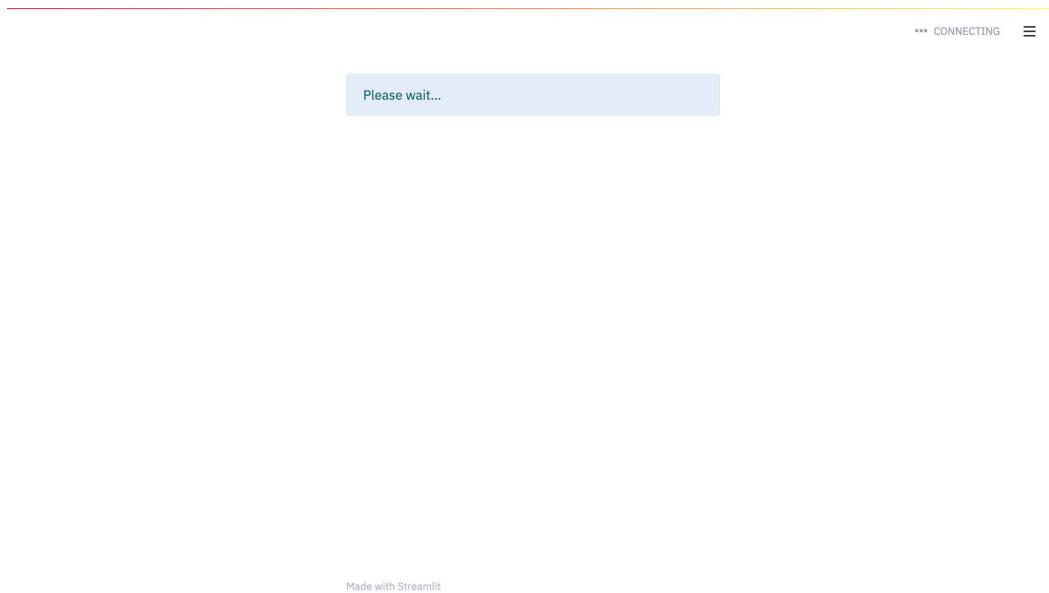
你自己来试试看。

请你打开浏览器，输入以下链接：



登录

你会看到下面的初始化界面。



Made with Streamlit

初始化完毕之后，页面会分成左右两栏。左面是两个下拉候选框，分别让你指定需要分析的数据范围。

The screenshot shows the Streamlit application interface. On the left, there are two dropdown menus: one for 'Which kind of event do you want to explore?' containing 'road closed due to construction' and another for 'Which county?' containing 'Denton'. On the right, the main area has a title 'my first app' and displays a table of data with columns: event_type, time, county, lat, lon. The data includes:

	event_type	time	county	lat	lon
0	road closed due to construction	11/7/2018 12:04:24 AM	Denton	33.0635	-97.2432
1	road closed due to construction	11/7/2018 12:06:10 AM	Denton	33.0635	-97.2432
2	traffic jam	11/7/2018 12:08:49 AM	Denton	33.0635	-97.2432
3	stopped car on the shoulder	11/7/2018 12:23:23 AM	Dallas	32.9262	-96.8195
4	road closed	11/7/2018 12:26:11 AM	Collin	33.1090	-96.8027

Below the table, a message says '根据你的筛选，数据包含 76 行'. At the bottom, there is a small map showing the location of Denton, Texas.

上面一个，是事件类型；

traffic jam
stopped car on the shoulder
road closed
other
object on roadway
major event
pothole
traffic heavier than normal
road construction
fog
accident
slowdown
stopped car
small traffic jam
stopped traffic
heavy traffic

下一个，是事件发生归属地。



登录

- Dallas
- Collin
- Johnson
- Ellis
- Tarrant
- Rockwall
- Van Zandt
- Hill
- Parker
- Kaufman
- Hunt
- Wise
- Palo Pinto
- Hopkins
- Henderson

Footh

如果你看过《如何用 Python 和循环神经网络预测严重交通拥堵？》，应该对这个数据集很熟悉。

而今天，我们是要进行探索性数据分析，也就是根据我们感兴趣的目标，对数据进行整理操作，然后可视化显示。

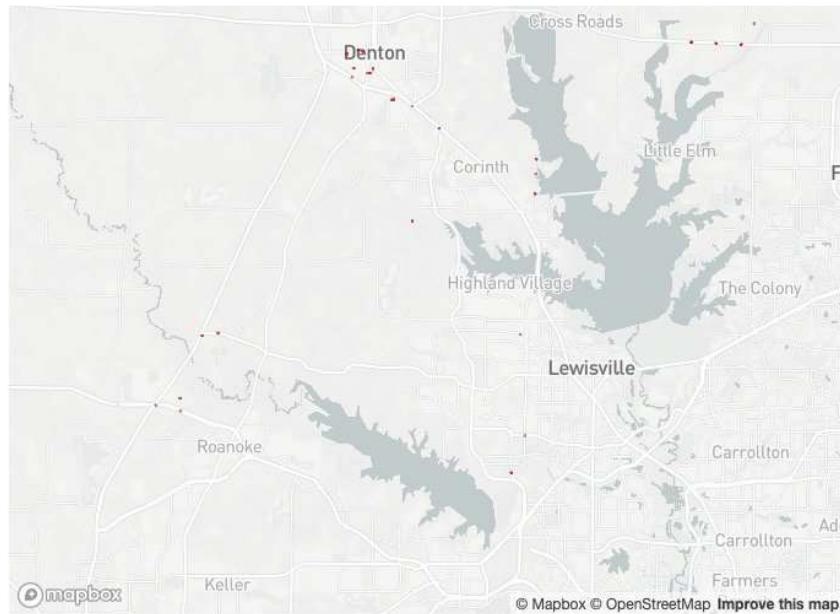


选定之后，你会看到右侧提示两个信息：

- 你筛选之后，数据框包含行数；
- 在层叠地图上的可视化结果。



登录



怎么样？

麻雀虽小，五脏俱全。

虽然咱们这个 Web 应用很简单，不过交互分析该有的功能和流程，基本上都涵盖了。

你可能会问：

王老师，编这么一个应用出来，不简单吧？

学完这篇教程，你就能**自己开发出**这样一个应用来。

幕后

我把这个应用的全部源代码，都为你存储到了 Github 上。请你访问[这个网址](#)获取。

可以看到，一共包含了 4 个文件。

有意思的是，其中 3 个，包括：



登录

- setup.sh
- requirements.txt

都只是部署到远程服务器时，需要用到的配置文件而已。

这些文件的具体使用方法，咱们后面会说明。

也就是说，只有最后一个 `helloworld.py` 是主角，它包含了实现咱们**全部交互式数据
分析功能**的 Python 脚本文件。

这代码，少说也得有几百行吧？

别担心，打开来看看：

```

Branch: master · demo-helloworld-streamlit / helloworld.py
Find file · Copy path
shuyi · 5cb914a · 18 seconds ago
1 contributor

58 lines (33 sloc) 1.29 KB
Raw Blame History ⚙️ 🗑️

1 import streamlit as st
2 import numpy as np
3 import pandas as pd
4
5 st.title("my first app")
6
7 #st.cache
8 def load_data():
9     df = pd.read_csv("data.csv")
10    df = df[["EVENT_TYPE", "CREATE_TIME", "COUNTRY", 'LAT', 'LON']]
11    df.columns = ['event_type', 'time', 'county', 'lat', 'lon']
12    return df
13
14 df = load_data()
15
16 st.table(df.head())
17
18 event_list = df["event_type"].unique()
19
20 event_type = st.sidebar.selectbox(
21     "Which kind of event do you want to explore?", event_list
22 )
23
24 county_list = df["county"].unique()
25
26 county_name = st.sidebar.selectbox(
27     "Which county?", county_list
28 )
29
30
31 part_df = df[(df["event_type"]==event_type) & (df['county']==county_name)]
32
33 st.write(f" 根据你的筛选，数据包含 {len(part_df)} 行 ")
34
35
36 st.map(part_df)

```

上面这张截图，就已经包含了实现交互数据分析功能的**全部代码**。

神奇吧？

解读

这么短的代码，为什么能有如此强大的功能？

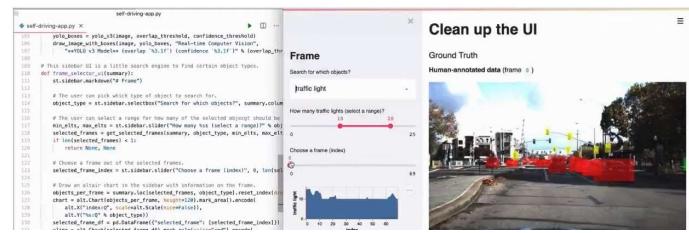
这是因为它背后使用的一个软件包，叫做 streamlit。



Streamlit.

The fastest way to build custom ML tools

Streamlit is an open-source app framework for Machine Learning and Data Science teams. Create beautiful [data apps](#) in hours, not weeks. All in pure Python. All for free.



下面我通过实际操作，带你初步领略一下 streamlit 的威力。

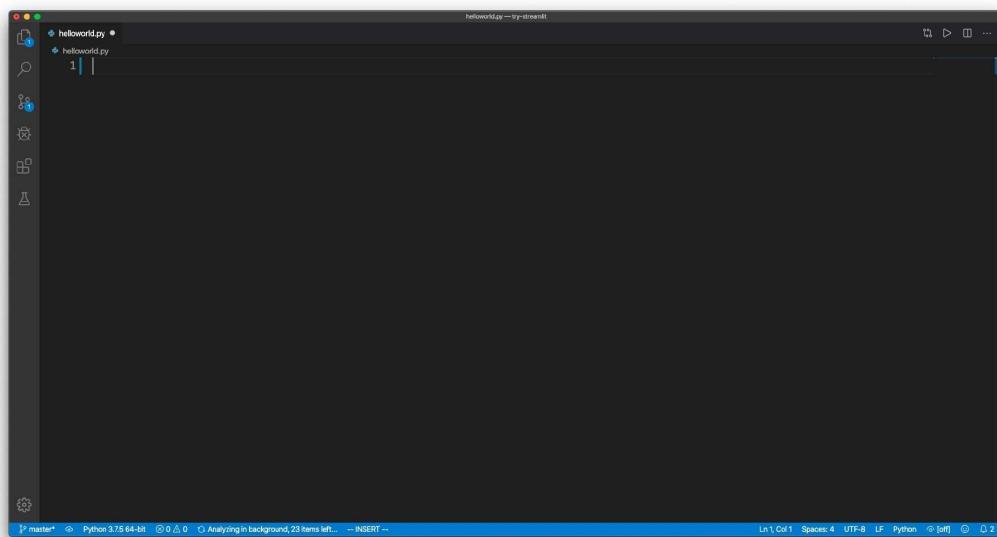
首先请你安装 Anaconda，这个请参考我为你做的视频教程《[如何安装 Python 运行环境 Anaconda?](#)》

然后，你需要打开终端，执行：

```
pip install streamlit
```

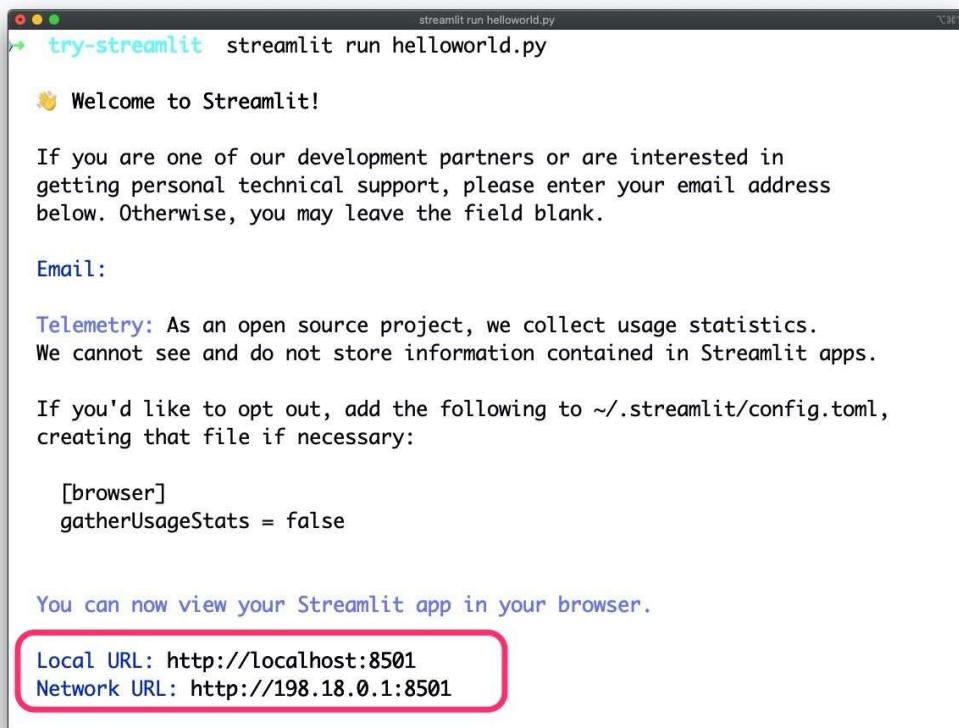
你可以创建一个新目录。然后在目录下新建一个 helloworld.py 文件，并且用任意编辑器打开它。

我这里用 Visual Studio Code 编辑器，来编辑和制作 Python 脚本文件。



```
streamlit run helloworld.py
```

如果一切顺利，你就会看到如下图的提示。



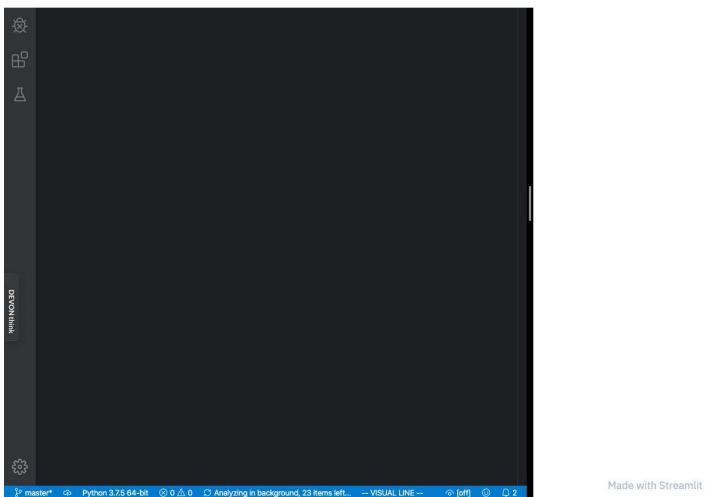
一般来说，你的浏览器会自动开启，并且访问上图中红色标识出的网址。

如果浏览器没有自动开启，你手动开启一个，并且输入上述网址即可。

为了演示方便，我这里把 Visual Studio Code 编辑器缩小到屏幕左侧半部；右边放置 Chrome 浏览器，来显示 Web 应用效果。



登录



我们可以开始尝试了。

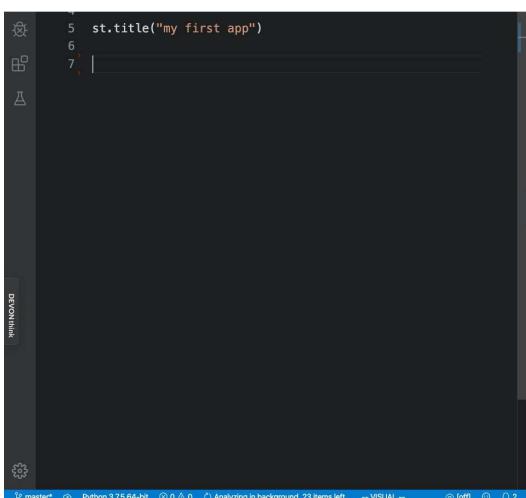
首先在 `helloworld.py` 中输入这些内容：

```
import streamlit as st
import numpy as np
import pandas as pd

st.title("my first app")
```

输入完之后，你不需要去找什么执行按钮。只需要保存一下你对 `helloworld.py` 文件的修改即可。

之后你会立即在右侧看到 Web 应用的运行效果。



```

1 import streamlit as st
2 import numpy as np
3 import pandas as pd
4
5 st.title("my first app")
6
7 x = st.slider("x")
8 y = x + 3
9

```

Made with Streamlit

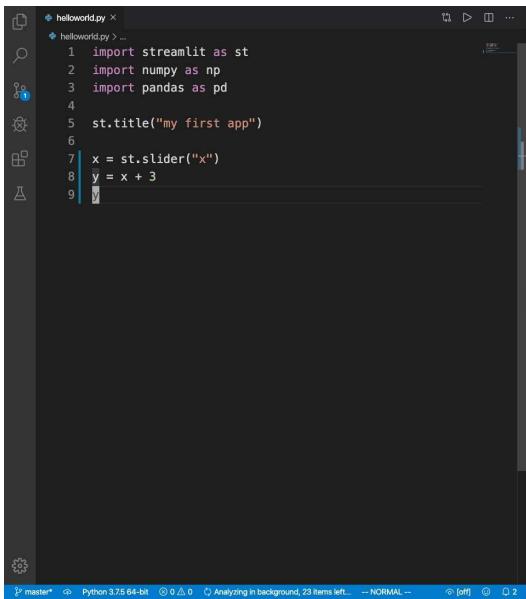
这里前几行语句，只是引入了几个软件包，然后设置了一下标题。

下面我们尝试点儿好玩儿的。

```

x = st.slider("x")
y = x + 3
y

```



```

1 import streamlit as st
2 import numpy as np
3 import pandas as pd
4
5 st.title("my first app")
6
7 x = st.slider("x")
8 y = x + 3
9

```

Made with Streamlit

这时候网页上出现了滑动条，告诉你这是 x 的取值。

我们定义了一个式子，让 y 总比 x 大 3，并且显示 y 。

你可以试试，在滑动条拖拽 x 的效果。



登录



Jan-14-2020-18-39-15.gif

y 值紧随你的拖动变化，对吧？

从这个简单的例子里，你可以看到 Streamlit 响应用户的输入和输出是多么方便。

而且应用上的控件一直运行。你输入的变化，会实时带来输出的变化。

下面我们还是步入正题吧。先注释掉刚才这三条语句，免得碍事儿。

```
# x = st.slider("x")
# y = x + 3
# y
```

我们定义一个函数：

```
@st.cache
def load_data():
    df = pd.read_csv("data.csv")
    df = df
    df.columns = ['event_type', 'time', 'county', 'lat', 'lon']
    return df
```

如果你学过那篇《[如何用 Python 和 Pandas 分析犯罪记录开放数据？](#)》，里面的其他语句你应该都认得。无非就是 Pandas 读入我们的 CSV 数据之后，取其中的 5 个列，包括：

- EVENT_TYPE : 事件类型；



登录

- COUNTY：事件发生位置所在郡名称；
- LAT：事件发生位置的纬度；
- LON：事件发生位置的经度

然后，我们把这几个列分别用小写的名称来命名。

值得一提的，是 `@st.cache`，这是一个新玩意儿。

它是什么呢？

这在 Python 里面，叫做装饰器（decorator）。其实这里没有什么魔法，它只是 streamlit 软件包里，一个预先定义的函数。

只不过这样写，相当于是你在自己的 `load_data()` 函数之外，又包裹了一个 `st.cache()` 函数的功能。每次执行的时候，`st.cache()` 都会参与进来。

`st.cache()` 这个函数做什么用呢？

那作用可太大了。

因为你每次更新代码，或者用户更新输入，整个儿 Python 脚本都相当于被重新执行了一遍。

而 `st.cache()` 装饰器可以告诉 Python：

查查看，我包裹的这个函数，内容或者输入改过没有？如果没有，就用已存储的上次调用结果好了，别再费事重新执行一遍了。

我们这里是读取一个外部文件。就这样一个 300MB 的文件，每次读起来也得花上近 10 秒钟。更别说是那些上 GB 规模，甚至更大的文件了。

所以，如果 Streamlit 能够帮助我们跳过一些无意义的重复操作，将节省大量的用户等待时长。

```

1 import streamlit as st
2 import numpy as np
3 import pandas as pd
4
5 st.title("my first app")
6
7 # x = st.slider("x")
8 # y = x + 3
9 # y
10
11 @st.cache
12 def load_data():
13     df = pd.read_csv("data.csv")
14     df = df[['EVENT_TYPE', 'CREATE_TIME', 'COUNTY', 'LA
15     df.columns = ['event_type', 'time', 'county', 'lat'
16     return df

```

Made with Streamlit

因为我们什么也没有输出啊。

下面我们让 Python 实际读数据，并且把读后的数据框前 5 行用列表形式（`st.table()`）展示给用户。

这一读数据不要紧，右上角会出现一个小人儿，做各种健身运动。

```

1 import streamlit as st
2 import numpy as np
3 import pandas as pd
4
5 st.title("my first app")
6
7 # x = st.slider("x")
8 # y = x + 3
9 # y
10
11 @st.cache
12 def load_data():
13     df = pd.read_csv("data.csv")
14     df = df[['EVENT_TYPE', 'CREATE_TIME', 'COUNTY', 'LA
15     df.columns = ['event_type', 'time', 'county', 'lat'
16     return df
17
18 df = load_data()
19 st.table(df.head())
20

```

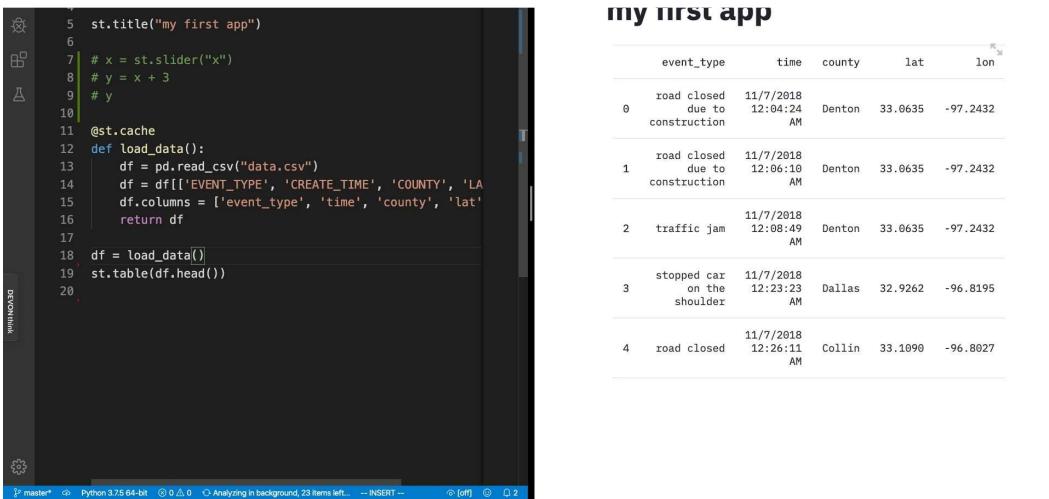
RUNNING... Stop

Running load_data().

Made with Streamlit

这就是告诉我们，程序在忙着呢。

忙完之后，这是结果：

The screenshot shows a Jupyter Notebook cell containing Python code for a Streamlit application. The code imports Streamlit and pandas, sets the title to "my first app", and defines a slider for 'x' and a variable 'y'. It uses a cache decorator for a function that reads a CSV file and filters it by event type. Finally, it displays the first few rows of the DataFrame. To the right, the resulting Streamlit dashboard titled "my first app" is shown, displaying a table with columns: event_type, time, county, lat, and lon. The table contains five rows of data.

```

5 st.title("my first app")
6
7 # x = st.slider("x")
8 # y = x + 3
9
10
11 @st.cache
12 def load_data():
13     df = pd.read_csv("data.csv")
14     df = df[['EVENT_TYPE', 'CREATE_TIME', 'COUNTY', 'LA']]
15     df.columns = ['event_type', 'time', 'county', 'lat']
16     return df
17
18 df = load_data()
19 st.table(df.head())
20

```

	event_type	time	county	lat	lon
0	road closed due to construction	11/7/2018 12:04:24 AM	Denton	33.0635	-97.2432
1	road closed due to construction	11/7/2018 12:06:10 AM	Denton	33.0635	-97.2432
2	traffic jam	11/7/2018 12:08:49 AM	Denton	33.0635	-97.2432
3	stopped car on the shoulder	11/7/2018 12:23:23 AM	Dallas	32.9262	-96.8195
4	road closed	11/7/2018 12:26:11 AM	Collin	33.1090	-96.8027

下面我们要让程序给用户选项，首先是选择观察哪一种事件类型。

```

event_list = df["event_type"].unique()

event_type = st.sidebar.selectbox(
    "Which kind of event do you want to explore?",
    event_list
)

```

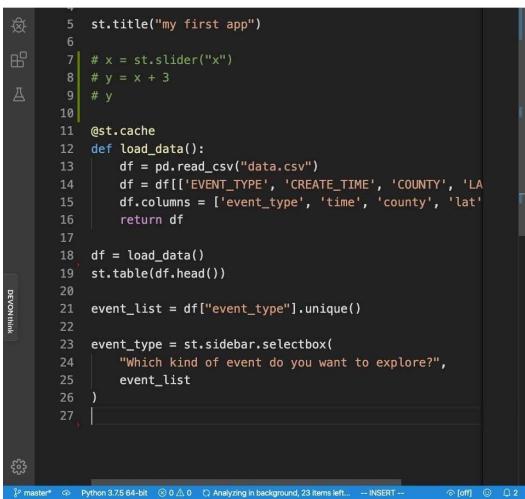
解释一下，第一句是在 event_type 里面寻找全部事件类型列表。

下面一段，采用了 st.sidebar.selectbox() 构造了一个左边栏里的下拉选择框。里面两个参数，第一个是显示给用户的提示语句，第二个，是选择列表内容。

问题是，我们存储了之后，好像什么也没有发生啊。



登录



```

5 st.title("my first app")
6
7 # x = st.slider("x")
8 # y = x + 3
9 # y
10
11 @st.cache
12 def load_data():
13     df = pd.read_csv("data.csv")
14     df = df[['EVENT_TYPE', 'CREATE_TIME', 'COUNTY', 'LA']]
15     df.columns = ['event_type', 'time', 'county', 'lat']
16     return df
17
18 df = load_data()
19 st.table(df.head())
20
21 event_list = df["event_type"].unique()
22
23 event_type = st.sidebar.selectbox(
24     "Which kind of event do you want to explore?",
25     event_list
26 )
27

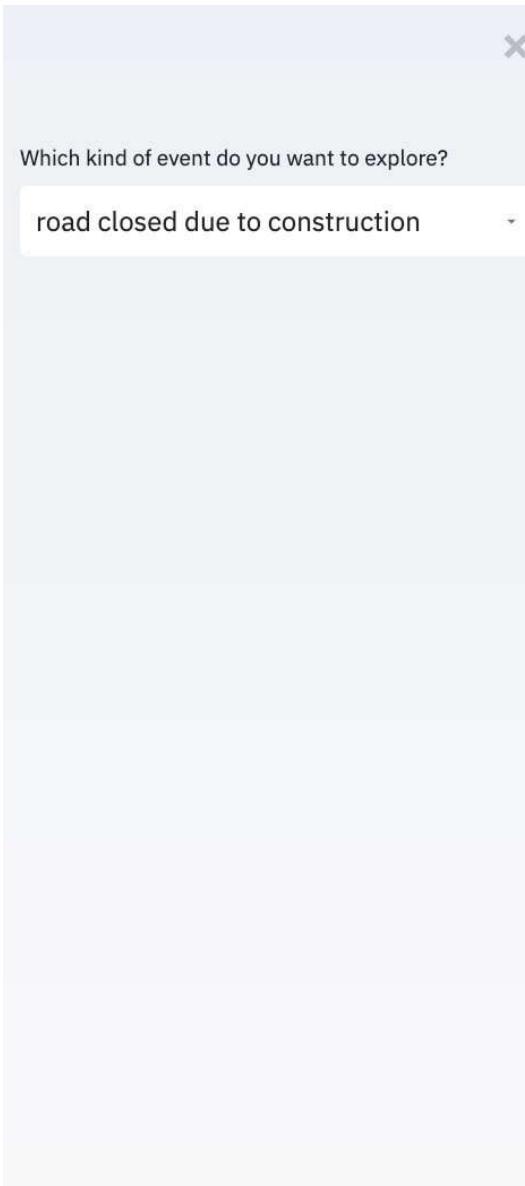
```

The screenshot shows a Jupyter Notebook cell containing Streamlit Python code. To the right, a running Streamlit application titled 'my first app' is displayed. The app features a sidebar with a dropdown menu set to 'road closed due to construction'. The main content area shows a table with five rows of event data:

	event_type	time	county	lat	lon
0	road closed due to construction	11/7/2018 12:04:24 AM	Denton	33.0635	-97.2432
1	road closed due to construction	11/7/2018 12:06:10 AM	Denton	33.0635	-97.2432
2	traffic jam	11/7/2018 12:08:49 AM	Denton	33.0635	-97.2432
3	stopped car on the shoulder	11/7/2018 12:23:23 AM	Dallas	32.9262	-96.8195
4	road closed	11/7/2018 12:26:11 AM	Collin	33.1090	-96.8027

没关系，看到上图里面红色标出的这个箭头没有？

点击它，选项就出现了。



The screenshot shows a Streamlit dropdown menu titled 'Which kind of event do you want to explore?'. The dropdown is currently set to 'road closed due to construction'. Below the dropdown, a table displays event data with columns: county, lat, and lon. The data is identical to the one shown in the Streamlit app above.

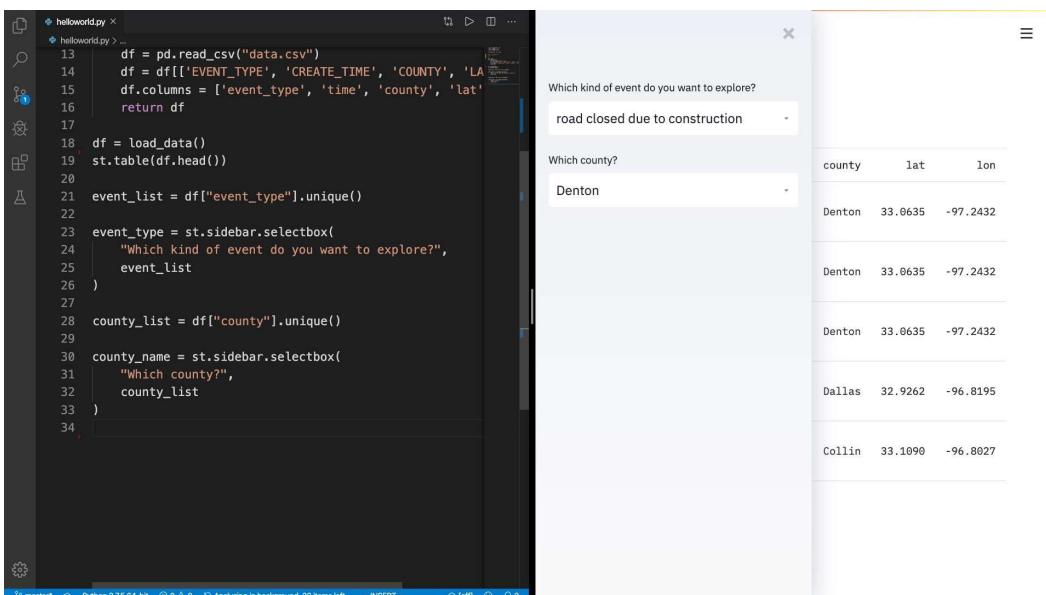
county	lat	lon
Denton	33.0635	-97.2432
Denton	33.0635	-97.2432
Denton	33.0635	-97.2432
Dallas	32.9262	-96.8195
Collin	33.1090	-96.8027



```
county_list = df["county"].unique()

county_name = st.sidebar.selectbox(
    "Which county?",
    county_list
)
```

这是效果：



然后，我们根据用户的输入做出反应，提示给用户经过他的选择，现在符合要求的行数还有多少。

```
part_df = df[(df["event_type"]==event_type) & (df['county']==county_name)]
st.write(f"根据你的筛选，数据包含{len(part_df)}行")
```



第一句里面用了个联合筛选，必须同时满足两个条件的数据，才会被保留在结果 part_df 中。

然后，我们把一个格式化后的字符串，用 st.write() 直接输出在网页上。



登录

	event_type	time	county	lat	lon
1	road closed due to construction	11/7/2018 12:04:24 AM	Denton	33.0405	-97.1432
2	road closed due to construction	11/7/2018 12:08:12 AM	Denton	33.0405	-97.1432
3	traffic jam	11/7/2018 12:08:12 AM	Denton	33.0405	-97.1432
4	stopped car on the shoulder	11/7/2018 12:21:12 AM	Dallas	32.7426	-96.8129

Jan-14-2020-19-04-42.gif

好了，下面可能是你最关心的一刻了。

老师，别卖关子了，那张标示了事件位置的叠层地图怎么画啊？一共都没多少行语句，你都讲了这么多了，怎么还没讲到？

请你输入下面这一行语句：

```
st.map(part_df)
```

然后保存。你就会看到下面的效果了。



登录

```

1 df = pd.read_csv("data.csv")
2 df = df[['EVENT_TYPE', 'CREATE_TIME', 'COUNTY', 'LAT', 'LONG']]
3 df.columns = ['event_type', 'time', 'county', 'lat', 'long']
4 return df
5
6 df = load_data()
7 st.table(df.head())
8
9 event_list = df["event_type"].unique()
10
11 event_type = st.sidebar.selectbox(
12     "Which kind of event do you want to explore?", 
13     event_list
14 )
15
16 county_list = df["county"].unique()
17
18 county_name = st.sidebar.selectbox(
19     "Which county?", 
20     county_list
21 )
22
23 part_df = df[(df["event_type"]==event_type) & (df['county']==county_name)]
24 st.write(f"根据你的筛选，数据包含{len(part_df)}行")
25
26 st.map(part_df)
27
28
29
30
31
32
33
34
35
36
37
38
39

```

The Streamlit application interface shows a sidebar with dropdown menus for 'event_type' (set to 'traffic jam') and 'county' (set to 'Dallas'). The main area displays a table titled 'my first app' with four rows of data:

	event_type	time	county
0	road closed due to construction	11/7/2018 12:04:24 AM	Denton 33
1	road closed due to construction	11/7/2018 12:06:10 AM	Denton 33
2	traffic jam	11/7/2018 12:08:49 AM	Denton 33
3	stopped car on the shoulder	11/7/2018 12:13:23 AM	Dallas 32
4	road closed	11/7/2018 12:26:13 AM	Collin 32

At the bottom, a note says '根据你的筛选，数据包含215966行'.

是不是很惊讶？

我第一次用的时候，也是这感觉。

在 HackNTX 2018 编程马拉松竞赛中，我曾经找不同的编程高手学了若干种地理信息可视化的工具。每一种都得花上很多时间学习演练。

没想到，短短一年的时间，这样的功能居然可以用一行代码就实现了。

还是集成在 Web 应用里，可以发布给全球用户与合作者，进行展示。

不是我不明白，这世界变化快啊。

部署

我知道，你又开始着急了。

老师，这么好的东西，我可不想在本地一个人玩儿。我也想把结果发布到网络上，让别人看到我的成果。快告诉我怎么办！

别急。



登录

虽然你写了半天，只是 Python 脚本。但是 Streamlit 已经把它转换成了一个动态的 Web 应用。

所以，只要是常见的 Web 应用发布平台，理论上你都可以用来部署你的交互式数据分析作品。

这些平台，常见的包括：

- EC2
- Glitch
- Heroku

这列表列下去就太多了。咱们这里只介绍 Heroku，也就是前文给你展示的，样例使用的部署平台。

这东西的好处，就是基础款**免费**。

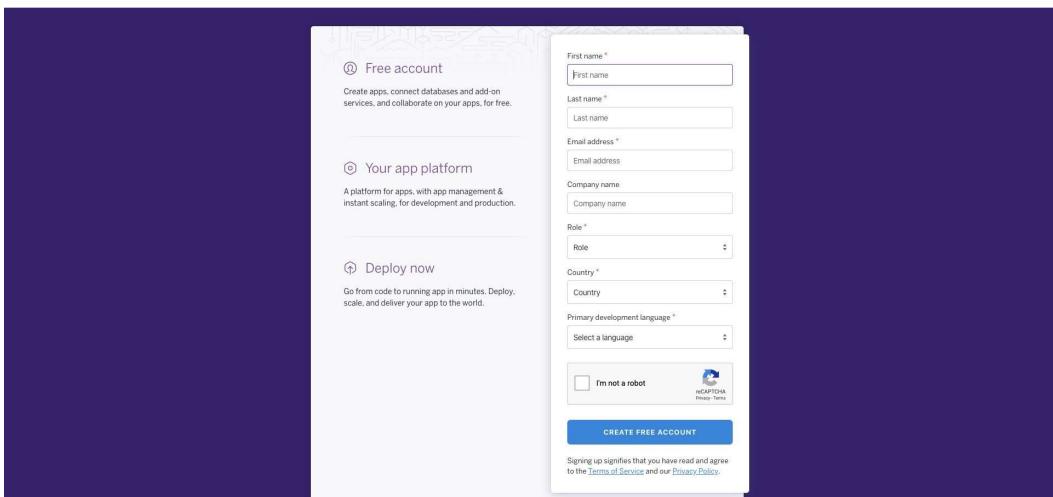
The screenshot shows the Heroku Pricing page. At the top, it says "Simple, flexible pricing" and "Plans for the needs of every app." Below this, there's a section for "Dynos" with a sub-section for "Free". The "Free" plan is described as ideal for experimenting with cloud applications in a limited sandbox. It includes "CORE PLATFORM FEATURES", "NEVER SLEEPS", "FREE SSL & AUTOMATED CERTIFICATE MANAGEMENT FOR CUSTOM DOMAINS", "APPLICATION METRICS", and "MULTIPLE WORKERS FOR MORE POWERFUL APPS". It also mentions "512 MB RAM | 10 Process Types" and costs "\$0 per dyno/month". To the right, there are sections for "Hobby", "Standard (1X | 2X)", and "Performance (M | L)". The "Standard" plan is described as perfect for small scale personal projects and hobby apps. It includes "ALL HOBBY FEATURES +", "SIMPLE HORIZONTAL SCALABILITY", "THRESHOLD ALERTS", and "PREBOOT". It also mentions "LANGUAGE RUNTIME METRICS" and "512MB OR 1GB RAM". The "Performance" plan is described as supporting performance when it's most critical for your larger scale, high traffic apps. It includes "ALL STANDARD FEATURES +", "DEDICATED", "AUTOSCALING", and "2.5GB OR 14GB RAM". A note at the bottom says "Process Types" and lists "✓ \$25 - \$500 per dyno/month prorated to the second".

对咱们今天的教程来说，基础款就足够了。

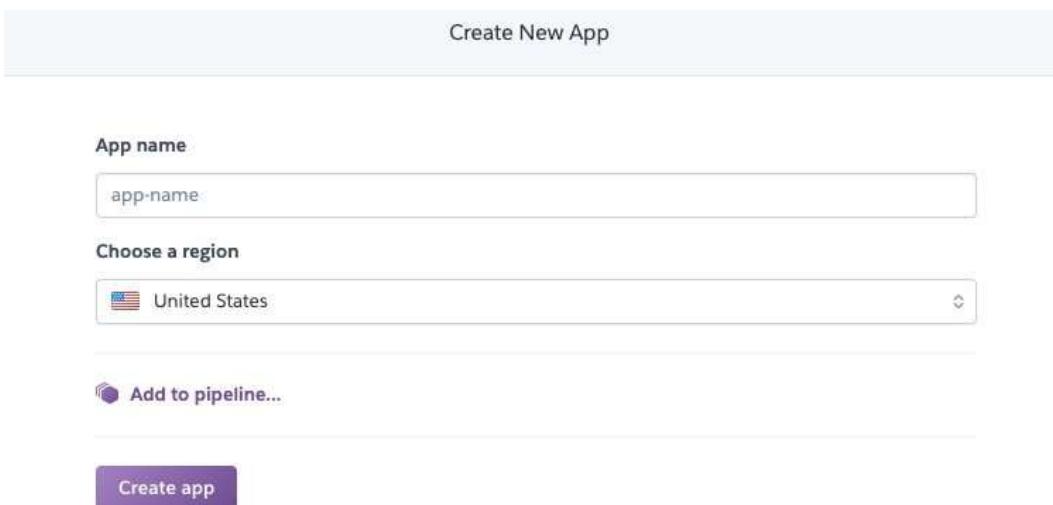
你需要先到 Heroku 平台注册一个账号。



登录



之后，注册一个应用。



我这里起的名字，叫做 helloworld-streamlit。

你可以根据自己的喜好，起名称。

之后我们就要部署了。



登录

The screenshot shows the Heroku deployment interface. At the top, there are sections for "Add this app to a pipeline" and "Add this app to a stage in a pipeline to enable additional features". Below that is a "Deployment method" section with three options: "Heroku Git" (selected), "GitHub" (disabled), and "Container Registry". Under "Heroku Git", there are instructions to "Deploy using Heroku Git" and "Install the Heroku CLI". It also includes command-line examples for "Clone the repository" and "Deploy your changes".

部署的步骤，在上图中，你可以参考。

注意，上图中，右上角的 Open App 按钮，就是你的应用链接地址，你可以把它记下来。

首先你需要准备一些配置文件。

全部的配置文件，我都给你展示在了前文介绍过的这个 [github 项目](#) 中，你可以下载回来复用。

The screenshot shows a GitHub repository page. The repository name is "wshuyi / demo-helloworld-streamlit". The page displays basic statistics: 1 commit, 1 branch, 0 packages, 0 releases, and 1 contributor. It shows a list of files including "wshuyi.ini", "Profile", "helloworld.py", "requirements.txt", and "setup.sh". A green button at the bottom right says "Add a README".

这里需要说明的是，几个不同配置文件的用途。

`setup.sh` 做一些初始设置，设定一些参数。

1 contributor

13 lines (11 sloc) | 217 Bytes

```

1 mkdir -p ~/.streamlit/
2
3 echo "\\
4 [general]\n\
5 email = yourmail@email.com\n\
6 " > ~/.streamlit/credentials.toml
7
8 echo "\\
9 [server]\n\
10 headless = true\n\
11 enableCORS=false\n\
12 port = $PORT\n\
13 " > ~/.streamlit/config.toml

```

Raw Blame History

注意你将来用的时候，需要把其中标红的部分，替换成自己注册 heroku 时候的邮箱。

`requirements.txt` 告诉机器，需要安装哪些 Python 依赖包。

Branch: master ▾ demo-helloworld-streamlit / requirements.txt

wshuyi init 5cb914a 3 minutes ago

1 contributor

3 lines (3 sloc) | 47 Bytes

```

1 numpy==1.15.0
2 streamlit==0.52.2
3 pandas==0.23.4

```

Raw Blame History

显然，教程这里需要的依赖包不多。

`Procfile` 是远端服务器上，Web 应用启动的时候，需要调用的脚本。其实里面只有一行。

Branch: master ▾ demo-helloworld-streamlit / Procfile

wshuyi init 5cb914a 4 minutes ago

1 contributor

1 lines (1 sloc) | 47 Bytes

```

1 web: sh setup.sh && streamlit run helloworld.py

```

Raw Blame History

请你下载，或者自行编辑上述 3 个文件后，与你的 Python 文件放在一个文件夹下面。

之后，请你到[这里](#)下载 heroku cli package。



登录

! The Heroku CLI requires **Git**, the popular version control system. If you don't already have Git installed, complete the following before installing the CLI:

- [Git installation](#)
- [First-time Git setup](#)

macOS

[Download the installer](#)

Also available via Homebrew:

```
$ brew tap heroku/brew && brew install heroku
```

Windows

Download the appropriate installer for your Windows installation:

[64-bit installer](#)

[32-bit installer](#)

下载后，根据提示安装即可。

进入终端。用 cd 命令切换到你的工作文件夹，也就是包含了你的 Python 脚本的目录。

输入：

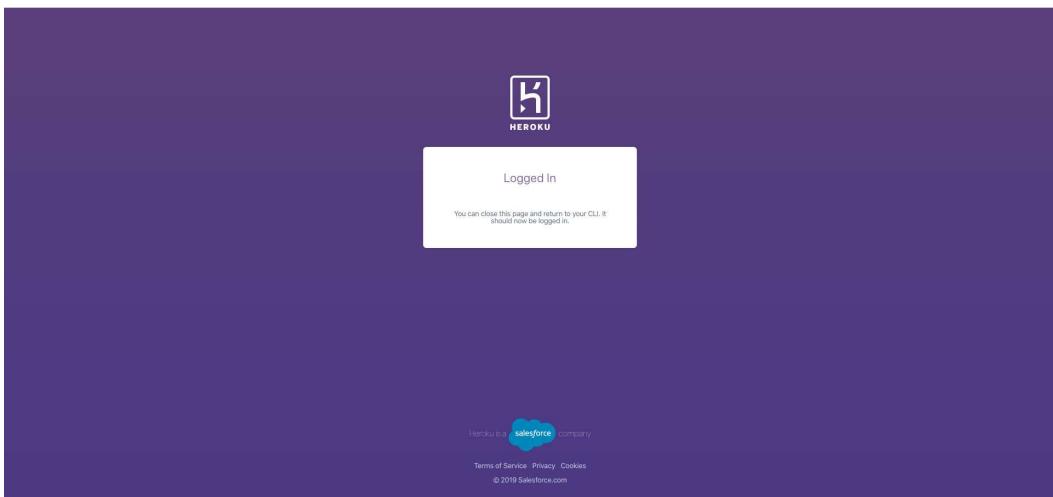
```
heroku login
```

因为你已经在 `setup.sh` 中指定了自己的邮箱，所以这里会尝试直接用它来登录。

这时按任意键，会跳出一个浏览器窗口。



登录



在浏览器中，点击确认即可登录。

```
➜ try-streamlit git:(master) ✘ heroku login ±[●][master]
heroku: Press any key to open up the browser to login or q to exit:
Opening browser to https://cli-auth.herokuapp.com/auth/cli/browser/a86ece04-4583-40
3b-8693-905505273af1
Logging in... done
Logged in as [REDACTED]@[REDACTED].com
```

看到上面的提示，证明登录成功了。

下面我们来设置 git，这是推送我们文件和更新改动的途径。

在终端下执行：

```
git init
```

之后设置一下与远端的 heroku 服务器的连接：

```
heroku git:remote -a helloworld-streamlit
```

若是看到下图，证明成功了：

```
➜ try-streamlit git:(master) ✘ heroku git:remote -a helloworld-streamlit
  > Warning: heroku update available from 7.35.0 to 7.35.1.
set git remote heroku to https://git.heroku.com/helloworld-streamlit.git
➜ try-streamlit git:(master) ✘ ±[●][master]
```

然后执行：

```
git commit -m "init"
```

```
↳ try-streamlit git:(master) ✘ git add .
↳ try-streamlit git:(master) ✘ git commit -m "init"
[master (root-commit) b0c586b] init
 6 files changed, 937603 insertions(+)
 create mode 100644 .vscode/settings.json
 create mode 100644 Procfile
 create mode 100644 data.csv
 create mode 100644 helloworld.py
 create mode 100644 requirements.txt
 create mode 100644 setup.sh
```

再执行：

```
git push heroku master
```

这样就可以把全部内容推送到 heroku 了。

推送的第一步，是上传文件。

```
↳ try-streamlit git:(master) git push heroku master
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 12 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (9/9), 43.87 MiB | 737.00 KiB/s, done.
Total 9 (delta 0), reused 0 (delta 0)
```

Heroku 发现咱们推送的是一个 Python App，所以自动执行许多安装设置工作。



三

登录

```
0ce3cc2abe92efd3cd61d764bee6ccdf1b667a1fb566f45dc249953/Pillow-7.0.0-cp36-cp36m-
manylinux1_x86_64.whl (2.1MB)
remote:     Collecting protobuf>=3.6.0 (from streamlit==0.52.2->-r /tmp/build_
_12579caa6fba301d7c82503ac4029c00/requirements.txt (line 2))
remote:         Downloading https://files.pythonhosted.org/packages/ca/ac/838c8
c8a5f33a58132dd2ad2a30329f6ae1614a9f56ffb79eaaf71a9d156/protobuf-3.11.2-cp36-cp3
6m-manylinux1_x86_64.whl (1.3MB)
remote:     Collecting base58 (from streamlit==0.52.2->-r /tmp/build_12579caa
6fba301d7c82503ac4029c00/requirements.txt (line 2))
remote:         Downloading https://files.pythonhosted.org/packages/09/b2/21ac9
591f055acc145afead895edeb73bb69d95cf366fc5c2233f2434cb/base58-1.0.3-py3-none-an
y.whl
remote:     Collecting tzlocal (from streamlit==0.52.2->-r /tmp/build_12579ca
a6fba301d7c82503ac4029c00/requirements.txt (line 2))
remote:         Downloading https://files.pythonhosted.org/packages/ef/99/53bd1
ac9349262f59c1c421d8fcc2559ae8a5effed9202684756b648d33/tzlocal-2.0.0-py2.py3-no
ne-any.whl
remote:     Collecting altair>=3.2.0 (from streamlit==0.52.2->-r /tmp/build_1
2579caa6fba301d7c82503ac4029c00/requirements.txt (line 2))
remote:         Downloading https://files.pythonhosted.org/packages/67/fe/2584a
b143719f073abb4884049a1ef2e5daf277d9c5a3a47df03cd7ad4ab/altair-4.0.0-py2.py3-non
e-any.whl (709kB)
```

这些安装和配置做完后，会出现下面这样的提示。



```
wsy@wangshuyideMacMini: ~/Dropbox/var/wsywork/learn/demo-workshops/try-streamlit
remote:     Running setup.py install for tornado: finished with status 'd
one'
remote:     Successfully installed MarkupSafe-1.1.1 PyYAML-5.3 altair-4.0.0 a
rgh-0.26.2 astor-0.8.1 attrs-19.3.0 base58-1.0.3 blinker-1.4 boto3-1.11.1 botoco
re-1.14.1 certifi-2019.11.28 chardet-3.0.4 click-7.0 decorator-4.4.1 docutils-0.
15.2 entrypoints-0.3 enum-compat-0.0.3 future-0.18.2 idna-2.8 importlib-metadata
-1.4.0 jinja2-2.10.3 jmespath-0.9.4 jsonschema-3.2.0 more-itertools-8.1.0 numpy-
1.15.0 pandas-0.23.4 pathtools-0.1.2 pillow-7.0.0 protobuf-3.11.2 pyrsistent-0.1
5.7 python-dateutil-2.8.0 pytz-2019.3 requests-2.22.0 s3transfer-0.3.0 six-1.13.
0 streamlit-0.52.2 toml-0.10.0 toolz-0.10.0 tornado-5.1.1 tzlocal-2.0.0 urllib3-
1.25.7 validators-0.14.1 watchdog-0.9.0 zipp-1.0.0
remote:
remote: -----> Discovering process types
remote:     Procfile declares types -> web
remote:
remote: -----> Compressing...
remote:     Done: 136.9M
remote: -----> Launching...
remote:     Released v3
remote:     https://helloworld-streamlit.herokuapp.com/ deployed to Heroku
remote:
remote: Verifying deploy... done.
To https://git.heroku.com/helloworld-streamlit.git
 * [new branch]      master -> master
→ try-streamlit git:(master) ±[master]
```

到这里，你的 Web 应用部署就搞定了。

回到浏览器里，用下图中标红的这个按钮开启你自己的应用吧。



登录

Add this app to a pipeline
Create a new pipeline or choose an existing one and add this app to a stage in it.

Pipelines let you connect multiple apps together and promote code between them.
[Learn more](#)

Pipelines connected to GitHub can enable review apps, and create apps for new pull requests.
[Learn more](#)

Choose a pipeline

Deployment method

Heroku Git Use Heroku CLI GitHub Connect to GitHub Container Registry Use Heroku CLI

Deploy using Heroku Git
Use git in the command line or a GUI tool to deploy this app.

Install the Heroku CLI
Download and install the [Heroku CLI](#)
If you haven't already, log in to your Heroku account and follow the prompts to create a new SSH public key.
`$ heroku login`

Clone the repository
Use Git to clone `helloworld-streamlit`'s source code to your local machine.
`$ heroku git:clone -a helloworld-streamlit
$ cd helloworld-streamlit`

Deploy your changes
Make some changes to the code you just cloned and deploy them to Heroku using Git.
`$ git add .
$ git commit -am "make it better"
$ git push heroku master`

怎么样？

很有成就感吧？

思考

尝试过之后，你应该不难发现，Streamlit 给你带来了什么。

如果你学过 Javascript 和 Flask, Django 等 Web 应用开发技术，Streamlit 可以加快你的 Web 应用开发与测试进程。

如果你还没有学过上述技术， Streamlit 就可以给你赋能，让你一下子有了把数据分析结果**变成产品**的能力。

给你讲点儿更激进的。

有人已经希望能用它替代掉 Flask 用于产品发布了。



Dr. Bruce H. Cottman [Follow](#)

Dec 23, 2019 · 5 min read ★



ok, the title is a little dramatic. How about ... Streamlit will replace a lot of Flask-based applications.



Can Flask be on the endangered technology list?

还有人说，将来写技术文档，也应该充分使用 Streamlit。

甚至，还把它比作了数据科学界的 iPhone。

The best way to think about the arrival of Streamlit is to think of the **arrival of the Iphone** being announced as **the smartest way to place calls and send text messages**. Well yes :-). But now we can see that it's a powerfull compute engine and an ecosystem of apps that has changed peoples lifes.

这里，它是借喻 iPhone 开启智能手机时代，说明 Streamlit 的划时代性。

我不希望你也变得如此激进。

因为这里提到的每一种功用，现在还都有非常专业的工具做的更好，而且新的工具也在不断涌现。

比如说，我们在多个教程中一直使用 Jupyter Notebook。



```

Name           Last Modified
imgs          20 minutes ago
demo.ipynb    seconds ago
 README.md     29 minutes ago

In [1]: from fastai import *
from fastai.vision import *
from fastai.core import *

In [2]: path = Path('imgs')

In [3]: data = ImageDataBunch.from_folder(path, test='test', ds_tfms=get_transforms(), size=224)

In [4]: data.show_batch(rows=3, figsize=(10,10))

In [5]: learn = ConvLearner(data, models.resnet34, metrics=accuracy)

In [6]: learn.fit_one_cycle(1)

In [7]: pred,y = learn.get_preds()
interp = ClassificationInterpretation(data, preds, y, loss_class=nn.CrossEntropyLoss)

In [8]: interp.plot_top_losses(5, figsize=(10,10))

In [9]: interp.plot_confusion_matrix()

In [10]: pred,y = learn.get_preds(is_test=True)

In [11]: data.test_dl.dataset.ds.x

In [12]: pred

In [13]: np.argmax(preds, axis=1)

In [14]: learn.unfreeze()
learn.fit_one_cycle(3, slice(1e-5,3e-4))

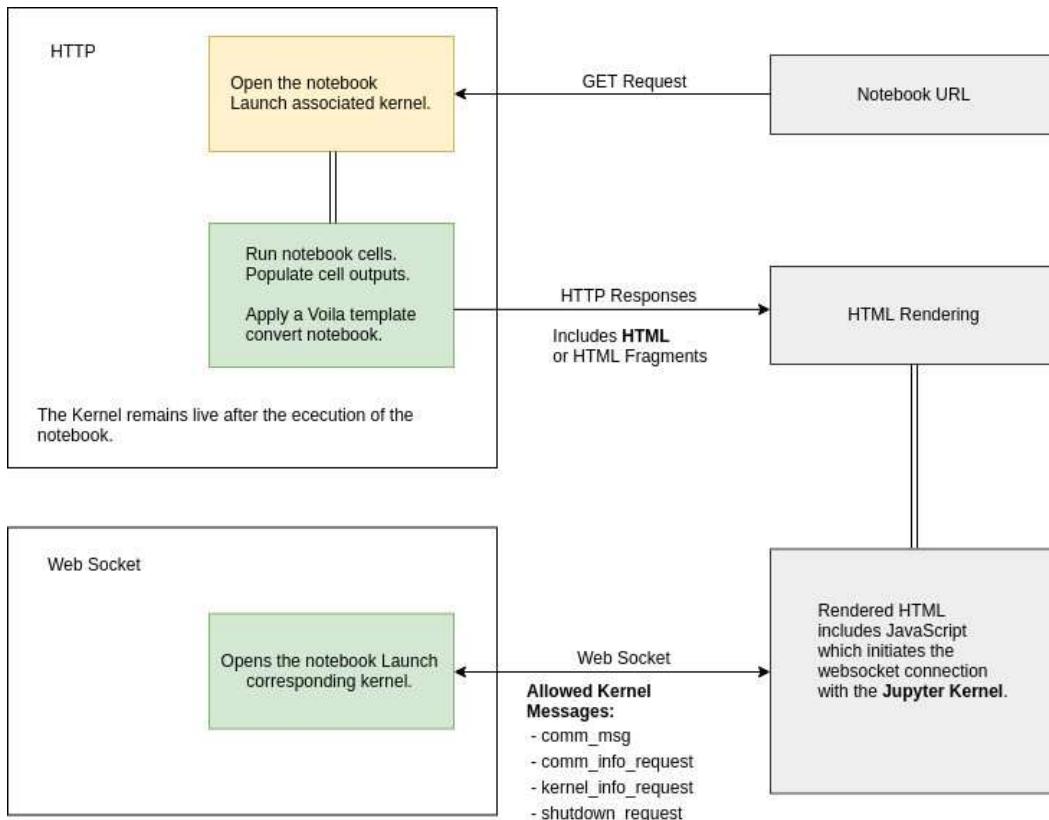
In [15]: pred,y = learn.get_preds(is_test=True)

In [16]: data.test_dl.dataset.ds.x

In [17]: np.argmax(preds, axis=1)

```

现在凭借 Voila 扩展的加持，你也可以很轻松地把 Jupyter Notebook 变成 Web app，而且可以免费运行在 [mybinder](#) 上面。



但是，你可以看到，一个新的工具，以一种简单，而不是更繁复的办法，解决一个功能痛点，是一件多么令人欣喜的事儿。

看了这篇文章，可能会给你一种误解，似乎 JavaScript 为代表的前端编程技术，再也不需要学了。

其实不是这样的。



登录

用 Streamlit 这样的方法，他们只是开发出了一个原型。

要是想打造精品，就必须精细调控很多细节。

这时候， Javascript 是绕不过去的。

如果你精通 Javascript，那你潜在的合作对象一下子就多了起来，你掌握的这门技术，也就有了更大的价值。

还记得吗？我不止一次给你强调过，**协作网络更重要**。忘了的话，记得复习《学 Python，能提升你的竞争力吗？》。

这就好像印刷术的发明，不是让会写字这件事儿变得失去价值，而是全社会都增大了对好作品的渴求。深刻的思考，加上有效的文字表达，会让你生存得更好。

当然，如果你不希望精通写作技艺，只是想做一个抄书匠糊口。那么印刷术就可能会替代你的工作，结果就不那么美妙了。

小结

本文我为你介绍了 Streamlit，它可以让你用 Python 脚本编写简洁实用的交互式 Web 应用。

通过学习本文，希望你掌握了以下知识点：

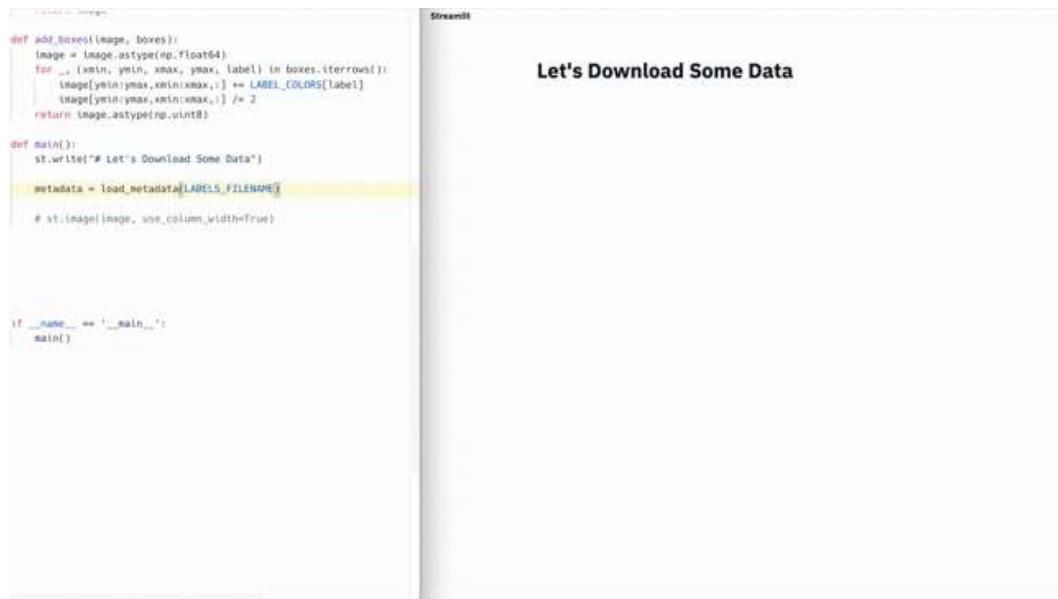
- 现在你有了一种选择，仅用纯 Python 做一个完整的交互式数据分析产品出来；
- 如何在读取数据等常用重复操作中，使用 `st.cache` 装饰器提升速度与效率；
- 如何使用滑动条、下拉框等基本组件；
- 如何在网页上输出文字、表格和图像；



登录

HITOKOTO

咱们是以数据分析和可视化为例，进行了讲解。而且为了讲解的清晰，我们只介绍了 Streamlit 可实现功能的一小部分。但请注意，即便是目前，Streamlit 能帮你达成的目标，也远远不止于此。



希望你能够举一反三，用 Streamlit 做出令人惊艳的作品。也欢迎你把作品的链接在留言区分享给咱们的同学。

祝编程愉快！

读过本文，如果觉得有收获，请[点赞](#)。

要读更多的文章，微信关注我的公众号 [“玉树芝兰” \(nkwangshuyi\)](#)。别忘了[加星标](#)，以免错过新推送提示。

如果本文对你身边的亲友有帮助，也欢迎你把本文通过微博或朋友圈分享给他们。

延伸阅读

你可能也会对以下话题感兴趣。点击链接就可以查看。

- [如何高效学 Python ?](#)



登录

- 论文读不懂怎么办？
- 如何不写 SQL，探索和分析数据库？

题图：Photo by Luke Chesser on Unsplash

效率有心得

⚡ 40



Gwanbou、LaDiDah、星桥liang 等 40 人为本文章充电



玉树芝兰 🔒

王树义。大学教师，终身学习者。稍微懂一点儿写作、演讲、Python和机器学习。欢迎关注我的公众号“玉树芝...

关注

Microsoft 365 工具升值包
触手可及的生产力

全部评论(11)

热门排序 ⚡



请在登录后评论...



Jackiexiao

感谢推荐！正愁学flask和django要花比较多的时间，但自己只是想做个简单的demo。可以用docker快速部署streamlit在阿里云ECS服务器上，最低配500元左右1年吧，当做一个远程linux服务器用了

评论 2 赞 1 投诉 0 举报 2020 年 09 月 15 日



登录

写了一个在阿里云上docker+streamlit的教程<https://zhuanlan.zhihu.com/p/245294921>

评论 0 赞同 0 举报 2020 年 09 月 15 日

Jackiexiao 回复 Jackiexiao

知乎审核要蛮久的...用这个

[https://jackiegeek.gitee.io/blog/%E6%8A%80%E6%9C%AF/%E4%BD%BF%E7%94%A8docker%E5%92%8Cstreamlit%E9%98%BF%E9%87%8C%E4%BA%91%E6%9C%8D%E5%8A%A1%E5%99%...展开](https://jackiegeek.gitee.io/blog/%E6%8A%80%E6%9C%AF/%E4%BD%BF%E7%94%A8docker%E5%92%8Cstreamlit%E9%98%BF%E9%87%8C%E4%BA%91%E6%9C%8D%E5%8A%A1%E5%99%...)

评论 0 赞同 0 举报 2020 年 09 月 15 日

少数派_962691

<https://helloworld-streamlit.herokuapp.com/>

昨天还能访问，今天访问不了，报错信息：

An error occurred in the application and your page could not be served. If you are the application...展开

评论 1 赞同 1 举报 2020 年 03 月 06 日

玉树芝兰 

我没给heroku付费，所以人家不保证服务质量与稳定性

评论 0 赞同 0 举报 2020 年 03 月 06 日

少数派_1034497

老师您好，我在heroku上部署streamlit，在执行git push heroku master 的时候报错error: RPC failed; curl 56 OpenSSL SSL_read: No error。我按网上的教程执行git config http.sslVerify "false"，还是报同样的错误。

所以想请教您有没有什么解决办法

评论 0 赞同 0 举报 2021 年 04 月 15 日



登录

如何用 Python 和 Streamlit 做交互式数据分析产品？ - 少数派

评论 0 赞 0 投诉 举报 2020 年 07 月 29 日

Flyingduck

太感谢作者了，我正在愁 如何把我用pandas分析的结果呈现出来，然而自己不会开发软件，单纯把python脚本发送给别人，还要别人安装各种依赖包和环境，有了这东西，仿佛发现了新大陆

评论 0 赞 0 投诉 举报 2020 年 06 月 17 日

少数派_955333

看着就有用，我按照做了一遍，卡在部署那里了，我push到远程以后，app无法打开，说有 Application error, 我总感觉是配置文件的问题，能不能给点思路，多谢了

评论 0 赞 0 投诉 举报 2020 年 02 月 04 日

少数派_948667

你在一个学校呀？

评论 1 赞 0 投诉 举报 2020 年 01 月 15 日

玉树芝兰

我在天津师范大学教书

评论 1 赞 0 投诉 举报 2020 年 01 月 15 日

没有更多评论了哦

推荐阅读



登录

的 10 款应用

huhuhang

79

写给 Mac 新手的入门 App 指南

Vanilla

2040

哪些订阅服务支持家庭共享？来看看我们为你整理的这些

ElijahLee

120

MoneyWiz 3.1 更新详解：全新的 UI 设计终于来了

李大超人Leo

119



三

登录

势，一次过全都分享给你

挨石

249

年度征文 | 买了 10 多款机械键 盘之后，我想聊聊如何找到「…

北鶲

328

[下载App](#) [联系我们](#)[商务合作](#) [成为作者](#)[关于我们](#) [用户协议](#)[常见问题](#)