Relax and Code on. Photo by Cody Black on Unsplash

# How to write Web apps using simple Python for Data Scientists?

Convert your Data Science Projects into cool apps easily without knowing any web frameworks

Rahul Agarwal
Oct 13, 2019 · 9 min read ★

A Machine Learning project is never really complete if we don't have a good way to showcase it.

and Dash, a good data scientist needs to have a fair bit of knowledge of web frameworks to get along.

And Web frameworks are hard to learn. I still get confused in all that HTML, CSS, and Javascript with all the hit and trials, for something seemingly simple to do.

Not to mention the many ways to do the same thing, making it confusing for us data science folks for whom web development is a secondary skill.

***So, are we doomed to learn web frameworks? Or to call our developer friend for silly doubts in the middle of the night?***

This is where StreamLit comes in and delivers on its promise to create web apps just using Python.

> Zen of Python: Simple is better than complex and Streamlit makes it dead simple to create apps.

***This post is about understanding how to create apps that support data science projects using Streamlit.***

To understand more about the architecture and the thought process that led to streamlit, have a look at this excellent post by one of the original developers/founder Adrien Treuille.

If you want to deploy a streamlit app on Amazon ec2 check out my next post.

. . .

## Installation

Installation is as simple as running the command:

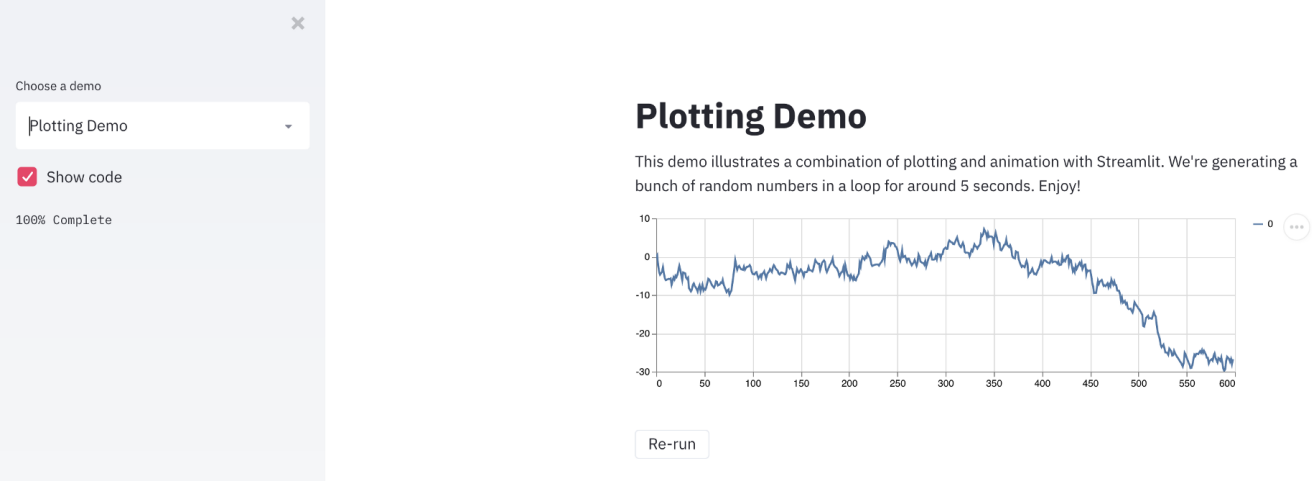
```
pip install streamlit
```

To see if our installation is successful, we can just run:

This should show you a screen that says:

```
~/web/new_blog/post_files/streamlit  streamlit hello

  👋 Welcome to Streamlit!

  If you are one of our development partners or are interested in
  getting personal technical support, please enter your email address
  below. Otherwise, you may leave the field blank.

  Email:

  Telemetry: As an open source project, we collect usage statistics.
  We cannot see and do not store information contained in Streamlit apps.

  If you'd like to opt out, add the following to ~/.streamlit/config.toml,
  creating that file if necessary:

    [browser]
    gatherUsageStats = false


  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8501
  Network URL: http://192.168.0.103:8501
```

You can go to the local URL: `localhost:8501` in your browser to see a Streamlit app in action. The developers have provided some cool demos that you can play with. Do take your time and feel the power of the tool before coming back.



# Streamlit Hello World

Streamlit aims to make app development easy using simple Python.

So let us write a simple app to see if it delivers on that promise.

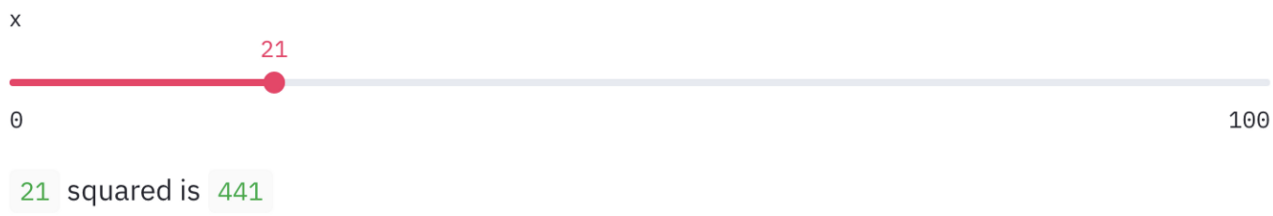
Here I start with a simple app which we will call the Hello World of streamlit. Just paste the code given below in a file named `helloworld.py`

```
import streamlit as st

x = st.slider('x')
st.write(x, 'squared is', x * x)
```

And, on the terminal run:

```
streamlit run helloworld.py
```

And voila, you should be able to see a simple app in action in your browser at `localhost:8501` that allows you to move a slider and gives the result.




A Simple slider widget app

It was pretty easy. In the above app, we used two features from Streamlit:

- the `st.slider` widget that we can slide to change the output of the web app.
- and the versatile `st.write` command. I am amazed at how it can write anything from charts, dataframes, and simple text. More on this later.

***Important: Remember that every time we change the widget value, the whole app***

. . .

## Streamlit Widgets

Widgets provide us a way to control our app. The best place to read about the widgets is the API reference documentation itself but I will describe some most prominent ones that you might end up using.

### 1. Slider

```
streamlit.slider(label, min_value=None, max_value=None, value=None,
step=None, format=None)
```

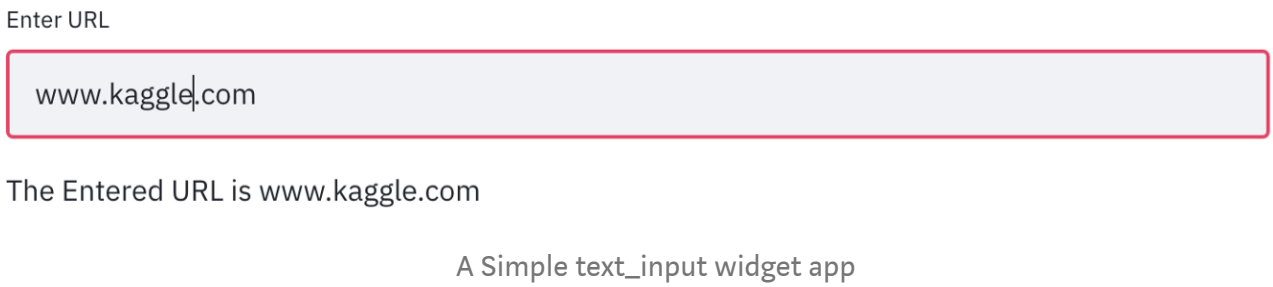
We already saw `st.slider` in action above. It can be used with min_value,max_value, and step for getting inputs in a range.

### 2. Text Input

The simplest way to get user input be it some URL input or some text input for sentiment analysis. It just needs a single label for naming the textbox.

```
import streamlit as st

url = st.text_input('Enter URL')
st.write('The Entered URL is', url)
```

This is how the app looks:




A Simple text_input widget app

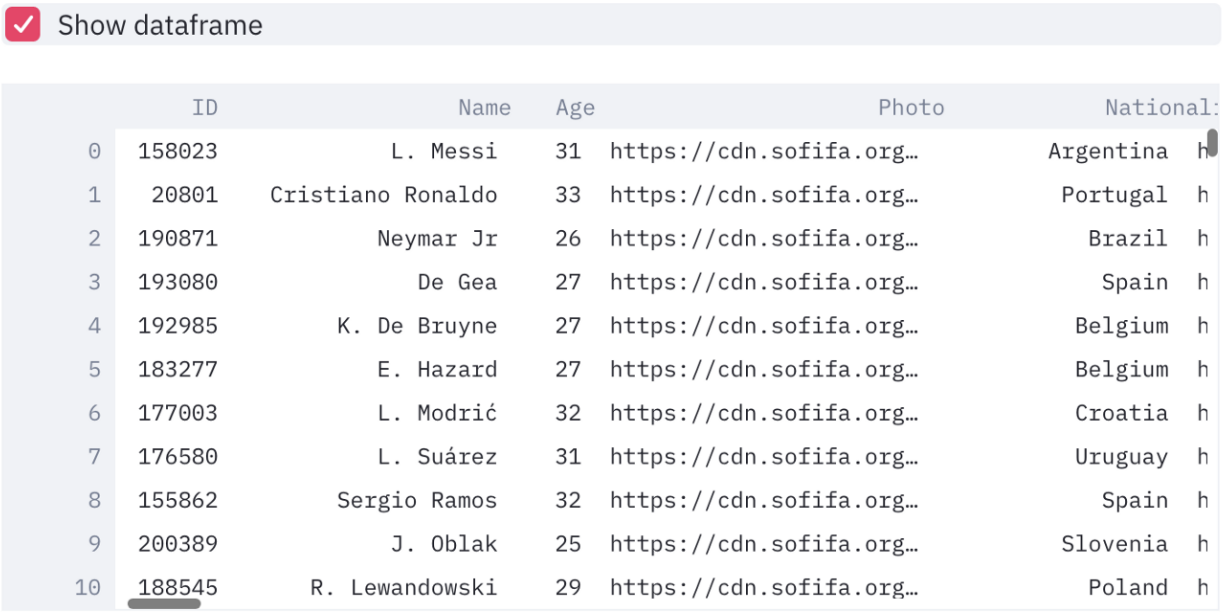
**Tip:** You can just change the file `helloworld.py` and refresh the browser. The way I

browser side by side.

## 3. Checkbox

One use case for checkboxes is to hide or show/hide a specific section in an app. Another could be setting up a boolean value in the parameters for a function. `st.checkbox()` takes a single argument, which is the widget label. In this app, the checkbox is used to toggle a conditional statement.

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

df = pd.read_csv("football_data.csv")
if st.checkbox('Show dataframe'):
    st.write(df)
```

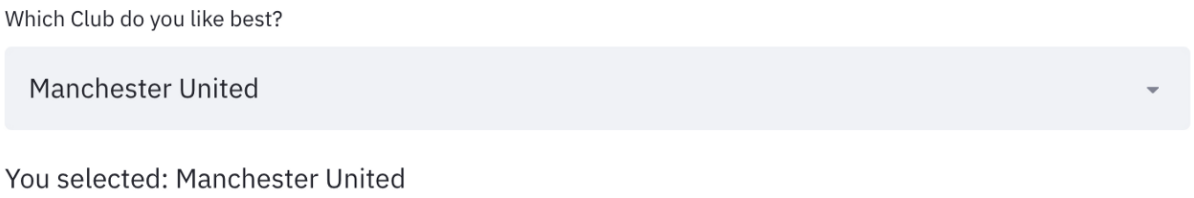



A Simple checkbox widget app

## 4. SelectBox

We can use `st.selectbox` to choose from a series or a list. Normally a use case is to use it as a simple dropdown to select values from a list.

```
df = pd.read_csv("football_data.csv")

option = st.selectbox(
    'Which Club do you like best?',
     df['Club'].unique())

'You selected: ', option
```

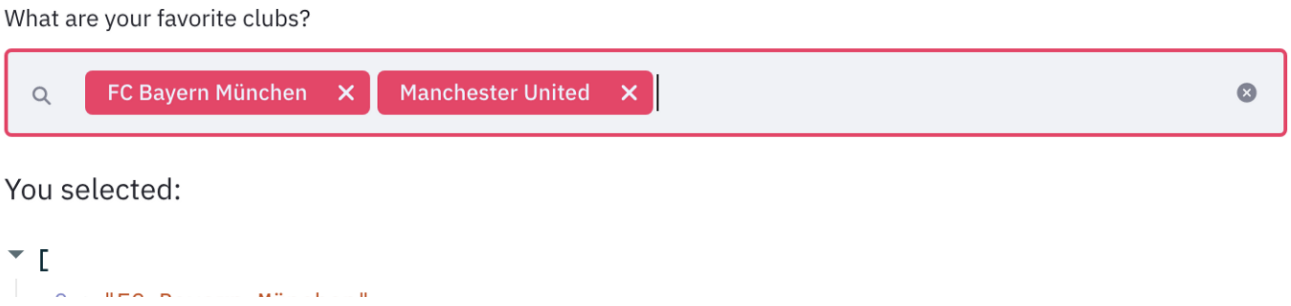



A Simple dropdown/selectbox widget app

## 5. MultiSelect

We can also use multiple values from a dropdown. Here we use `st.multiselect` to get multiple values as a list in the variable `options`

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

df = pd.read_csv("football_data.csv")

options = st.multiselect(
 'What are your favorite clubs?', df['Club'].unique())

st.write('You selected:', options)
```



A Simple multiselect widget app

. . .

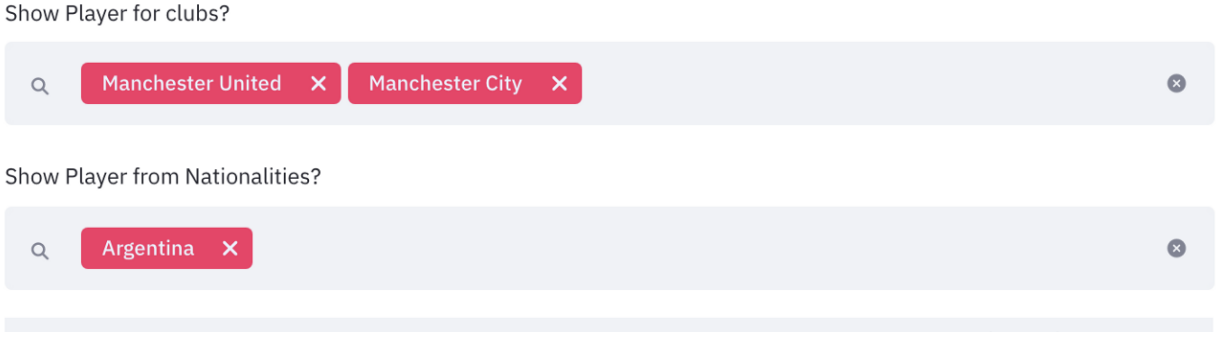
## Creating Our Simple App Step by Step

So much for understanding the important widgets. Now, we are going to create a simple app using multiple widgets at once.

To start simple, we will try to visualize our football data using streamlit. It is pretty much simple to do this with the help of the above widgets.

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

df = pd.read_csv("football_data.csv")

clubs = st.multiselect('Show Player for clubs?',
df['Club'].unique())

nationalities = st.multiselect('Show Player from Nationalities?',
df['Nationality'].unique())

# Filter dataframe
new_df = df[(df['Club'].isin(clubs)) &
(df['Nationality'].isin(nationalities))]

# write dataframe to screen
st.write(new_df)
```

Our simple app looks like:



| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 526 | 26 | 173373 | S. Romero | 31 | https://cdn.sofifa.org… | | Argentina | https: |
| 629 | 29 | 201862 | M. Rojo | 28 | https://cdn.sofifa.org… | | Argentina | https: |
| 15210 | 10 | 241632 | B. Garré | 17 | https://cdn.sofifa.org… | | Argentina | https: |

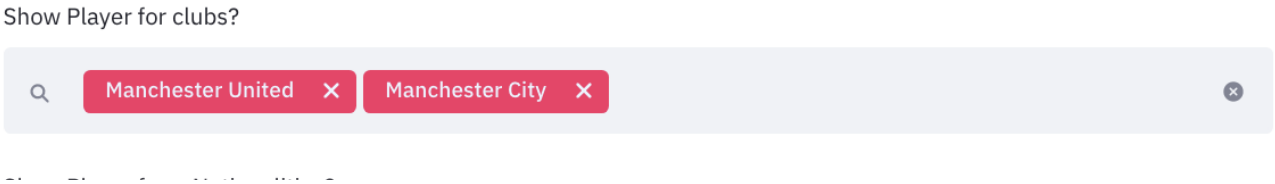
Using multiple widgets in conjunction

That was easy. But it seems pretty basic right now. Can we add some charts?

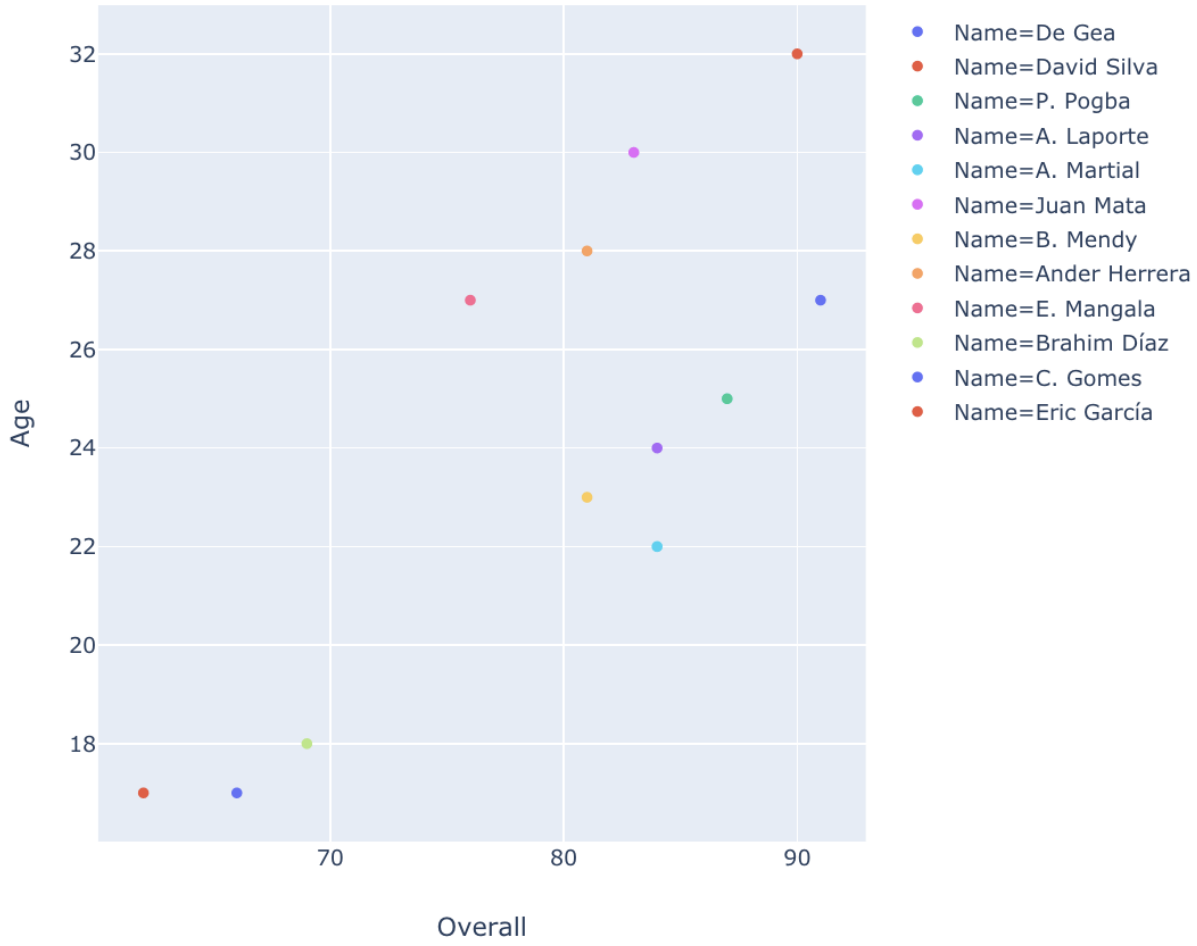
Streamlit currently supports many libraries for plotting. ***Plotly, Bokeh, Matplotlib, Altair, and Vega charts*** being some of them. ***Plotly Express*** also works, although they didn't specify it in the docs. It also has some inbuilt chart types that are "native" to Streamlit, like `st.line_chart` and `st.area_chart`.

We will work with `plotly_express` here. Here is the code for our simple app. We just used four calls to streamlit. Rest is all simple python.

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

df = pd.read_csv("football_data.csv")

clubs = st.multiselect('Show Player for clubs?',
df['Club'].unique())
nationalities = st.multiselect('Show Player from Nationalities?',
df['Nationality'].unique())

new_df = df[(df['Club'].isin(clubs)) &
(df['Nationality'].isin(nationalities))]
st.write(new_df)

# create figure using plotly express
fig = px.scatter(new_df, x ='Overall',y='Age',color='Name')

# Plot!
st.plotly_chart(fig)
```



| | Unnamed: 0 | ID | Name | Age | Photo | National |
|---|---|---|---|---|---|---|
| 3 | 3 | 193080 | De Gea | 27 | https://cdn.sofifa.org… | S |
| 13 | 13 | 168542 | David Silva | 32 | https://cdn.sofifa.org… | S |
| 45 | 45 | 195864 | P. Pogba | 25 | https://cdn.sofifa.org… | Fr |
| 113 | 113 | 212218 | A. Laporte | 24 | https://cdn.sofifa.org… | Fr |
| 116 | 116 | 211300 | A. Martial | 22 | https://cdn.sofifa.org… | Fr |
| 211 | 211 | 178088 | Juan Mata | 30 | https://cdn.sofifa.org… | S |
| 352 | 352 | 204884 | B. Mendy | 23 | https://cdn.sofifa.org… | Fr |
| 374 | 374 | 191740 | Ander Herrera | 28 | https://cdn.sofifa.org… | S |
| 1304 | 1304 | 190531 | E. Mangala | 27 | https://cdn.sofifa.org… | Fr |
| 6559 | 6559 | 231410 | Brahim Díaz | 18 | https://cdn.sofifa.org… | S |
| 8966 | 8966 | 245035 | C. Gomes | 17 | https://cdn.sofifa.org… | Fr |




Adding charts

In the start we said that each time we change any widget, the whole app runs from start to end. This is not feasible when we create apps that will serve deep learning models or complicated machine learning models. Streamlit covers us in this aspect by introducing ***Caching***.

## 1. Caching

In our simple app. We read the pandas dataframe again and again whenever a value changes. While it works for the small data we have, it will not work for big data or when we have to do a lot of processing on the data. Let us use caching using the `st.cache` decorator function in streamlit like below.

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

df = st.cache(pd.read_csv)("football_data.csv")
```

Or for more complex and time taking functions that need to run only once(think loading big Deep Learning models), using:

```
@st.cache
def complex_func(a,b):
    DO SOMETHING COMPLEX

# Won't run again and again.
complex_func(a,b)
```

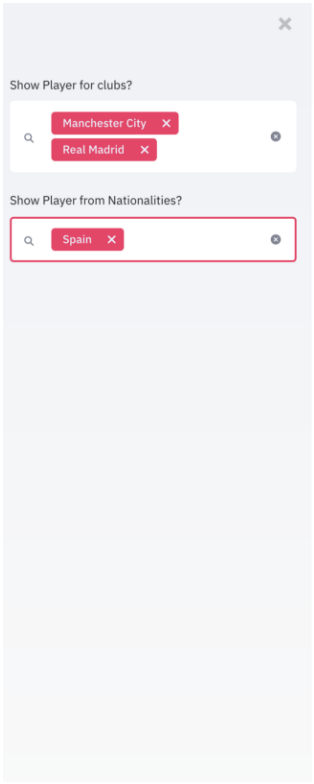
When we mark a function with Streamlit's cache decorator, whenever the function is called streamlit checks the input parameters that you called the function with.

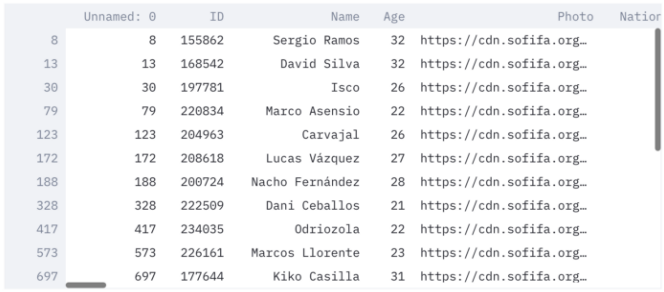
***If this is the first time Streamlit has seen these params, it runs the function and stores the result in a local cache.***

When the function is called the next time, if those params have not changed, Streamlit knows it can skip executing the function altogether. It just uses the results from the cache

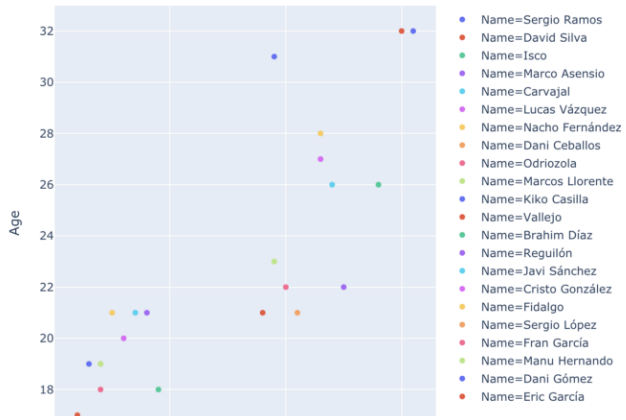
For a cleaner look based on your preference, you might want to move your widgets into a sidebar, something like Rshiny dashboards. ***This is pretty simple. Just add `st.sidebar` in your widget's code.***

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

df = st.cache(pd.read_csv)("football_data.csv")

clubs = st.sidebar.multiselect('Show Player for clubs?',
df['Club'].unique())
nationalities = st.sidebar.multiselect('Show Player from
Nationalities?', df['Nationality'].unique())

new_df = df[(df['Club'].isin(clubs)) &
(df['Nationality'].isin(nationalities))]
st.write(new_df)

# Create distplot with custom bin_size
fig = px.scatter(new_df, x ='Overall',y='Age',color='Name')

# Plot!
st.plotly_chart(fig)
```



| | Unnamed: 0 | ID | Name | Age | Photo | Natior |
|---|---|---|---|---|---|---|
| 8 | 8 | 155862 | Sergio Ramos | 32 | https://cdn.sofifa.org… | |
| 13 | 13 | 168542 | David Silva | 32 | https://cdn.sofifa.org… | |
| 30 | 30 | 197781 | Isco | 26 | https://cdn.sofifa.org… | |
| 79 | 79 | 220834 | Marco Asensio | 22 | https://cdn.sofifa.org… | |
| 123 | 123 | 204963 | Carvajal | 26 | https://cdn.sofifa.org… | |
| 172 | 172 | 208618 | Lucas Vázquez | 27 | https://cdn.sofifa.org… | |
| 188 | 188 | 200724 | Nacho Fernández | 28 | https://cdn.sofifa.org… | |
| 328 | 328 | 222509 | Dani Ceballos | 21 | https://cdn.sofifa.org… | |
| 417 | 417 | 234035 | Odriozola | 22 | https://cdn.sofifa.org… | |
| 573 | 573 | 226161 | Marcos Llorente | 23 | https://cdn.sofifa.org… | |
| 697 | 697 | 177644 | Kiko Casilla | 31 | https://cdn.sofifa.org… | |



Move widgets to the sidebar

## 3. Markdown?

I love writing in Markdown. I find it less verbose than HTML and much more suited for data science work. So, can we use Markdown with the streamlit app?

Yes, we can. There are a couple of ways to do this. In my view, the best one is to use Magic commands. Magic commands allow you to write markdown as easily as comments. You could also have used the command `st.markdown`

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

'''
# Club and Nationality App

This very simple webapp allows you to select and visualize players
from certain clubs and certain nationalities.
'''
df = st.cache(pd.read_csv)("football_data.csv")

clubs = st.sidebar.multiselect('Show Player for clubs?',
df['Club'].unique())
nationalities = st.sidebar.multiselect('Show Player from
Nationalities?', df['Nationality'].unique())

new_df = df[(df['Club'].isin(clubs)) &
(df['Nationality'].isin(nationalities))]
st.write(new_df)

# Create distplot with custom bin_size
fig = px.scatter(new_df, x ='Overall',y='Age',color='Name')

'''
### Here is a simple chart between player age and overall
'''

st.plotly_chart(fig)
```

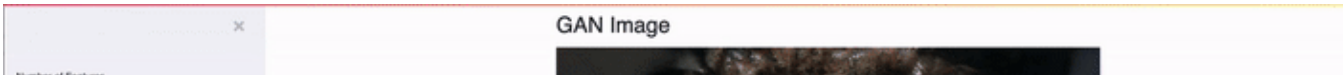

Our final App Demo

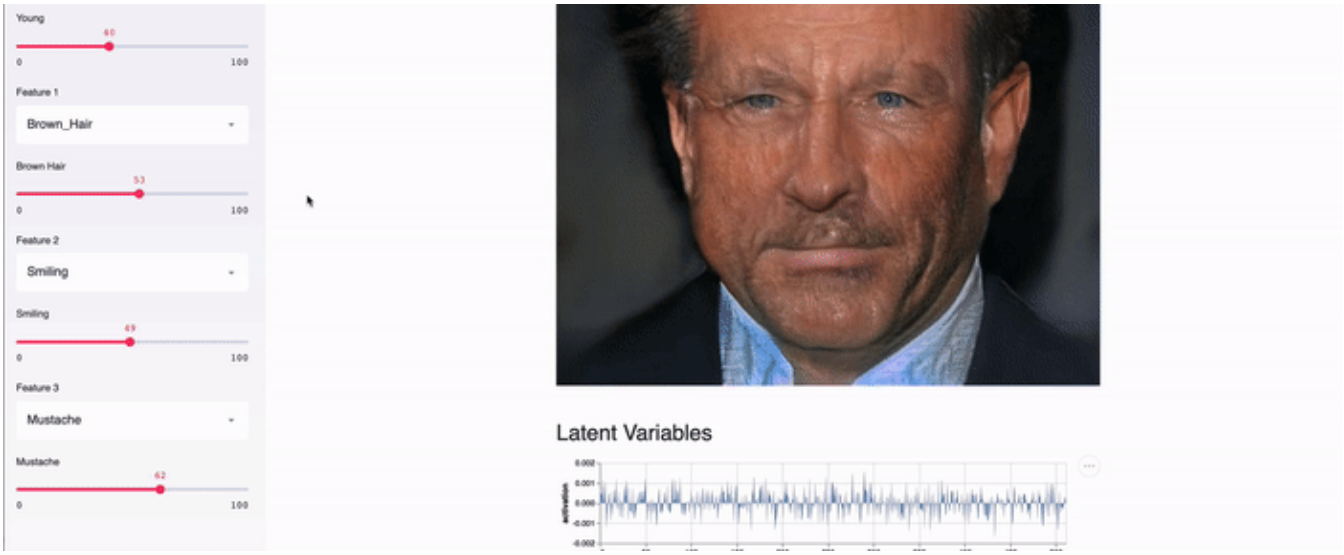
. . .

## Conclusion

Streamlit has democratized the whole process to create apps, and I couldn't recommend it more.

In this post, we created a simple web app. But the possibilities are endless. To give an example here is face GAN from the streamlit site. And it works by just using the same guiding ideas of widgets and caching.






I love the default colors and styles that the developers have used, and I found it much more comfortable than using Dash, which I was using until now for my demos. You can also include audio and video in your streamlit apps.

**On top of that, Streamlit is a free and open-source rather than a proprietary web app that just works out of the box.**

In the past, I had to reach out to my developer friends for any single change in a demo or presentation; now it is relatively trivial to do that.

> I aim to use it more in my workflow from now on, and considering the capabilities it provides without all the hard work, I think you should too.

I don't have an idea if it will perform well in a production environment yet, but its a boon for the small proof of concept projects and demos. I aim to use it more in my workflow from now on, and considering the capabilities it provides without all the hard work, I think you should too.

You can find the full code for the final app here. If you want to deploy this app on Amazon ec2 check out my next post.

. . .

call out an excellent course about **Data Visualization and applied plotting** from the University of Michigan, which is a part of a pretty good **Data Science Specialization with Python** in itself. Do check it out.

. . .

Thanks for the read. I am going to be writing more beginner-friendly posts in the future too. Follow me up at **Medium** or Subscribe to my **blog** to be informed about them. As always, I welcome feedback and constructive criticism and can be reached on Twitter @mlwhiz.

Also, a small disclaimer — There might be some affiliate links in this post to relevant resources, as sharing knowledge is never a bad idea.

Python Data Science Programming Artificial Intelligence Machine Learning