BBM467 - Blog Post Project

Sheep

21945815, Fatih AY

How to Deploy a Machine Learning Model with Streamlit on Windows

Most ml projects can't see the deployment phase. There are many reasons for this. Maybe the project was abandoned, maybe it wasn't successful, maybe the homework wasn't about deployment and many more.

I am writing this blog to solve one of these reasons: developing a web app.
There are several ways for developing web apps. And one of them stands out for being easy and free; Streamlit.

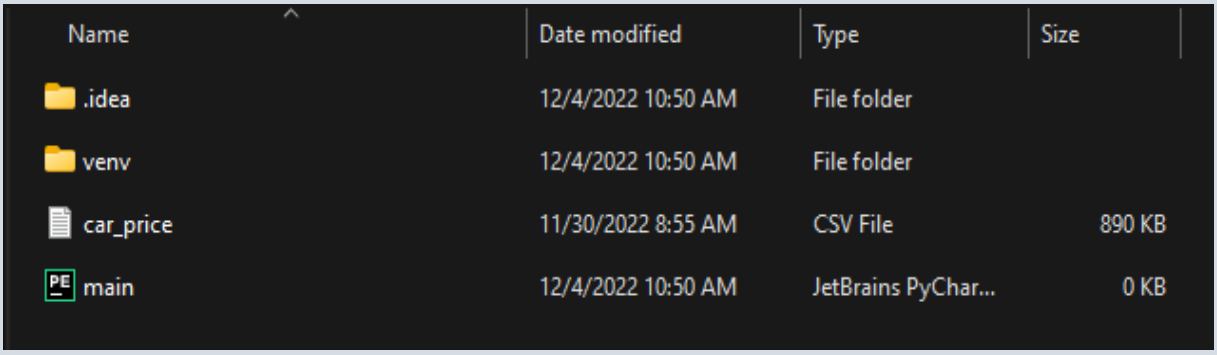
Streamlit is a free and open-source framework for rapidly building, sharing machine learning and building web apps. It is a Python-based library. If you don't use Windows, but want to use this library, don't worry. Streamlit also works on macOS and Linux.

You can find the dataset I used while writing this blog here.

[Preparing to Use The Library]

Open your project folder and upload your dataset to this folder. Open a new python file in this folder. Install the library using your IDE terminal with the command:

```
Pip install streamlit
```



Your project folder should look like this

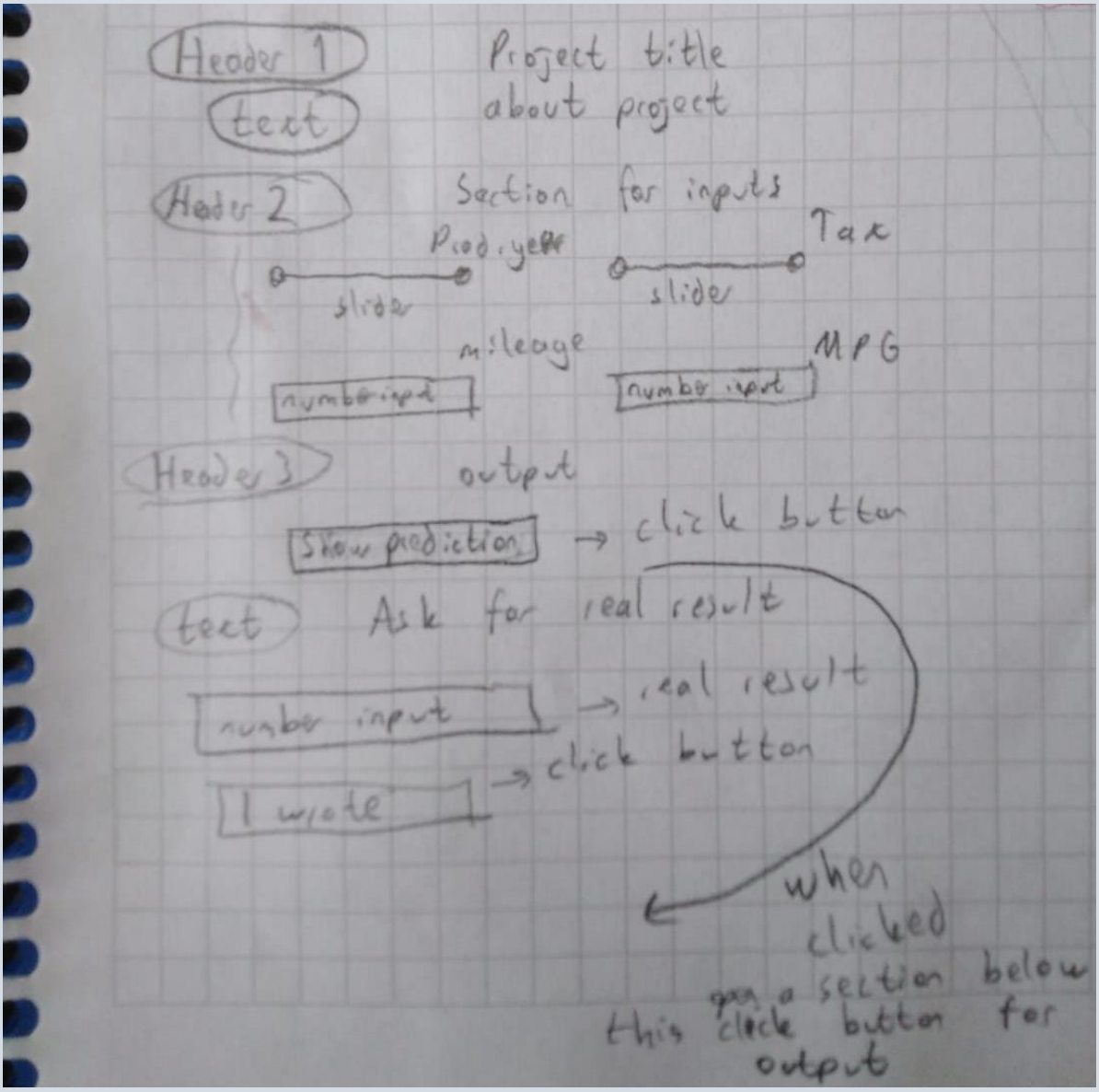
I would have liked to say that we are ready to write code now but first we have to decide on our design.

[Design]

Of course, to do this, you need to know what you can do with streamlit.

Basically, by default, everything you want to show goes below the previous thing if you are using Streamlit. In the Streamlit version that I used (1.15.2), you can divide your content into containers and columns. Containers are horizontal sections and columns are vertical sections. Columns go inside containers.

Taking this information to account, here is my design:



Don't force yourself to write beautifully like me 😊 Don't forget, you are preparing the design for yourself, you do not need to show it to anyone else.

Now you can breathe a sigh of relief because we're finally starting the coding part.

First of all, make all the horizontal sections your design needs. For my design I used:

```
import streamlit as st

Header1 = st.container()
Header2 = st.container()
Header3 = st.container()
```

And for adding title and texts:

```
with Header1:
    st.title("ML model for the price of cars")
    st.text("This project was made to set an example for people.")
```

Easy, right? Maybe you are wondering what's up with the "with" divider. "with" divider helps to write code that belongs to a container.

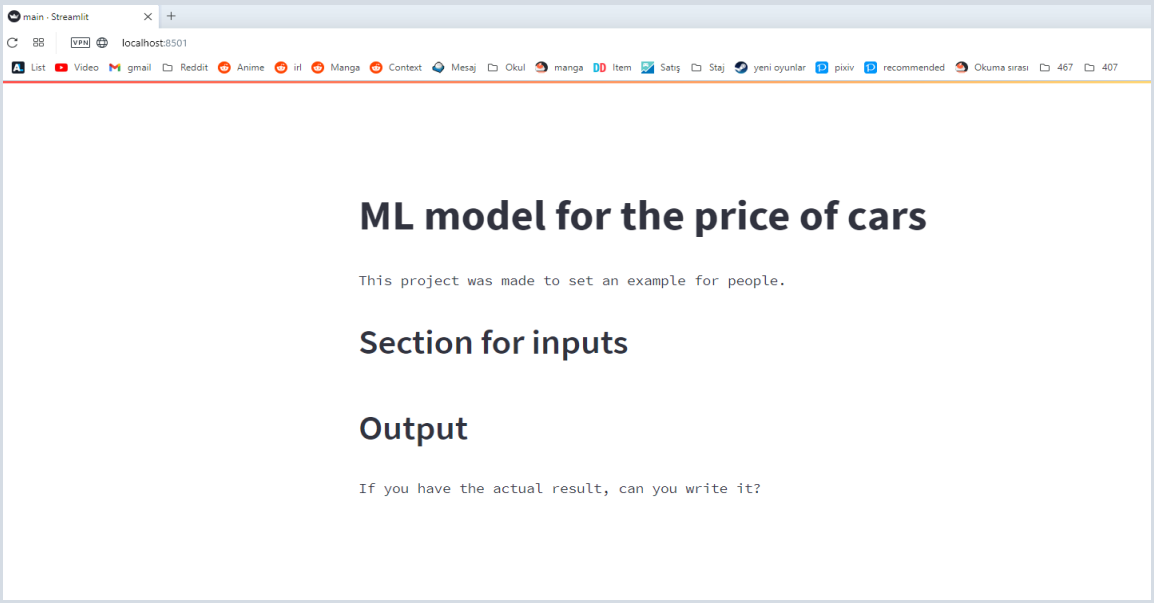
When I add the texts and titles that I needed to add according to my design, my code looks like this:

```
import streamlit as st

Header1 = st.container()
Header2 = st.container()
Header3 = st.container()

with Header1:
    st.title("ML model for the price of cars")
    st.text("This project was made to set an example for people.")
with Header2:
    st.header("Section for inputs")
with Header3:
    st.header("Output")
    st.text("If you have the actual result, can you write it?")
```

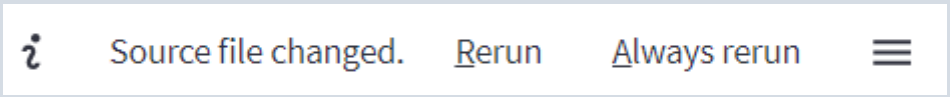
You might wonder what the app looks like after I've gotten this far. To see that you can open the IDE's terminal and run this command:

```
streamlit run pythonFile_path
```

A tab like this needs to open in your default browser as a result of running this:



If you are thinking "Do I have to enter the terminal every time I want to see this site?", the answer is no.
Let's assume that you have not closed the tab yet, you will see the rerun options on the top right side of your screen.



You might want to add your model before doing the rest. But there is one problem. Every time you open your app, click on buttons, change inputs, etc. your code will probably run all over again. So, it is constantly retraining your model. Because of this, your app is probably too slow. Well then, you might be wondering why does anyone use Streamlit? Because there is a solution for this, and it is called caching.

[Caching]

When you mark a particular piece of function for caching, the application saves the result of that piece of code for the first run, and after that uses the record instead of calling it again each time as long as the input for this function, some global parameters in this function etc. given to this code does not change.

It is easy to use:

```
@st.cache
def trained_model():
    data = pd.read_csv("car_price.csv")
    #  If you are wondering about the rest of the code, you can check the repository I gave the link at the end of the blog.
    return new_model
```

[Collecting the inputs]

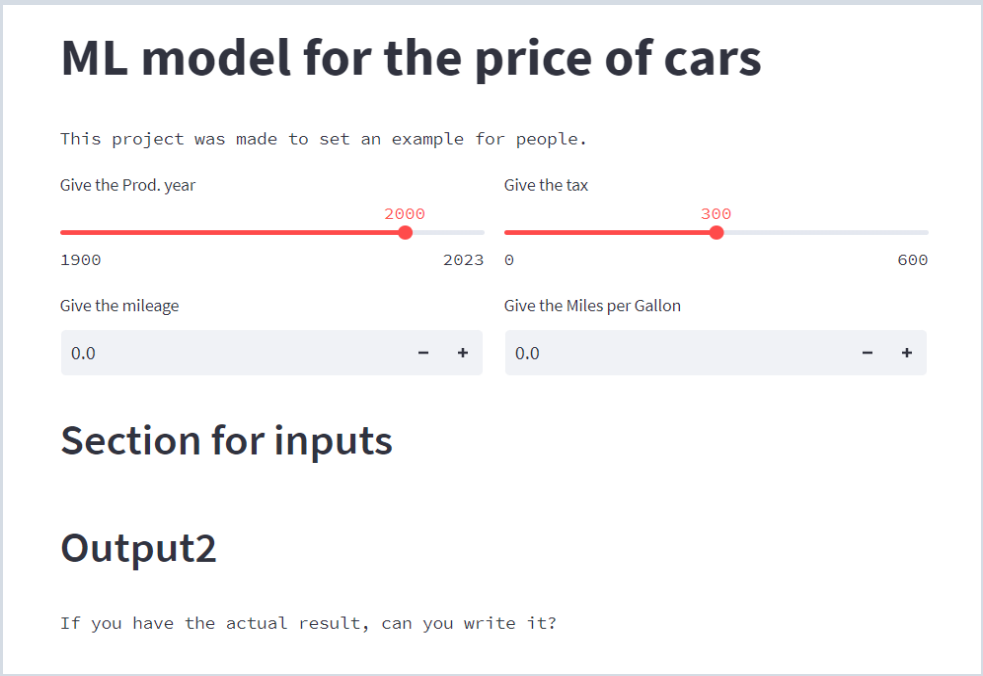
Okay, we've come this far. But how do we get the input?
It may seem complicated at first, thus before anything, I'll show you how to add a slider:

```
with Header1:
        column1, column2 = st.columns(2)
        year = column1.slider("Give the Prod. year", min_value=1900, max_value=2023,
step=1, value=2000)
```

The variable named "year" will be equal to the value in this slider. Here you can see my final input section:

```
with Header2:
   st.header("Section for inputs")
   inputProdYear, inputTax = st.columns(2)
   inputMileage, inputMPG = st.columns(2)  # These will be under the columns we made
in number 2
   year = inputProdYear.slider("Give the Prod. year", min_value=1900, max_value=2023,
step=1, value=2000)
   tax = inputTax.slider("Give the tax", min_value=0, max_value=600, step=5, value=300)

   mileage = inputMileage.number_input("Give the mileage", step=500.0, format="%.1f")
   mpg = inputMPG.number_input("Give the Miles per Gallon", step=1.0, format="%.1f")
```

I rerun my app again to check if i did it right before moving on to the next phase:



Good, there is no problem

You have your inputs, and your model is ready. So, what do we have to do before finally deploying our code?

[Using the inputs]

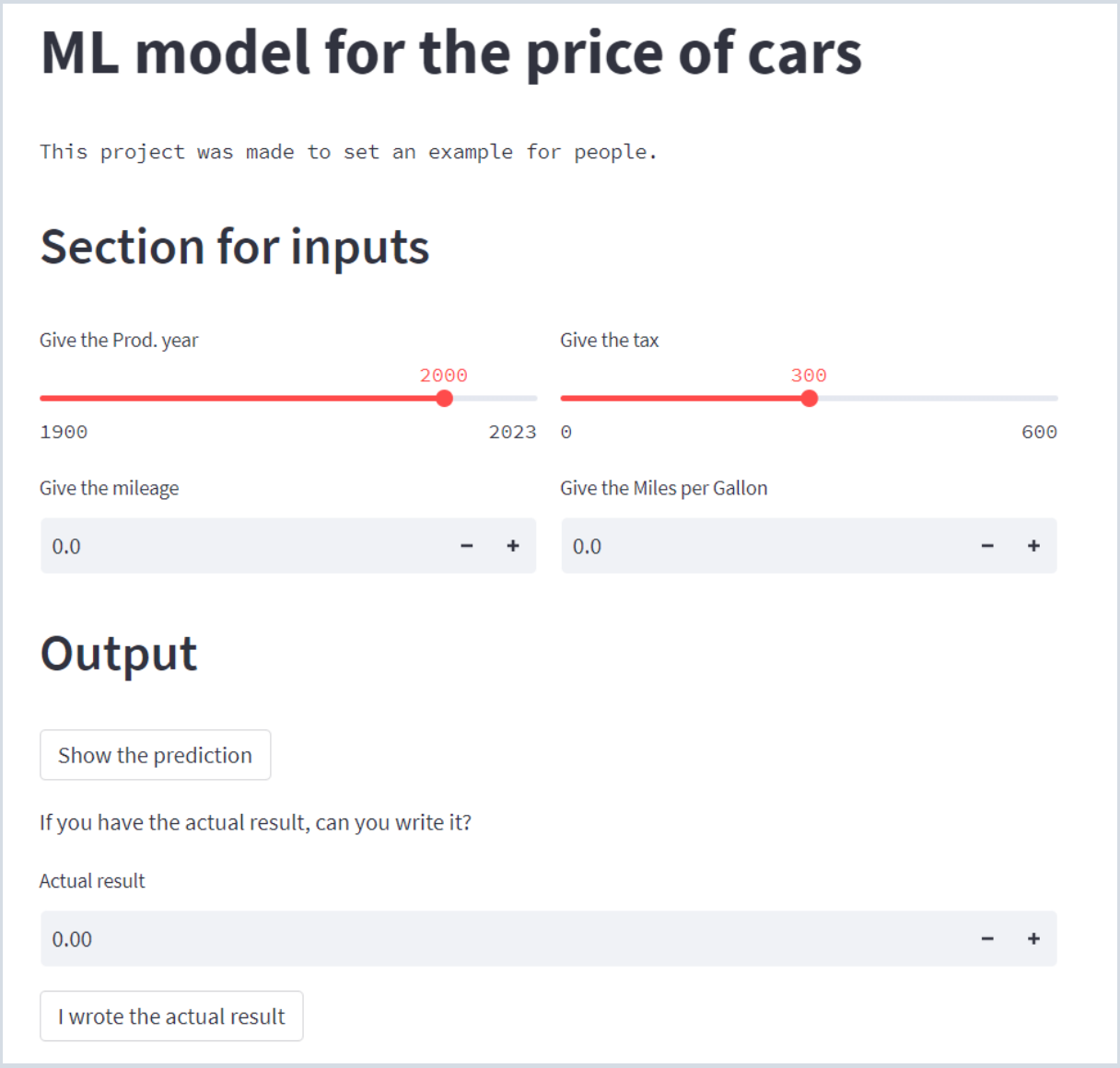
The only new tool we will use here is the click button. We should not forget that the click button will return true every time it is clicked. Or, from a different perspective, if our app reruns for some reason other than clicking the click button, the click button returns false.

```
with Header3:
        predict_button = st.button("Show the prediction")
        output = st.container()
        if predict_button: #  add section that opens on button click
                #  Do something
```

If you understand this section fully, you should not have any difficulties in this part. You can see my final output section below:

```
with Header3:

   model = trained_model()
   st.header("Output")
   predict_button = st.button("Show the prediction")
   output = st.container()

   if predict_button:
      output.write(model.predict([[year, mileage, tax, mpg]]))  # Displaying the prediction

   # To get feedback
   st.write("If you have the actual result, can you write it?")
   number = st.number_input("Actual result", step=10.0, format="%.2f")
   getRealButton = st.container()

   if getRealButton.button("I wrote the actual result"):
      st.write("Do something")
   # Here you can save the actual result with the model's prediction for later comparison
```

You may be thinking that you can finally start deployment. I advise you to stop that thought. Unsurprisingly, we have to check if it is correct first.



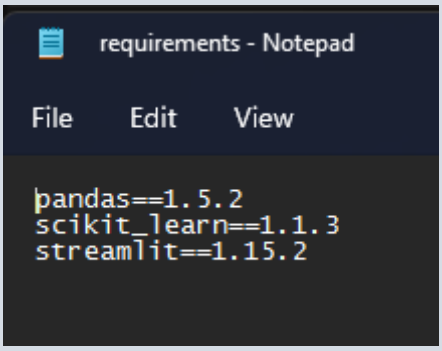
Beautiful, we are finally done with the coding part. Now we can start the deployment part.

[Deployment]

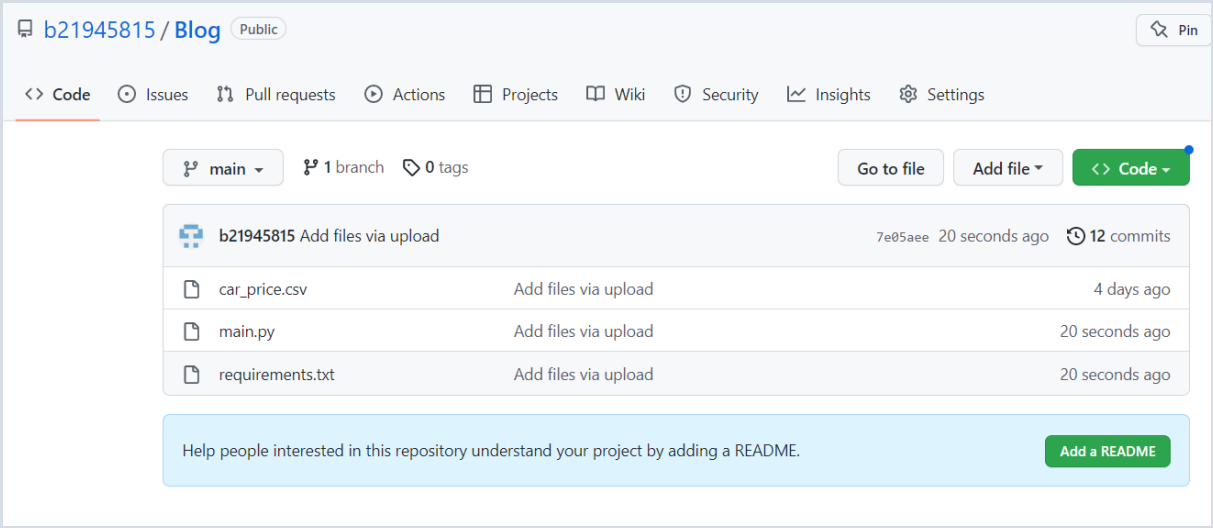
Our first job is readying the requirements file. For this, run the following codes in the Windows Terminal, respectively:

```
cd applicationFolderPath
pip install pipreqs
pipreqs ./
```

Your requirements file (that is in the project folder) should look like this:



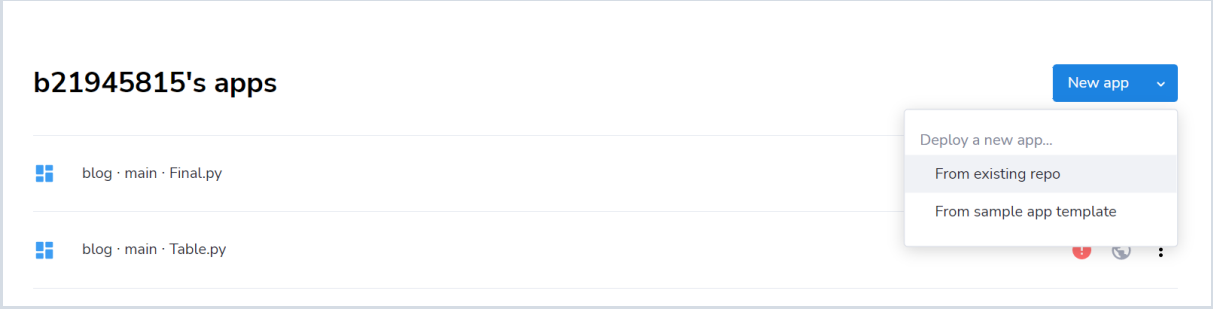
Afterwards, open a public repository on Github. Don't worry, it doesn't have to be public. Anyway, assign a meaningful name to it and upload your application folder to the repository.



It should look like this

Finally, open the Streamlit website and log in, and link your Github account to your streamlit account from account settings.

After that, go here



And choose New App -> From existing repo

Than you can enter the necessary information and press the deploy button. All that's left to do is to wait for the application to bake in the oven 😊

I followed these steps and my app is running smoothly.

You can check my web app here
If you want to check my repository it is here
And for extra information, the documentation for streamlit at here

Thank you for reading this post. If you have anything to say/object/correct, please drop a comment down below.