- ⭐ **Marmitex**

Rationale Open in Streamlit

Streamlit doesn't support running under HTTPS, but you might want to do that. One way to achieve this is to use a reverse proxy, and enable HTTPS on the proxy server. The user's browser connects to the proxy over HTTPS. The proxy server terminates the HTTPS connection and forwards requests over HTTP to Streamlit. Responses from Streamlit go via the proxy, back to the user.

``` ┌──────────────┐ ┌──────────┐ ┌──────────────┐ │ │──────────►│
│──────────►│ │ │ Browser │ HTTPS │ Proxy │ HTTP │ Streamlit │ │ │◄- - - -
-│ │◄- - - - - -│ │ └──────────────┘ └──────────┘ └──────────────┘

──────────► Request

◄- - - - - - Response ```

## Tech Stacks Used

PYTHON
```

Libraries Used

PANDAS PLOTLY STREAMLIT

Prerequisites

- Python 3.6.X
- Pandas
- numpy
- plotly
- streamlit
- docker
- openpyxl

Model Deployment

- The web application is built using a python library -> Streamlit
- The entire application is finally deployed on Streamlit CLoud Share Platform
- See the deployed application here

Basics of Streamlit

What is Streamlit?

Streamlit is an open source web framework that lets you build webpages using only Python :snake:. Thereby, removing the need for developers to use HTML, CSS and Js. It uses what is called as widgets/components to construct your website.

Developer Trivia: Streamlit components are essentially Python wrappers of React code (Judges, extra points :grin:?).

Installing streamlit

~~~ pip install streamlit ~~~

Write some streamlit code

Running streamlit files

Let us assume our file name to be 'streamlit_basics.py' and that we are in the correct directory. ~~~ streamlit run streamlit_basics.py ~~~
~~~

Installation

Clone this repository :

```
git clone https://github.com/
cd bond-valuation-python
```

You may want to use a virtual environment to keep a clean Python installation :

```
python3 -m venv myenv/
source myenv/bin/activate
```

Frontend Using Streamlit

Step -1 Install the requirements

```
pip install -r requirements.txt
```

Step -2 Run the app.py

```
streamlit run app.py --server.address 0.0.0.0 --server.port 8080
```

Dockerized Streamlit App

> Prototyping a user interface with Streamlit

Sometimes we need to resolve very quickly a delivery or we have to validate a protopyte.

The objective of this project is to provide an interface that allows to a person (who don`t handle Conda or Jupyter Notebook) select an xls, process its content and obtain the result needed.

Here we go!

Run the app

The first time the project is used, the following commands should be run:

1. Build container: `docker build -f docker/Dockerfile -t streamlitapp:latest .`
2. Run container: `docker run -d -p 8501:8501 streamlitapp:latest`

3. You can now view your Streamlit app in your browser http://localhost:8501

4. Browse to upload the xls file and select the example file located ./example/

5. Download the new file generated width the transform

6. To shut down container: `docker ps` `docker stop CONTAINER_ID`

Deploying to Heroku

First, install heroku and login. To create a new deployment, run inside `traingenerator`:

```
heroku create git push heroku main heroku open
```

To update the deployed app, commit your changes and run:

```
git push heroku main
```

Optional: If you set up a Github repo to enable the "Open in Colab" button (see above), you also need to run:

```
heroku config:set GITHUB_TOKEN=<your-github-access-token> heroku
config:set REPO_NAME=<user/notebooks-repo>
```

Built With

- Numpy - Fundamental package for scientific computing with Python,
- Pandas - Library providing data structures and data analysis tools for Python,
- Streamlit - Open-source app framework,

Dependencies

All the Python packages needed are listed in requirements.txt and on the Dependency Graph page.

Examples

amazing examples by Streamlit https://streamlit.io/gallery

For more, please refer to the Documentation

Contributing

Contributions are what make the open source community such an amazing place to learn, inspire, and create. Any contributions you make are **greatly appreciated**.

1. Fork the Project
2. Create your Feature Branch (`git checkout -b feature/AmazingFeature`)
3. Commit your Changes (`git commit -m 'Add some AmazingFeature'`)
4. Push to the Branch (`git push origin feature/AmazingFeature`)
5. Open a Pull Request

License

This project is licensed under the MIT License.

Authors