



TOPIC: Streamlit Application Debugging and Deployment

SUMMARY:

The summary of this project is to create a streamlit web application on airline dataset collected on the month of February 2015 that displays all the graphs and other visualizations created and also to predict the sentiment of the tweets are provided on the web page using a machine learning model. Here the model used is Logistic Regression to predict the sentiment of the tweet. Finally, also deploying the web application using Heroku cloud server.

TOOL AND REQUIRED:

Let's understand some of the terms before going through the project.

- Heroku CLI
- Git CLI
- Python

HEROKU CLI AND HEROKU:

The Heroku Command Line Interface (CLI) makes it easy to create and manage your Heroku apps directly from the terminal. It's an essential part of using Heroku.

Heroku is a container-based cloud Platform as a Service (PaaS). Developers use Heroku to deploy, manage, and scale modern apps. Our platform is elegant, flexible, and easy to use, offering developers the simplest path to getting their apps to market.

GIT CLI:

Git is a set of command line utility programs that are designed to execute on a Unix style command-line environment. Modern operating systems like Linux and macOS both include built-in Unix command line terminals.

PYTHON:

Python is a computer programming language often used to build websites and software, automate tasks, and conduct data analysis. Python is a general purpose language, meaning it can be used to create a variety of different programs and isn't specialized for any specific problems.

The python libraries used for this project are as follows

- Streamlit
- Nltk
- WordCloud
- Matplotlib and Seaborn
- Plotly
- Sklearn
- Regular Expression

- Pickle

STREAMLIT:

Streamlit is an open-source Python library that makes it easy to create and share beautiful, custom web apps for machine learning and data science. In just a few minutes you can build and deploy powerful data apps.

NLTK:

The Natural Language Toolkit (NLTK) is a platform used for building Python programs that work with human language data for applying in statistical natural language processing (NLP). It contains text processing libraries for tokenization, parsing, classification, stemming, tagging and semantic reasoning.

WORDCLOUD:

Word Clouds (also known as wordle, word collage or tag cloud) are visual representations of words that give greater prominence to words that appear more frequently.

MATPLOTLIB AND SEABORN:

Matplotlib, is a Python library used for plotting graphs with the help of other libraries like Numpy and Pandas. It is a powerful tool for visualizing data in Python. It is used for creating statical interferences and plotting 2D graphs of arrays.

Seaborn, also a Python library used for plotting graphs with the help of Matplotlib, Pandas, and Numpy. It is built on the roof of

Matplotlib and is considered as a superset of the Matplotlib library. It helps in visualizing univariate and bivariate data. It uses beautiful themes for decorating Matplotlib graphics

PLOTLY:

Plotly graphing library makes interactive, publication-quality graphs. It more sophisticated data visualization tool that is better suited for creating elaborate plots more efficiently.

SKLEARN:

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python.

REGULAR EXPRESSION:

Regular expressions are useful in search and replace operations. The typical use case is to look for a sub-string that matches a pattern and replace it with something else.

PICKLE:

Pickle in Python is primarily used in serializing and deserializing a Python object structure. In other words, it's the process of converting a Python object into a byte stream to store it in a file/database, maintain program state across sessions, or transport data over the network.

DEBUGGING:

The python files present in the application are

- project.py (Runs the entire project.)
- ml_app.py (Contains the ML part of the project.)
- data_app.py (Contains the streamlit code of the project.)
- create_wordcloud.py (Contains the word cloud generation code.)

The initial code provided contained some bugs which was removed by step by step analyzing. Most of the errors present in the data_app.py which I debugged. The final web app after removing all the bugs can be executed by running the following command on the command prompt.

streamlit run project.py

It runs on the "<http://localhost:8501/>" where it can be accessed only by the local machine.

Streamlit also provides a network URL where multiple system connected to the network can access the link. Here the system that runs the application acts as a server. The final running application can be accessed after the deployment.

DEPLOYMENT:

Before deploying the application into the cloud server, there are certain steps that needs to carried out. The steps are as follows

STEP 1: Place all the code that is responsible for the proper running of the application.

STEP 2: Along with the project files the project directory must contain three additional files. They are

- requirements.txt (Contains all the dependencies to run the application.)
- Procfile (Contains the command that runs the application.)

It contains the following line of code.

web: sh setup.sh && streamlit run project.py

- setup.sh (This is a shell file, that needed to add the following shell command inside the file.)

It contains the following lines of code.

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

```

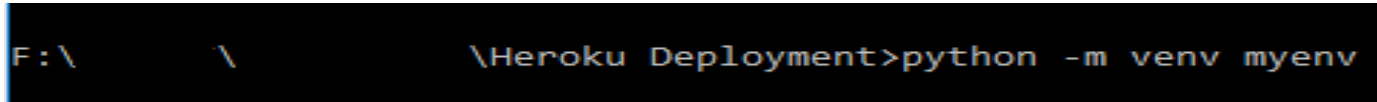
```
" > ~/.streamlit/config.toml
```

Note: setup.sh is used here because of the usage of streamlit.
For libraries other than streamlit it is not required.

STEP 3: Since the project requires only fewer number of libraries, we can use a python virtual environment to implement the project. After reaching the project directory we can create a virtual environment by using the following command.

```
python -m venv virtual_environment_name
```

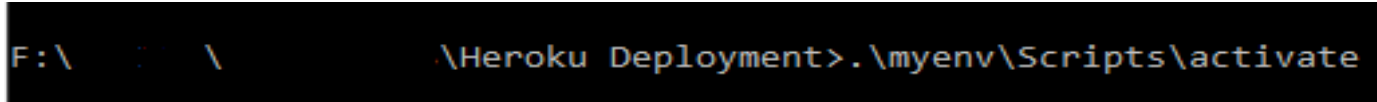
I have named my virtual environment myenv which can be seen in the image below.

A terminal window with a black background and yellow text. The prompt is 'F:\ > ' followed by the command 'python -m venv myenv'.

```
F:\ > python -m venv myenv
```

STEP 4: After creation of the virtual environment, we must enter the virtual environment to implement the project. The following command is used to enter the virtual environment. The following command can be used to enter the virtual environment.

```
.\environment_name\Scripts\activate
```

A terminal window with a black background and yellow text. The prompt is 'F:\ > ' followed by the command '.\myenv\Scripts\activate'.

```
F:\ > .\myenv\Scripts\activate
```

After successful running of the above command, we can see the environment name on the left of the command line, i.e we have entered the environment.

```
F:\          \          \Heroku Deployment>.\myenv\Scripts\activate
(myenv) F:\          \          \Heroku Deployment>
```

STEP 5: After entering the virtual environment we have install all the required packages for the implementation of the project. In python we can install a package using the following command.

pip install package_name

```
(myenv) F:\          \          \Heroku Deployment>pip install streamlit matplotlib seaborn plotly nltk wordcloud sklearn numpy pandas
```

STEP 6: Next, we have to create requirements.txt, that contains all the packages that are required to run the application. The following command is used to the following command to create the requirements.txt

pip freeze > requirements.txt

```
(myenv) F:\          \          \Heroku Deployment>pip freeze > requirements.txt
```

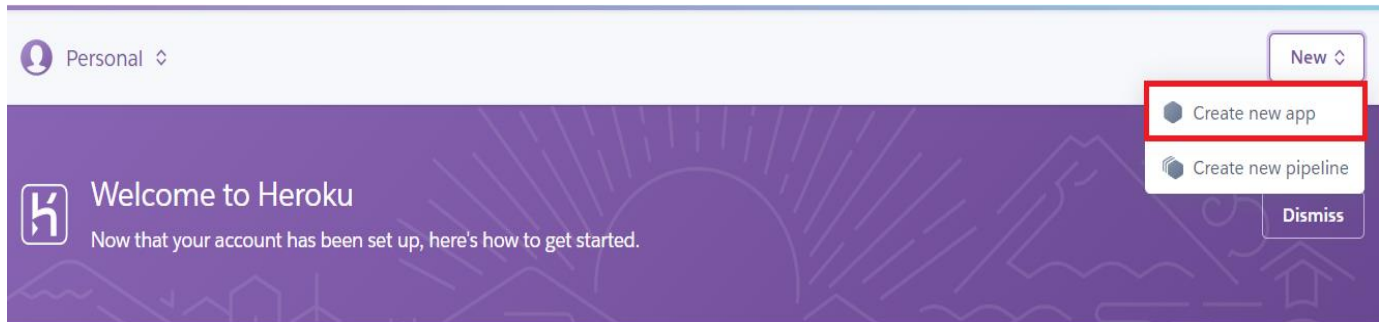
Also, we should create the Procfile and setup.sh as mentioned earlier.

STEP 7: After creating all the required files, we have to login to Heroku. We can login to Heroku using the command

heroku login

It will ask us to press any button which gets redirected to browser where we can see the login page to log into Heroku.

STEP 8: After logging into Heroku, we have to create a new app by selecting New → Create new app. Enter the name for the app and click create app.



STEP 9: Now we have create a empty repository. For that we have to run the following command.

git init

This command automatically initializes git by creating a folder(named .git) which is used to track the changes that are made in the project.

```
(myenv) F:\          \          \Heroku Deployment>git init .  
Initialized empty Git repository in F:/          /          /Heroku Deployment/.git/
```

Note: The .git folder is a hidden folder so it might not be visible normally.

STEP 10: Since .git in a empty repository we have to save the changes manually. We can check if the changes are saved using the command

git status

```
(myenv) F:\... \Heroku Deployment>git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    Procfile
    __pycache__/
    create_wordcloud.py
    data/
    data_app.py
    img/
    ml_app.py
    myenv/
    pickle/
    project.py
    requirements.txt
    setup.sh

nothing added to commit but untracked files present (use "git add" to track)
```

Here we can see that all the files are in red color indicating that they are not saved in the repository. To stage those files, we can run the command

git add *

After running this command, we can see that all the files present in the virtual environment are staged and are in green color.

STEP 11: After successful running of the above command, we need to commit those changes to git, which can be done by using the command

git commit -m "message"

message, contains the message to be passed with the command for reference. I have given the message as "first commit".

```
(myenv) F:\... \Heroku Deployment>git commit -m "first commit"
```

STEP 12: Since we already have an existing repository, we have use remote command to push the project. So, we can the command

heroku git:remote -a app_name

I have passed my app name as airline-sentiment-analyzer.

```
(myenv) F:\          \          \Heroku Deployment>heroku git:remote -a airline-sentiment-analyzer
```

STEP 13: Finally, to push all the changes into Heroku and run the application we have to run the command

git push heroku master

```
(myenv) F:\          \          \Heroku Deployment>git push heroku master
```

STEP 14: Next, we can see that Heroku starts installing some applications and also starts installing all the required packages from requirements.txt.

Note: For some Windows users there may occur some error while installing some packages like

- **pywin32** – It occur since it a Windows default package and Heroku internally uses Linux. To counter this problem simply remove pywin32 from requirements.txt. After saving that, we track the change made by git using the command **git status**.

```
(myenv) F:\          \          \Heroku Deployment>git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   requirements.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

To save those changes we can again run the command (**git add ***) and check its status (**git status**) and commit those changes using (**git commit -m "message"**).

```
(myenv) F:\      \      \Heroku Deployment>git add *
(myenv) F:\      \      \Heroku Deployment>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   requirements.txt

(myenv) F:\      \      \Heroku Deployment>git commit -m "Removed pywin32 from requirements.txt"
[master 0618f5ae] Removed pywin32 from requirements.txt
1 file changed, 1 deletion(-)
```

- **pywinpty** – This library has a version issue that must be resolved. In the requirements.txt replace (pywinpty==1.1.6) with (pywinpty<1.1.6) and follow the same procedures used for pywin32 for adding and committing the changes.

STEP 15: After making every change and committing the we can run the app using the command

git push heroku master

STEP 16: Finally, we can run the application using the URL provided in the terminal. We can see at the end it shows that deployment is done and it provides the URL to for the application.

Here is the URL for my application: <https://airline-sentiment-analyzer.herokuapp.com/>

```
(myenv) F:\      \      \Heroku Deployment>git push heroku master
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 340 bytes | 340.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Compressing source files... done.
remote: Building source:
remote:
remote: -----> Building on the Heroku-20 stack
remote: -----> Using buildpack: heroku/python
remote: -----> Python app detected
remote: -----> No Python version was specified. Using the same version as the last build: python-3.9.9
remote: -----> To use a different version, see: https://devcenter.heroku.com/articles/python-runtimes
remote: -----> No change in requirements detected, installing from cache
remote: -----> Using cached install of python-3.9.9
remote: -----> Installing pip 21.3.1, setuptools 57.5.0 and wheel 0.37.0
remote: -----> Installing SQLite3
remote: -----> Installing requirements with pip
remote: -----> Downloading NLTK corpora...
remote: !      'nltk.txt' not found, not downloading any corpora
remote: !      Learn more: https://devcenter.heroku.com/articles/python-nltk
remote: -----> Discovering process types
remote:      Procfile declares types -> web
remote:
remote: -----> Compressing...
remote:      Done: 468.6M
remote: -----> Launching...
remote: !      Warning: Your slug size (468 MB) exceeds our soft limit (300 MB) which may affect boot time.
remote:      Released v4
remote:      https://airline-sentiment-analyzer.herokuapp.com/ deployed to Heroku
remote:
remote: Verifying deploy... done.
To https://git.heroku.com/airline-sentiment-analyzer.git
    a3657068..ef43ec3c  master -> master

(myenv) F:\      \      \Heroku Deployment>
```