Indiana University Bloomington

Time Series Analysis

Final Project Part 3

Project title: UFO sightings analysis
Project team: Anas Omary (individual project)

Part 0. Introduction

Unidentified Flying Object (UFO) as a topic has gained traction in recent years, and has been an intriguing "spooky" topic for years that it became part of our culture with many Hollywood movies and series about UFOs, although it is part of our culture, the subject of UFOs has been stigmatized and laughed at in science communities and scientists have approached this subject with caution in fear of being called pseudo scientists or being described as tin foil hat conspiracy theorists.

Objectives

For this project I will follow a scientific approach in analyzing UFO related data to try to draw subjective conclusions on wither UFOs are real or not and where those sightings happen, For the pure fun of it!

Project description

Usefulness

In this project I will use shiny library to create an interactive dashboard that shows the history and progress of UFO sightings worldwide with multiple criteria filters, I haven't found many visualizations on the web that tackles all aspects of UFO sightings and haven't found any interactive dashboards, so I will make the dashboard interactive to give the ability for users to view data from different angles, stakeholders for this project are any person interested in the UFO subject.

Dataset

I will use UFO sightings dataset from data.world ((https://data.world/timothyrenner/ufo-sightings))

Origin: The National UFO Research Center (NUFORC) This dataset contains the UFO sightings collected worldwide from 1969 till 2022 for the purpose of investigating and analyzing those sightings, as far as data reliability the dataset seems reliable and a result of actual sightings reported by actual people across the world, the dataset includes attributes like observation date and time, location, duration, UFO shape and other attributes in the raw form as it is recorded on the NUFORC site which makes this dataset a good candidate for time series analysis.

Communication and sharing

This is an individual project where I will be working solo to create the UFO sightings interactive dashboard

Github

All project files can be found on github in the following repository:
https://github.com/ANASOMARY/TimeSeriesAnalysis_FinalProject

Part 1. Time Series Application Architecture

Summary

The UFO analysis application is hosted entirely on google cloud platform, I created a new project named “TimeSeriesAnalysis” on GCP and under the project I created a storage bucket and a virtual machine that runs Linux ubuntu and did the needed firewall configurations for VM to allow connections to the deployed application, then installed python and the needed libraries like Dash, Tensorflow …etc. LSTM model and dataset are stored on a cloud bucket along with a backup of the web application app.py, and the application is deployed on a virtual machine on GCP that calls the dataset file which is a CSV dataset and the LSTM model which is used to predict time series, from the storage bucket.

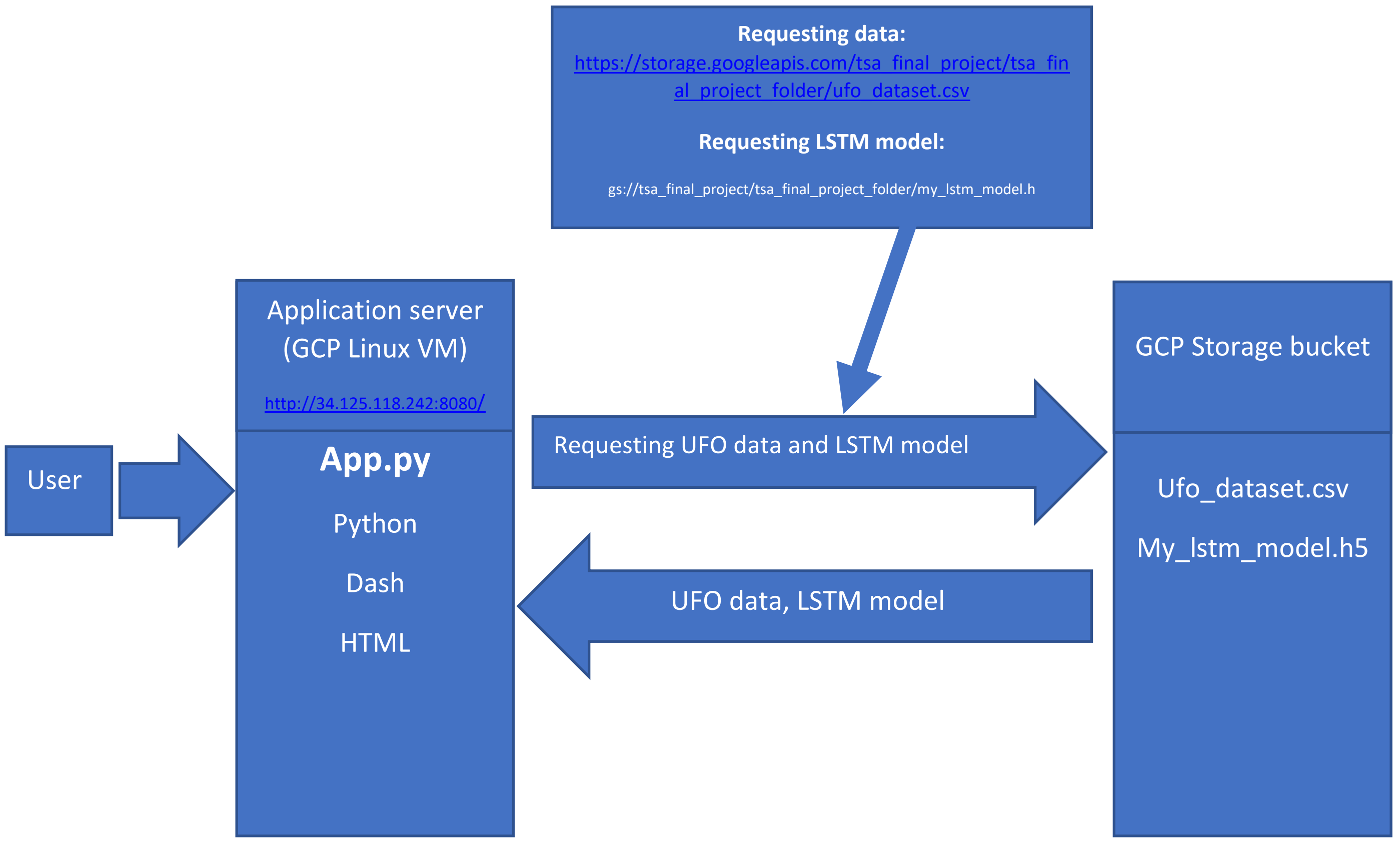
The app.py application running on the virtual machine is a web application developed using python and dash, it connects directly to the CSV dataset that is located on a GCP storage bucket and is configured to be publicly available, the app.py calls a saved LSTM model which is the outcome of part 2 of this project where 3 models were trained and tested then finally I decided to use the LSTM model because it scored the highest, the LSTM model is saved on the same storage bucket (my_lstm_model.h5) , the connection security was not a big concern since the data is already publicly available and does not contain any sensitive information.

The user can reach the application by calling the URL (http://34.125.118.242:8080/)

The front-end layout is created using dash and HTML, I used “dash_html_components” library to create HTML elements, and used dash_core_components to create the UI controls, the UI consists of the following elements:

1- H1, H2, H3 headers: those are created using dash_html_components
2- Date range picker: this is a date range picker where the user will choose the start and end date for the period that will be analysed.
3- Drop down lists: there will be 3 drop-down lists where the user will choose the country, state, and city in which UFO sightings happened, whatever the user chooses will be reflected in the visualizations
4- Button: the submit button is created using HTML and will be used to fire the event which will redraw the visualizations based on the user choices.
5- 2 graph elements: I used dash to create two graph elements, one will show a bar-chart and the other one will show a time series visualization with prediction.

Web app architecture

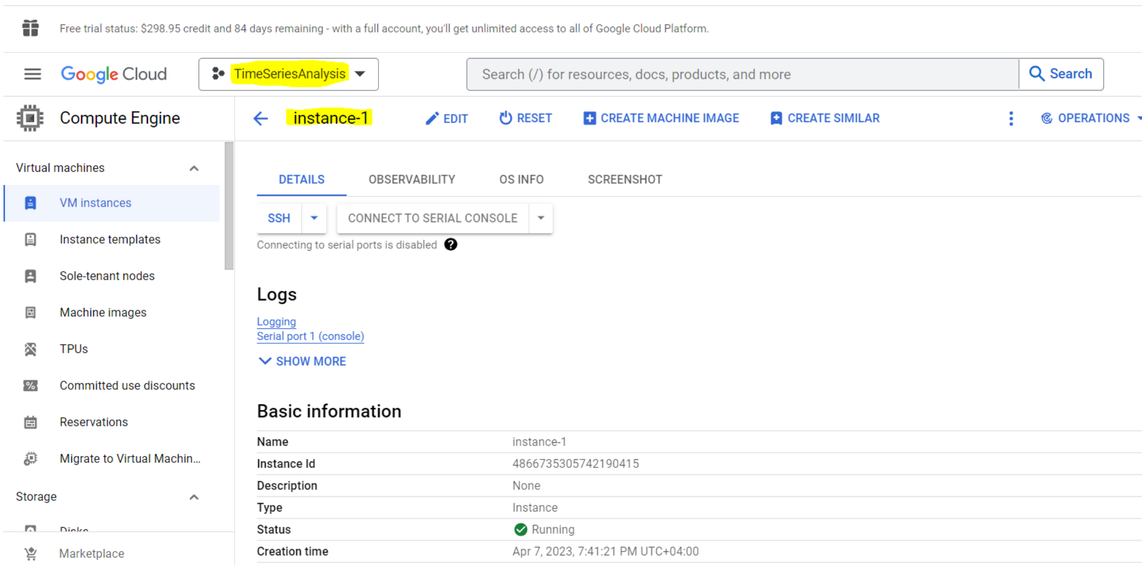

Requesting data:
https://storage.googleapis.com/tsa_final_project/tsa_final_project_folder/ufo_dataset.csv
Requesting LSTM model:
gs://tsa_final_project/tsa_final_project_folder/my_lstm_model.h
Application server (GCP Linux VM)
http://34.125.118.242:8080/
User
App.py
Python
Dash
HTML
Requesting UFO data and LSTM model
UFO data, LSTM model
GCP Storage bucket
Ufo_dataset.csv
My_lstm_model.h5

Environment preparation

In this step I prepared the environment for the application deployment:

On GCP I created a new project and under the project I created a storage bucket and a VM:

- VM: I created a VM "instance-1" and changed the configuration of the VM to allow HTTP traffic and applied firewall rule to allow connections on port 8080:



Free trial status: $298.95 credit and 84 days remaining - with a full account, you'll get unlimited access to all of Google Cloud Platform.

Compute Engine — instance-1

Connecting to serial ports is disabled

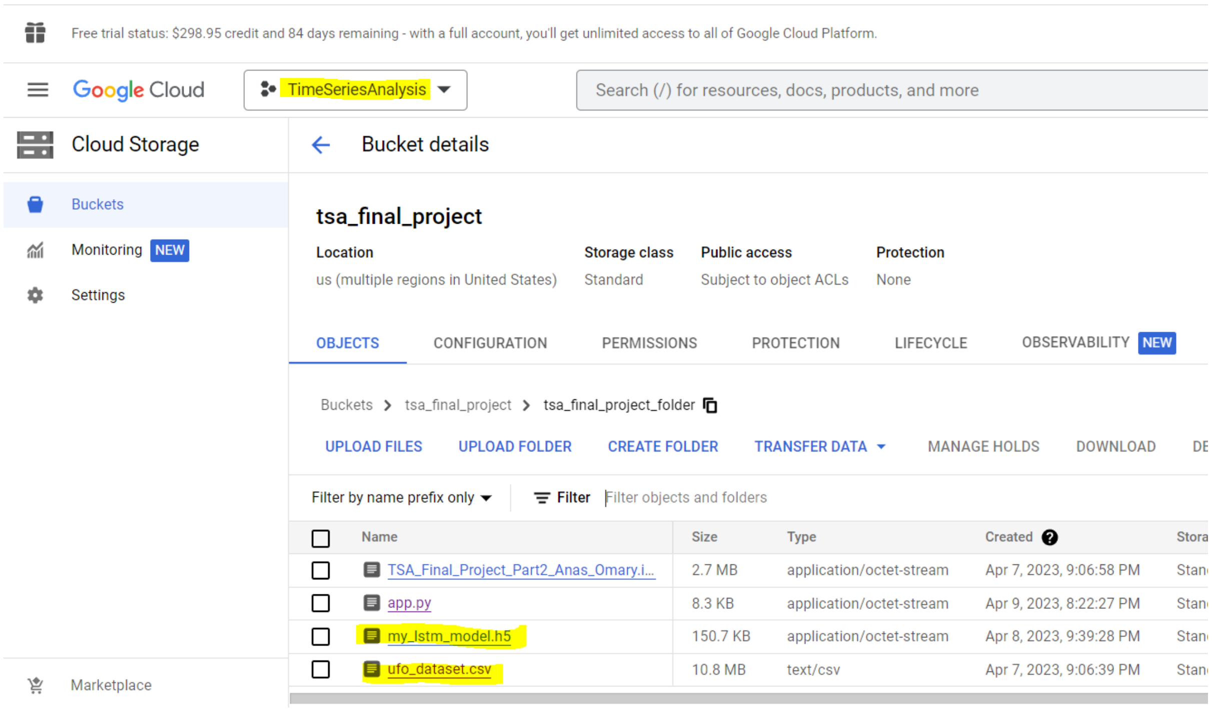
Logs

Logging
Serial port 1 (console)

Basic information

Name	instance-1
Instance Id	4866735305742190415
Description	None
Type	Instance
Status	Running
Creation time	Apr 7, 2023, 7:41:21 PM UTC+04:00

- I then created a storage bucket "tsa_final_project" and then created a folder inside the bucket "tsa_final_project_folder" and uploaded the dataset "ufo_dataset.csv" and the LSTM model "my_lstm_model.h5":



Free trial status: $298.95 credit and 84 days remaining - with a full account, you'll get unlimited access to all of Google Cloud Platform.

Cloud Storage — Bucket details

tsa_final_project

Location	Storage class	Public access	Protection
us (multiple regions in United States)	Standard	Subject to object ACLs	None

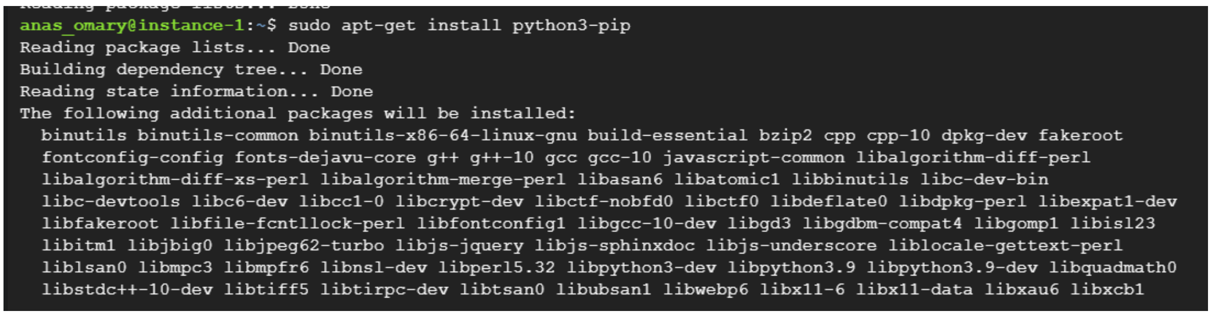
Buckets > tsa_final_project > tsa_final_project_folder

Name	Size	Type	Created	Stora
TSA_Final_Project_Part2_Anas_Omary.i...	2.7 MB	application/octet-stream	Apr 7, 2023, 9:06:58 PM	Stan
app.py	8.3 KB	application/octet-stream	Apr 9, 2023, 8:22:27 PM	Stan
my_lstm_model.h5	150.7 KB	application/octet-stream	Apr 8, 2023, 9:39:28 PM	Stan
ufo_dataset.csv	10.8 MB	text/csv	Apr 7, 2023, 9:06:39 PM	Stan

App deployment

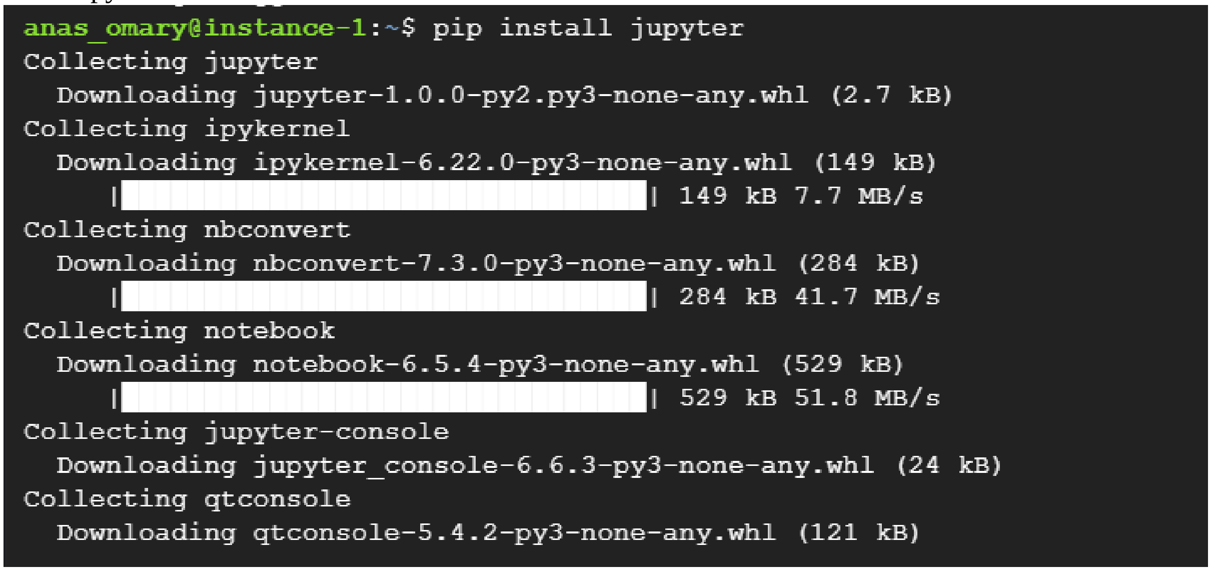
I prepared the VM and deployed app.py:

- Installed python:



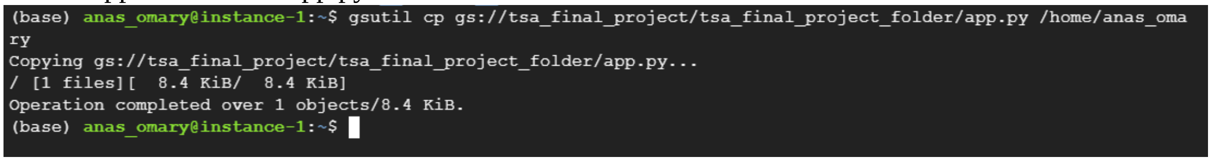
```
anas_omary@instance-1:~$ sudo apt-get install python3-pip
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  binutils binutils-common binutils-x86-64-linux-gnu build-essential bzip2 cpp cpp-10 dpkg-dev fakeroot
  fontconfig-config fonts-dejavu-core g++ g++-10 gcc gcc-10 javascript-common libalgorithm-diff-perl
  libalgorithm-diff-xs-perl libalgorithm-merge-perl libasan6 libatomic1 libbinutils libc-dev-bin
  libc-devtools libc6-dev libcc1-0 libcrypt-dev libctf-nobfd0 libctf0 libdeflate0 libdpkg-perl libexpat1-dev
  libfakeroot libfile-fcntllock-perl libfontconfig1 libgcc-10-dev libgd3 libgdbm-compat4 libgomp1 libisl23
  libitm1 libjbig0 libjpeg62-turbo libjs-jquery libjs-sphinxdoc libjs-underscore liblocale-gettext-perl
  liblsan0 libmpc3 libmpfr6 libnsl-dev libperl5.32 libpython3-dev libpython3.9 libpython3.9-dev libquadmath0
  libstdc++-10-dev libtiff5 libtirpc-dev libtsan0 libubsan1 libwebp6 libx11-6 libx11-data libxau6 libxcb1
```

- Installed Jupyter:



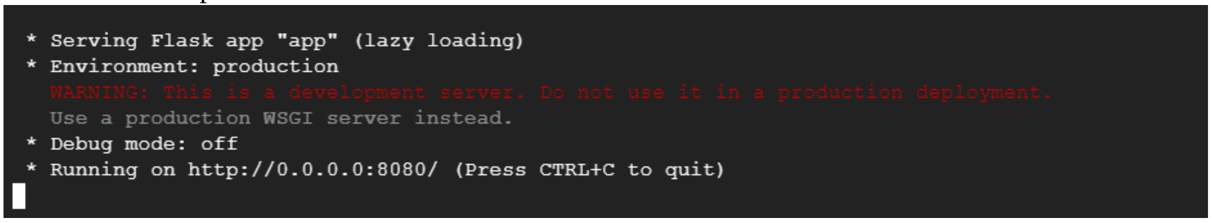
```
anas_omary@instance-1:~$ pip install jupyter
Collecting jupyter
  Downloading jupyter-1.0.0-py2.py3-none-any.whl (2.7 kB)
Collecting ipykernel
  Downloading ipykernel-6.22.0-py3-none-any.whl (149 kB)
     |████████████████████████████████| 149 kB 7.7 MB/s
Collecting nbconvert
  Downloading nbconvert-7.3.0-py3-none-any.whl (284 kB)
     |████████████████████████████████| 284 kB 41.7 MB/s
Collecting notebook
  Downloading notebook-6.5.4-py3-none-any.whl (529 kB)
     |████████████████████████████████| 529 kB 51.8 MB/s
Collecting jupyter-console
  Downloading jupyter_console-6.6.3-py3-none-any.whl (24 kB)
Collecting qtconsole
  Downloading qtconsole-5.4.2-py3-none-any.whl (121 kB)
```

- Copied the application file “app.py” from the bucket into the VM:



```
(base) anas_omary@instance-1:~$ gsutil cp gs://tsa_final_project/tsa_final_project_folder/app.py /home/anas_oma
ry
Copying gs://tsa_final_project/tsa_final_project_folder/app.py...
/ [1 files][  8.4 KiB/  8.4 KiB]
Operation completed over 1 objects/8.4 KiB.
(base) anas_omary@instance-1:~$ █
```

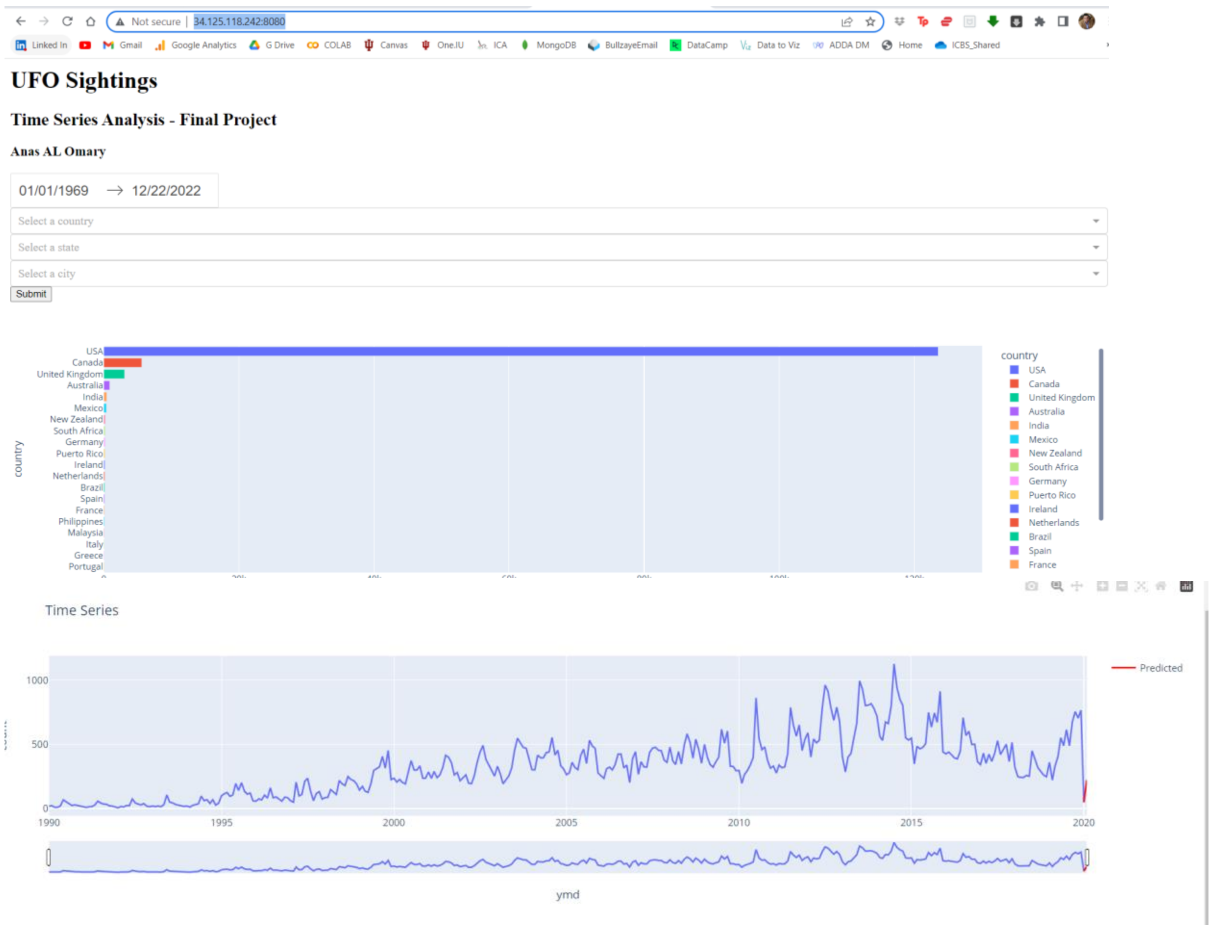
- Deployed the app.py using the command $ python app.py, I have configured app.py to run in production mode on port 8080



```
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://0.0.0.0:8080/ (Press CTRL+C to quit)
█
```

App Testing

I tested the app: after preparing the environment and deploying app.py, I tested the application to make sure it is accessible and runs fine:



UFO Sightings

Time Series Analysis - Final Project

Anas AL Omary

01/01/1969 → 12/22/2022

Select a country

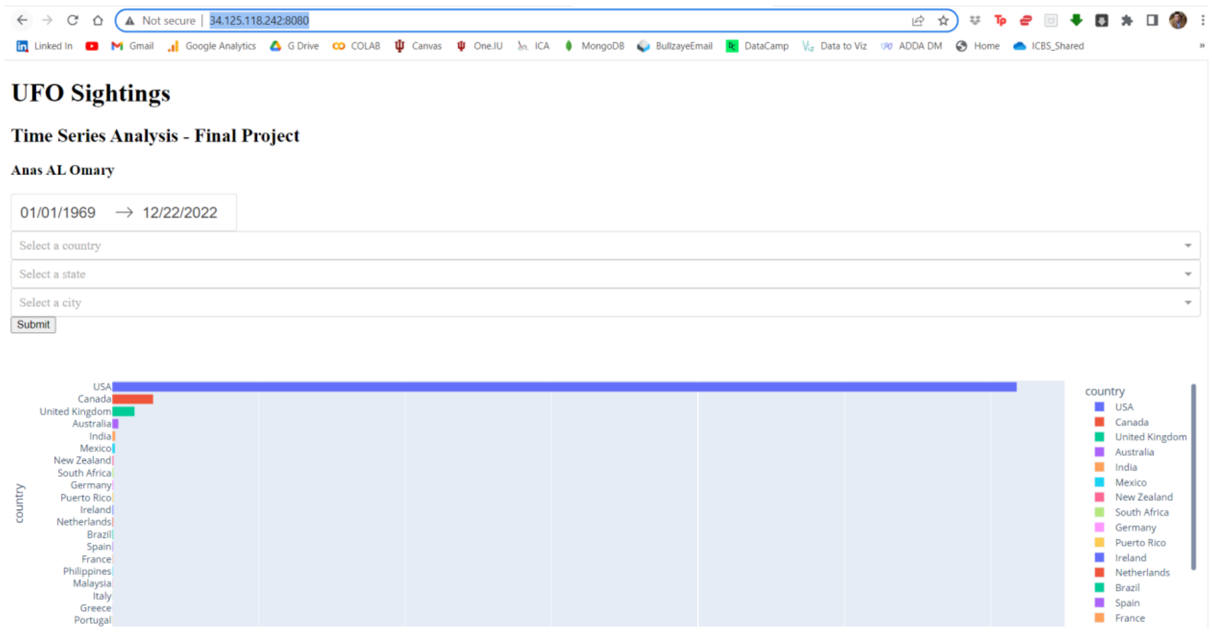
Select a state

Select a city

Submit

Part 2. App Layout

I have already created the web app and deployed it on GCP following is the application:



UFO Sightings

Time Series Analysis - Final Project

Anas AL Omary

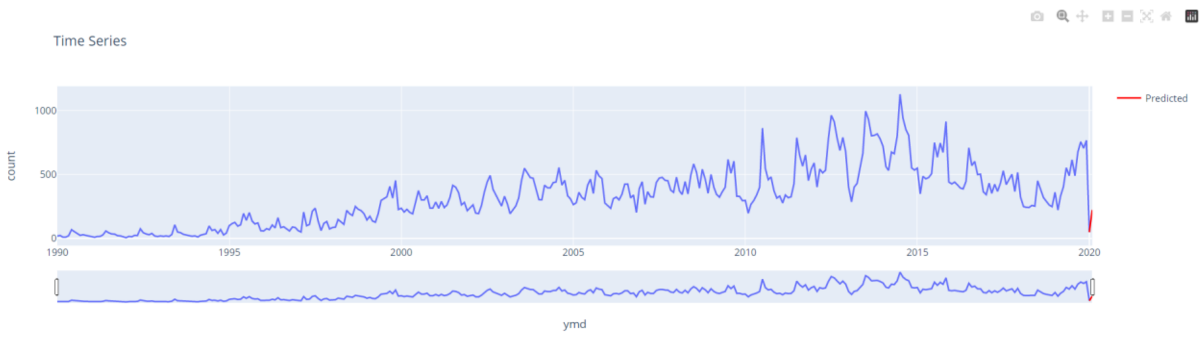
01/01/1969 → 12/22/2022

Select a country

Select a state

Select a city

Submit

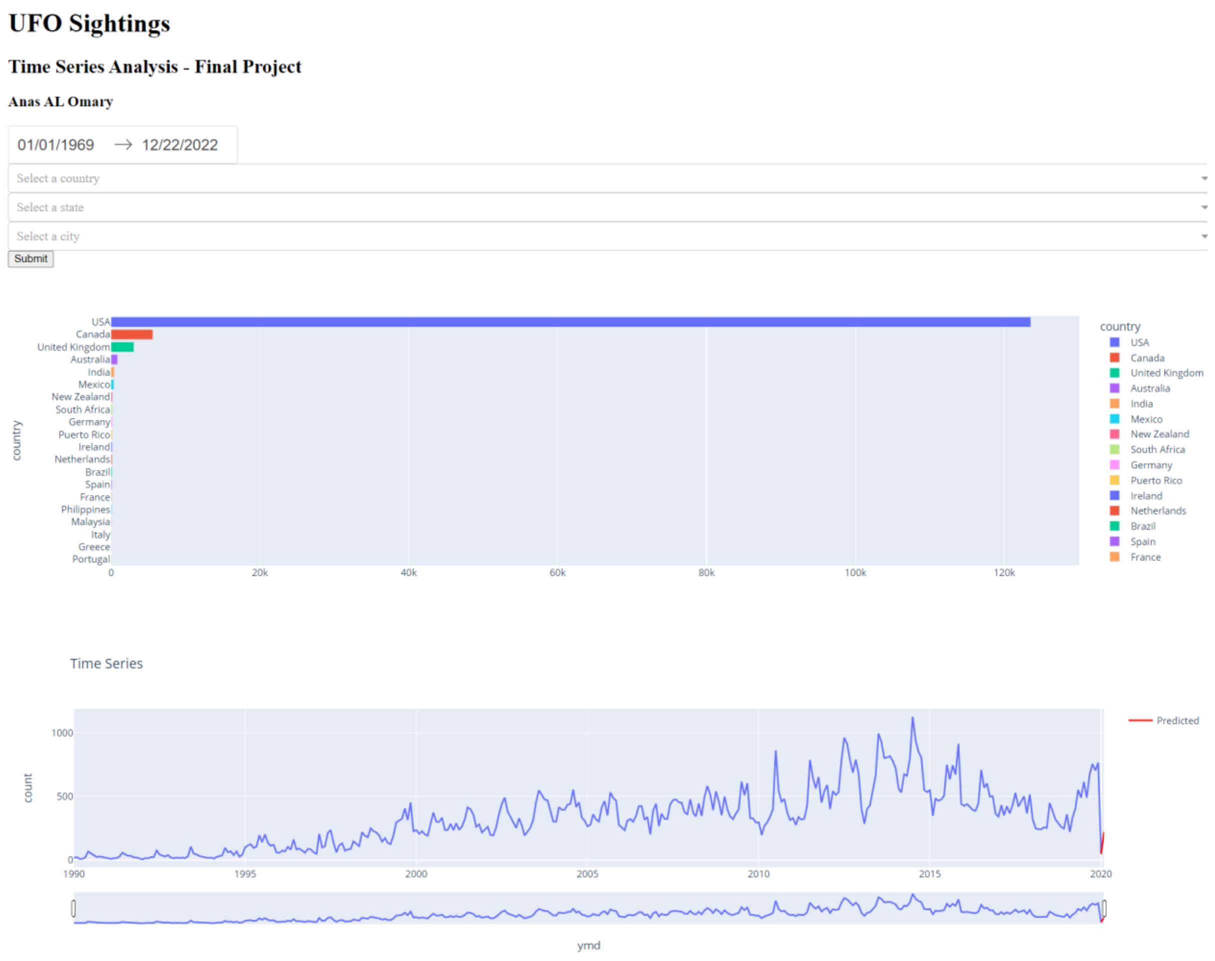


The user calls the application using the URL (http://34.125.118.242:8080/) then the web server serves the application app.py, the app consists of one page which has a header, date range picker, three drop-down lists and two graph elements:

```
app.layout = html.Div([
    html.H1('UFO Sightings'),
    html.H2('Time Series Analysis - Final Project'),
    html.H3('Anas AL Omary'),

    dcc.DatePickerRange(
        id='date-range-picker',
        min_date_allowed=df['date'].min(),
        max_date_allowed=df['date'].max(),
        initial_visible_month=df['date'].max(),
        start_date=df['date'].min(),
        end_date=df['date'].max(),
        #display_format='MMM Do, YYYY'
    ),

    dcc.Dropdown(
        id='country-dropdown',
        options=[{'label': country, 'value': country} for country in
df.groupby(['country']).size().reset_index(name='counts').sort_val-
ues(by='counts', ascending=False).head(15)['country']],
        value=None,
        placeholder='Select a country'
    ),

    dcc.Dropdown(
        id='state-dropdown',
        options=[{'label': state, 'value': state} for state in
df.groupby(['state']).size().reset_index(name='counts').sort_val-
ues(by='counts', ascending=False).head(15)['state']],
        value=None,
        placeholder='Select a state'
    ),

    dcc.Dropdown(
        id='city-dropdown',
        options=[{'label': city, 'value': city} for city in
df.groupby(['city']).size().reset_index(name='counts').sort_val-
ues(by='counts', ascending=False).head(30)['city']],
        value=None,
        placeholder='Select a city'
    ),

    html.Button(id='submit-button', n_clicks=0, children='Submit'),

    dcc.Graph(id='counts-graph'),

    dcc.Graph(id='timeseries-graph')
])
```

When the app first loads, it shows the date range picker with date period between January first 1969 and December 22 2022, then it loads data into the three drop down lists.

By default, the app loads the entire time series data from January first 1969 till December 22 2022 showing observations worldwide.



UFO Sightings

Time Series Analysis - Final Project

Anas AL Omary

01/01/1969 → 12/22/2022

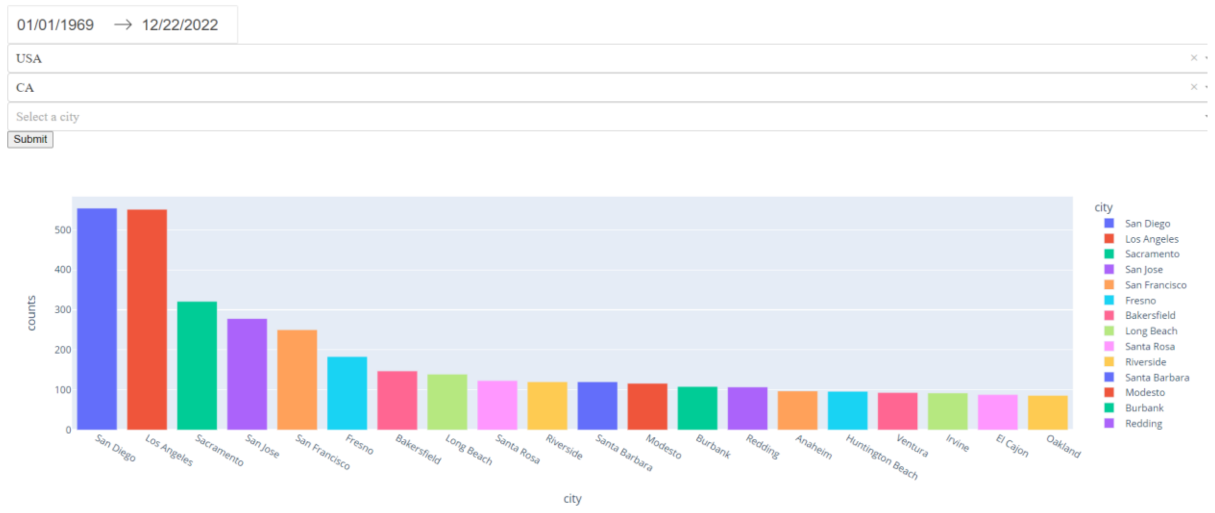
Select a country

Select a state

Select a city

Submit

The user can filter by a specific time period, country, state and city, then click on submit, this will apply the filter on the bar-chart accordingly, then it will apply the filter on the time series visualization and recalculate predictions.



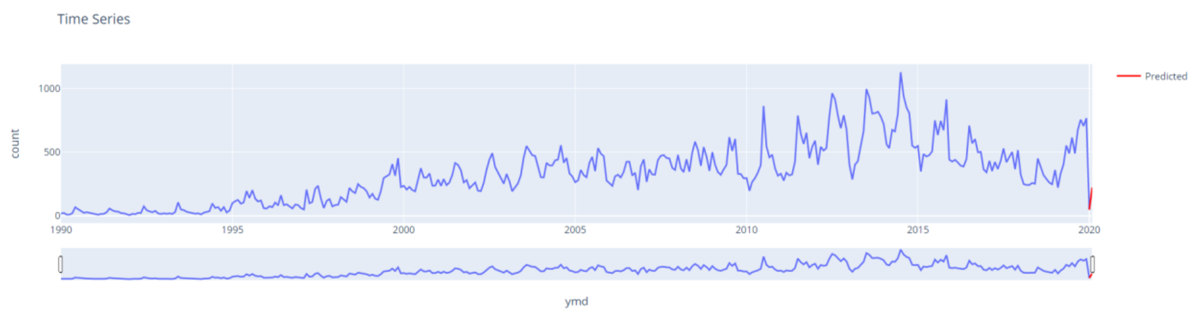
01/01/1969 → 12/22/2022

USA

CA

Select a city

Submit



Following code shows how I am loading the LSTM model which I saved in part 2 of the project, then I reshape the data to be suitable for the model then I generate predictions and include it in the time series visualization:

```
loaded_model = 
load_model(r'gs://tsa_final_project/tsa_final_project_folder/my_lstm_model.
h5', compile = False)
    #C:\Users\anaso\Google 
Drive\Indiana_Master_of_Data_Science\Year3_Spring\TimeSeriesAnalysis\FinalP
roject\Part3\my_lstm_model.h5
    df2.set_index('ymd',inplace=True)
    # Select the most recent 12 months of data
    input_data = df2[-12:]
    time_steps=12
    # Scale the input data using the same scaler used to train the model
    scaler = MinMaxScaler()
    scaled_input_data = scaler.fit_transform(input_data)
    # Reshape the input data to match the shape expected by the LSTM model
    reshaped_input_data = scaled_input_data.reshape((1, time_steps, 1))
    # Make a prediction using the loaded model
    predicted_value = loaded_model.predict(reshaped_input_data)
    # Rescale the predicted value to the original scale
    unscaled_predicted_value = scaler.inverse_transform(predicted_value)
    data = {'ymd': [df2.tail(1).index.values[0],(df2.tail(1).index+
pd.DateOffset(months=1)).values[0]],
        'count': 
[df2.tail(1)['count'].values[0],unscaled_predicted_value[0][0]]}
    dfpred = pd.DataFrame(data)
    timeseries_df.reset_index(inplace=True)
    timeseries_fig = px.line(timeseries_df, x='ymd', y='count', title='Time 
Series')
    timeseries_fig.add_scatter(x=dfpred['ymd'], y=dfpred['count'], 
mode='lines', name='Predicted', line=dict(color='red'))
    timeseries_fig.update_xaxes(rangeslider_visible=True)
```

Part 3. Team Work

This is an individual project, and while I was designing the application I decided to go ahead and build the app locally on my machine, then I ended up deploying it on GCP, in the next phase, I will test with different design themes and further test the application and validate the predictions.