Scraping data from Wikipedia

# Introduction

As my Capstone project I have chosen this topic as working in the data field I know that sometimes its tedious to not able to get the data in a proper format so that we can analyse it. Hence as an application of what I have learned I have written this code. Although the project seems on a small scale, I chose to work on this because I wanted to write the code from scratch and this was a feasible option.

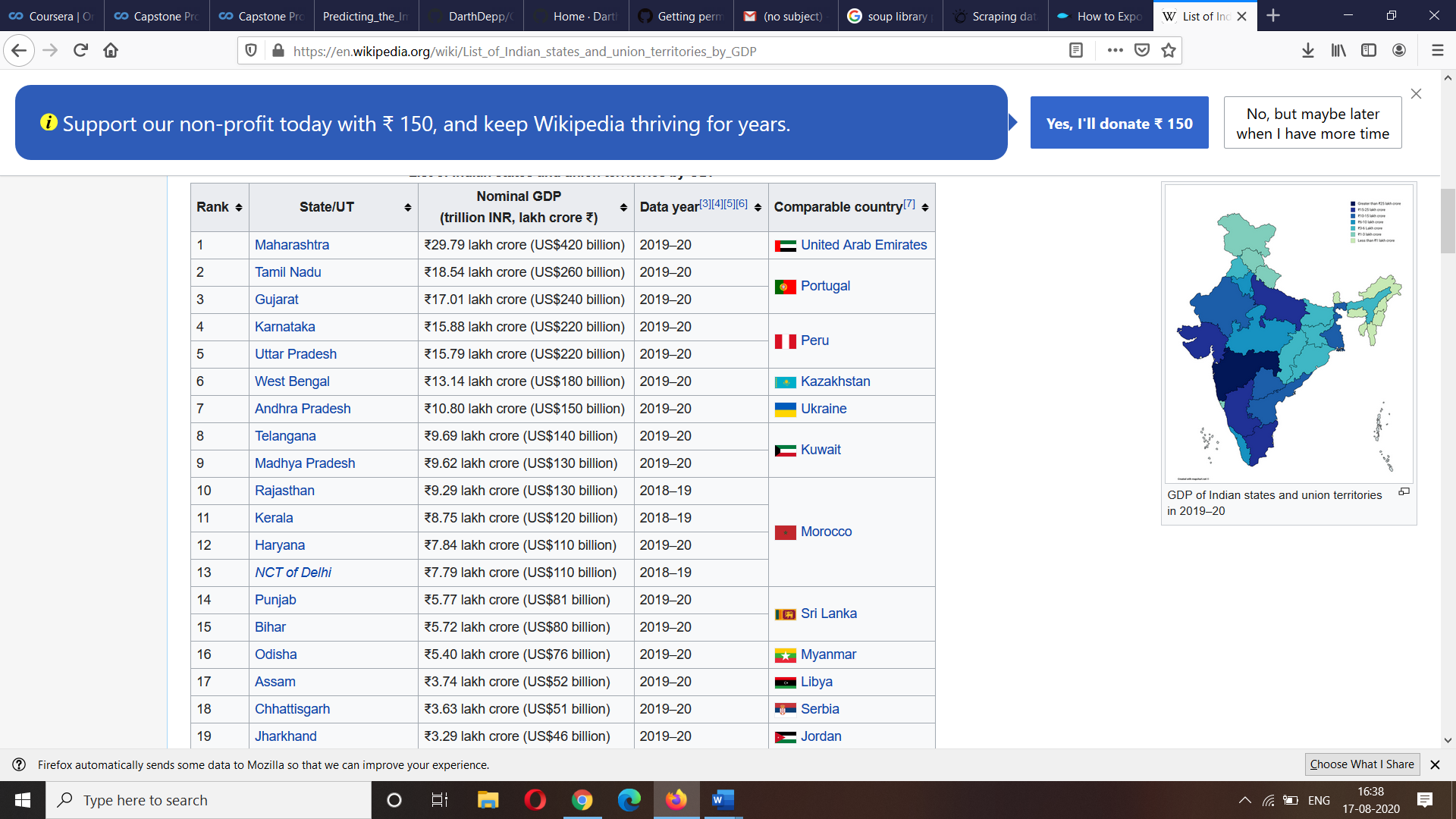
# Objective

To scrape data from a Wikipedia page through which I will convert raw data into structured data.

# Methodology

I chose the Indian states by its GDP page. The page classifies Indian states based on their GDP and also compares them with respective countries. For this I used libraries like numpy, pandas, beautiful soup etc.

## Data



The data I used was in a very raw format and extracting it in a proper structure required coding with a sound basics of List and Arrays.

# Code

import numpy as np # library to handle data in a vectorized manner

import pandas as pd # library for data analsysis

pd.set\_option("display.max\_columns", None)

pd.set\_option("display.max\_rows", None)

import json # library to handle JSON files

from geopy.geocoders import Nominatim # convert an address into latitude and longitude values

import requests # library to handle requests

from bs4 import BeautifulSoup # library to parse HTML and XML documents

from pandas.io.json import json\_normalize # tranform JSON file into a pandas dataframe

# Matplotlib and associated plotting modules

import [matplotlib.cm](http://matplotlib.cm) as cm

import matplotlib.colors as colors

# import k-means from clustering stage

from sklearn.cluster import KMeans

print("Libraries imported.")

# send the GET request

data = requests.get('<https://en.wikipedia.org/wiki/List_of_Indian_states_and_union_territories_by_GDP>').text

# parse data from the html into a beautifulsoup object

soup = BeautifulSoup(data, 'html.parser')

# create three lists to store table data

StateList = []

GDPList = []

ComparableCountryList = []

# find the table

gdp = soup.find\_all('table')[1].find\_all('tr')

row\_list = list()

#

# append the data into the respective lists

for row in gdp:

td = row.find\_all('td')

row = [i.text.replace(u'\u20b9', '').replace(u'\xa0', '').replace(u'\u2013','') for i in td]

row\_list.append(row)

# if(len(cells) > 0):

# StateList.append(cells[0].text)

# GDPList.append(cells[1].text)

# ComparableCountryList.append(cells[2].text.rstrip('\n'))

print(row\_list)

df\_bs = pd.DataFrame(row\_list, columns=['Rank','State/UT','Nominal GDP','Data year','Comparable country'])

df\_bs.set\_index('State/UT',inplace=True)

# df\_bs.to\_csv('beautifulsoup.csv')

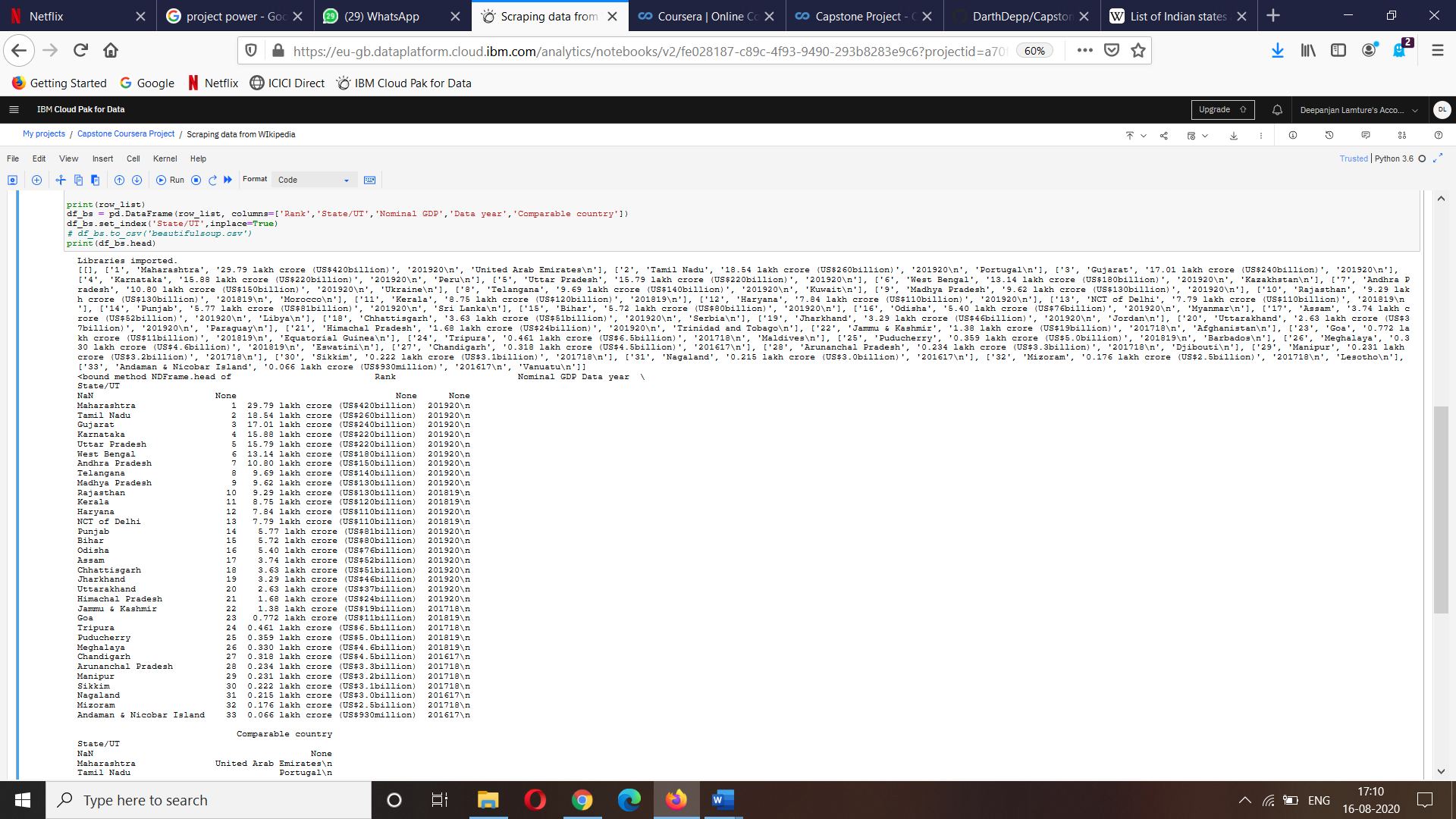
print(df\_bs.head)

# Problems

During the code I faced many problems where I was unable to create a proper array as well as list, but constant trial and error approach helped me to reach my final goal. I also faced some ASCI characters issue so I had to install some directories. Researching more about this topic I found out that this can be done easily using Selenium. So maybe going further in this domain I will try to explore more possibilities through Selenium.

# Results

I was able to load the data in a structured data frame that too including the column I needed as well as excluding the ones which were not needed.



# Conclusion

This code can also be converted into a tool and deployed so that scraping data from the website becomes easy. It will need slight modifications while deploying for other websites. Scraping tools are more in demand since the data boost due to internet. Tools like these will help businesses extract more data which will ultimately help in better decision making.