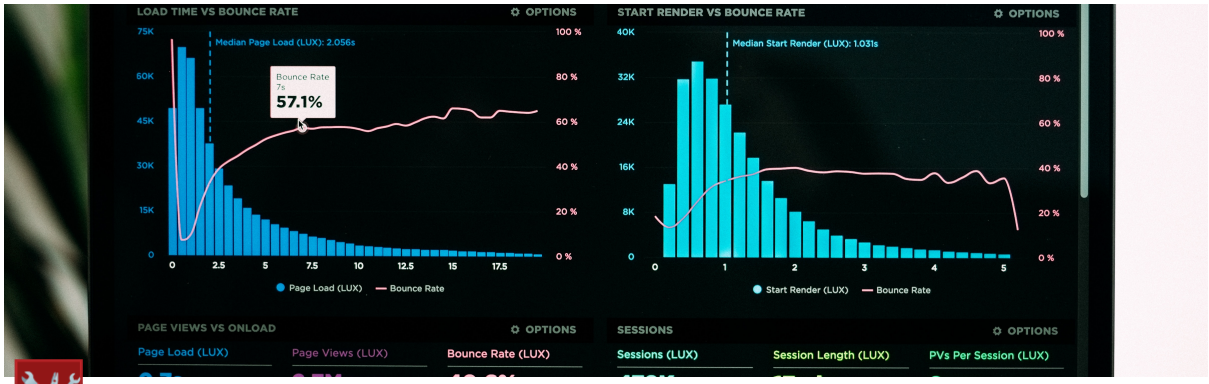

Tools for Data Science Week 2 and 3

For enhanced type setting visit the webpage for the notion note on

Tools for Data Science Week 2 and 3

Important concepts

- [ ] Pandas (pd)
- [ ] Beautiful Soup (bs4)
- [ ] Pivot Tables
- [ ] requests

Week 2

L2.1:Get the Data - Introduction

https://www.youtube.com/watch?v=1LyblMkJzOo&list=PLZ2ps__7DhBZJ2q_hd8ZbDRgOJlB0CZLw&index=3

L2.2: Get the data - Nominatim Open Street Maps (OSM)

https://www.youtube.com/watch?v=f0PZ-pphAXE&list=PLZ2ps__7DhBZJ2q_hd8ZbDRgOJlB0CZLw&index=4

- Geocoding API - Nominatim - Open Street Maps (OSM)

```
#import nominatim api
from geopy.geocoders import Nominatim

#activate nominatim geocoder
locator = Nominatim(user_agent="myGeocoder")
#type any address text
location = locator.geocode("Champ de Mars, Paris, France")

print("Latitude = {}, Longitude = {}".format(location.latitude, location.longitude))

#the API output has multiple other details as a json like altitude, lattitude, longitude, correct raw addres, etc.
#printing all the informaton
location.raw,location.point,location.longitude,location.latitude,location.altitude,location.address
```

L2.3: Get the data - BBC Weather location service ID Scraping

https://www.youtube.com/watch?v=IafLrvnamAw&list=PLZ2ps__7DhBZJ2q_hd8ZbDRgOJlB0CZLw&index=5

L2.4: Get the data - Scraping with Excel

https://www.youtube.com/watch?v=OCl6UdpmzRQ&list=PLZ2ps__7DhBZJ2q_hd8ZbDRgOJlB0CZLw&index=6

L2.5: Get the data - Scraping with Python

https://www.youtube.com/watch?v=TTzcXj92zaw&list=PLZ2ps__7DhBZJ2q_hd8ZbDRgOJlB0CZLw&index=7

```
from bs4 import BeautifulSoup as bs
import requests #to access website
import pandas as pd

r = requests.get("https://www.imdb.com/chart/top/")

# Convert to a beautiful soup object
soup = bs(r.content)

# Print out HTML
contents = soup.prettify()
print(contents[:100])

for row in movie_titlecolumn:
    title = row.a.text # tag content extraction
    movie_title.append(title)
movie_title

# Creating a Dataframe
movie_df = pd.DataFrame({'Movie Title': movie_title, 'Year of Release': movie_year, 'IMDb Rating': movie_rating})
movie_df.head(30) #View first 30 rows of dataframe
```

L2.6: Get the data - Wlkimedia

https://www.youtube.com/watch?v=b6puvm-QEY0&list=PLZ2ps__7DhBZJ2q_hd8ZbDRgOJlB0CZLw&index=8

```
import wikipedia as wk

print(wk.search("IIT Madras"))
print(wk.summary("IIT Madras"))

# Full Page
full_page = wk.page("IIT Madras")
print(full_page.content)

# Extracting Tables

#extract html code of wikipedia page based on any search text
html = wk.page("IIT Madras").html().encode("UTF-8")

import pandas as pd
df = pd.read_html(html)[6]

df
```

Google Colaboratory

https://colab.research.google.com/drive/1UZky5JdOn2oMYIkls23WefTaT8VinYyg#scrollTo=ovcFMuFDDN06

L2.7: Get the data - Scrape BBC weather with Python

https://www.youtube.com/watch?v=Uc4DgQJDRoI&list=PLZ2ps__7DhBZJ2q_hd8ZbDRgOJlB0CZLw&index=9

- Querying data in the form of an **API**
  - Go to network and find the API information
  - use URLEncode

```
from urllib.parse import urlencode
import requests               # to get the webpage
from bs4 import BeautifulSoup # to parse the webpage

import pandas as pd
import re                     # regular expression operators
```

- use of Pandas started here for the first time
- Some Important lines of code here are

```
url      = 'https://www.bbc.com/weather/'+result['response']['results']['results'][0]['id']
response = requests.get(url)

#using beautifulsoup finally
soup = BeautifulSoup(response.content,'html.parser')

# using text.strip().split()
daily_high_values_list = [daily_high_values[i].text.strip().split()[0] for i in range(len(daily_high_values))]
daily_high_values_list

#split the string on uppercase
daily_summary_list = re.findall('[a-zA-Z][^A-Z]*', daily_summary.text)
```

- Using **zip and Pandas**

```
zipped = zip(datelist, daily_high_values_list, daily_low_values_list, daily_summary_list)
df = pd.DataFrame(list(zipped), columns=['Date', 'High','Low', 'Summary'])
display(df)

# Converting to csv / excel file
filename_csv = location.text.split()[0]+'.csv'
df.to_csv(filename_csv, index=None)
filename_xlsx = location.text.split()[0]+'.xlsx'
df.to_excel(filename_xlsx)
```

L2.8: Get the data - Scraping PDFs

https://www.youtube.com/watch?v=3Xw9YGh00aM&list=PLZ2ps__7DhBZJ2q_hd8ZbDRgOJlB0CZLw&index=10

- Scraping data from PDF Files
- Beautiful Soup Implementation

```
import os
import requests
import urllib.request
import pandas as pd
from urllib.parse import urljoin
from bs4 import BeautifulSoup

response = requests.get(url)
soup = BeautifulSoup(response.text, "html.parser")

# Loop through all PDF links in the page
for link in soup.select("a[href$='.pdf']"):
    # Local lile name is the same as PDF file name in the URL (ignoring rest of the path)
    # https://premierleague-static-files.s3.amazonaws.com/premierleague/document/2016/07/02/e1648e96-4eeb-456e-8ce0-d937d2bc7649/2011-
    filename = os.path.join(folder_location, link['href'].split('/')[-1])
    with open(filename, 'wb') as f:
        f.write(requests.get(urljoin(url,link['href'])).content)

from google.colab import drive
drive.mount('/content/drive')

# Save contents from url into folder_location
url = 'https://www.premierleague.com/publications'
folder_location = r'/content/drive/MyDrive/Colab Notebooks/premier_league'
if not os.path.exists(folder_location):
    os.mkdir(folder_location)
```

Tabula

```
# Tabula scrapes tables from PDFs
!pip install tabula-py
import tabula

tabula.read_pdf(combined_pdf,  pages='18')

from tabula import convert_into

convert_into(combined_pdf, folder_location +"/table_output.csv", output_format="csv",pages = 18,area=[[275,504,640,900]])
pd.read_csv(folder_location+"/table_output.csv")
```

Week 3

L 3.1: Prepare the Data

https://www.youtube.com/watch?v=dF3zchJJKqk&list=PLZ2ps__7DhBZJ2q_hd8ZbDRgOJlB0CZLw&index=11

- **How can you use tools to get data to the form that you want, and clean them up.**

- To begin with, we will be looking at number 1, what are the tools you can use to load and preview the data. Specifically, we will be looking at excel, and we will be looking at pandas profiling. Second, you look at how you can create derived metrics from that data, add new columns that will give you new information, transform the data in different ways and you will be looking at Google Sheets, you will be looking at excel as a tool
- you will also be looking at **Trifacta's** wrangler as another tool that will help you do this.
- We won't restrict ourselves to just text, you will also learn how to transform image data using the Python image library or the newer version, which is **Pillow**.
- And finally, you will also learn how to clean or collect missing data, which you can do with libraries like **Tabula** for PDF files, which will help you extract tables, you can do with tools like **OpenRefine**, which will help you work with structured data and correct spelling mistakes for example.
- And you will also learn how to do **image labeling** for images using simple tools like excel.
- To recap, this module will give you the tools that will hopefully provide you a competitive edge when it comes to taking data that is raw and converting it to data that is useful for analysis.

L 3.2: Prepare the data: Data Aggregation

https://www.youtube.com/watch?v=NkpT0dDU8Y4&list=PLZ2ps__7DhBZJ2q_hd8ZbDRgOJlB0CZLw&index=12

L 3.3: Prepare the data: Cleaning with Excel

https://www.youtube.com/watch?v=2n1qqEidxe0&list=PLZ2ps__7DhBZJ2q_hd8ZbDRgOJlB0CZLw&index=13

L 3.4: Prepare the data: Data Pandas Profiling

https://www.youtube.com/watch?v=CDwZPie29QQ&list=PLZ2ps__7DhBZJ2q_hd8ZbDRgOJlB0CZLw&index=16

Using Pandas Profiling Library!

```
!pip install pandas_profiling==2.9.0
from pandas_profiling import ProfileReport
import pandas as pd
from google.colab import files
```

Using the url straight from google drive

```
url='https://drive.google.com/file/d/1KjrSid8AfVggkCX3pmcpRE-OAi8CHVUr/view?usp=sharing'
url2='https://drive.google.com/uc?id=' + url.split('/')[-2]
df = pd.read_csv(url2,encoding='latin-1')
df

prof = ProfileReport(df)
prof.to_file('report.html')
files.download('report.html')
```

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/bf51d342-354d-473c-bde3-5da76b049689/report.html

file:///Users/kautukdoshi/Downloads/SeeyaMac/report.html

- Statistics
- Histogram
- Common Values
- Extreme Values (min5, max5)
- Missing data
- Warnings
- 5 types of Correlation - their matrix
- Interactions

L 3.5: Prepare the data: Cleaning with OpenRefine

https://www.youtube.com/watch?v=cX_2MkShlJk&list=PLZ2ps__7DhBZJ2q_hd8ZbDRgOJlB0CZLw&index=17

- Google project for cleaning data
- For e.g. Ltd. = limited = ltd (without fullstop)
- OpenRefine is downloadable - then runs as a localhost
- csv file and upload

What can you do after creating a project?

1. **Create project**
2. Run clustering from drop down menu
3. **Facet —> Text Facet**
4. Key Collision clustering
5. You can browse the cluster - similar in spelling but **special characters and spacing and all might be different**
6. Merge selected and re-cluster

L 3.6: Prepare the data: Image Labelling

https://www.youtube.com/watch?v=9b5ZvIRFCek&list=PLZ2ps__7DhBZJ2q_hd8ZbDRgOJlB0CZLw&index=14

Downloading and then labelling the data

Code for downloading images

```
headers = {
    "User-Agent":
    "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.102 Safari/537.36 Edge/18.19582
}

params = {
    "q": "chess pawn",
    "sourceid": "chrome",
}
query_term = params['q']
html = requests.get("https://www.google.com/search", params=params, headers=headers)
soup = BeautifulSoup(html.text, 'lxml')

for result in soup.select('div[jsname=dTDiAc]'):
    link = f"https://www.google.com{result.a['href']}"
    being_used_on = result['data-lpage']
    print(f'Link: {link}\nBeing used on: {being_used_on}\n')

# finding all script (<script>) tags
```

```
script_img_tags = soup.find_all('script')

# https://regex101.com/r/L3IZXe/4
img_matches = re.findall(r"s='(.*?)';", str(script_img_tags))

for index, image in enumerate(img_matches):
    try:
        # https://stackoverflow.com/a/6966225/15164646
        final_image = Image.open(BytesIO(base64.b64decode(str(image))))

        # https://www.educative.io/edpresso/absolute-vs-relative-path
        # https://stackoverflow.com/a/31434485/15164646
        final_image.save(f'{query_term}_{index}.jpg', 'JPEG')
    except:
        pass
```

Use of window box macros in excel and more

L 3.7: Prepare the data: Cleaning with OpenRefine2

https://www.youtube.com/watch?v=zguYP_cUC6g&list=PLZ2ps__7DhBZJ2q_hd8ZbDRgOJlB0CZLw&index=18

- **Text Facet**
- Keying function fingerprint
- Methods
  - **Key Collision :** It is the most stringent algorithm. So, what essentially it does is it removes the special characters in the text, it converts them all to lowercase, and then does the clustering.
  - **Nearest Neighbour: levenshtein distance -** the number of edits which needs to be done **between two strings to make the both the strings same.** Essentially number of edits is the distance
  - Even more lenient **ppm partial matching** : what essentially happens in partial matching is if any of the subtext or sub words are matching with the cluster, they are grouped together.
- User control
  - Select all
  - Deselect some
  - Rename multiple entries in a cluster easily

L 3.8: Data Transformation: Excel

https://www.youtube.com/watch?v=gR2IY5Naja0&list=PLZ2ps__7DhBZJ2q_hd8ZbDRgOJlB0CZLw&index=15

Pivot tables and Pivot Charts

Credits: Kautuk D aka @winterrolls