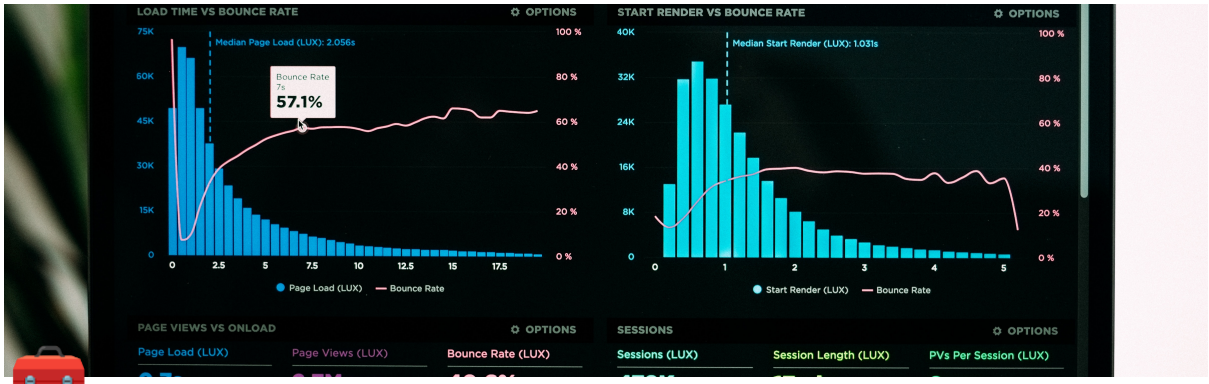

Tools for Data Science Notes

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

🧰 Tools for Data Science Notes

I have videos for weeks 2 and 3 for the open book midterms.

▼ Important concepts in Week 2 and 3 For MidTerm

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

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 - WIkimedia

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

- T**ext 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

Week 4

- Excel
    - Correlation
    - Regression
    - Outlier Detection
- Python
    - Classification
    - Forecasting
    - Clustering
- Others
    - R / RStudio
    - Rattle
    - PyCaret

L4.1: Model the Data: Introduction

L4.3: Model the data: Correlation with Excel

- Data Analysis Tool Pack from addins
- Correlation Matrix

L4.4: Model the data: Regression with Excel

- See adjusted R for % modelled relation
- Data Analysis tool pack, regression

L4.5: Model the data: Forecasting with Python

- Pandas, NumPy, and Matplot, for plotting visualizations, and then we are using a library from **SKLearn called the mean absolute error** to actually diagnose the forecasting error of the techniques that we are using.
- we are using the **autocorrelation plot** and get there when we reach. And then, we also use the **ARIMA model from the stats models library** because - Autoregression for time series forecasting

- See Autocorrelation plots for seeing how much of a variable can be explained by itself!

```
from sklearn.metrics import mean_absolute_error
mae5day = mean_absolute_error(testdf['new_deaths'], testdf['moving_avg_5day'])
print(f'Mean absolute error from 5day moving average prediction: {mae5day}')

from pandas.plotting import autocorrelation_plot
autocorrelation_plot(data['new_cases'])
plt.show()

from statsmodels.tsa.arima_model import ARIMA
model = ARIMA(data.new_deaths, order=(1,1,0))
model_fit = model.fit()
model = ARIMA(history, order=(1,1,0))
model_fit = model.fit()
output = model_fit.forecast()

from statsmodels.graphics.tsaplots import plot_acf,plot_pacf
import statsmodels.api as sm
fig = plt.figure(figsize=(12,8))
ax1 = fig.add_subplot(211)
fig = sm.graphics.tsa.plot_acf(data['new_deaths'].dropna(),lags=50,ax=ax1)
```

L4.6: Model the data: Classification with Python

- Use OrdinalEncoder where order is important and LabelEncoder where order is not important
- Using SMOTE to counter imbalance in the data
- ARIMA was for Autoregression

```
from sklearn.preprocessing import OrdinalEncoder, LabelEncoder
encoder = OrdinalEncoder(categories=[['Student', 'Pensioner', 'Working','Commercial associate','State servant']])
df.NAME_INCOME_TYPE = encoder.fit_transform(df.NAME_INCOME_TYPE.values.reshape(-1, 1))

from sklearn.preprocessing import MinMaxScaler
encoder2 = LabelEncoder()
df[i] = encoder2.fit_transform(df[i].values.reshape(-1, 1))

from imblearn.over_sampling import SMOTE, BorderlineSMOTE, ADASYN
sm = SMOTE(random_state = 42)

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)
mms = MinMaxScaler()

from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier()
model = classifier.fit(X_train_res, y_train_res) #DecisionTree Classifier has predict() function

import seaborn as sns
sns.scatterplot(x='ID', y='CNT_CHILDREN', data=df, ax=ax[0][0], color= 'orange')
sns.scatterplot(x='ID', y='AMT_INCOME_TOTAL', data=df, ax=ax[0][1], color='orange')

from sklearn.metrics import accuracy_score, confusion_matrix
print('Accuracy Score is {:.5}'.format(accuracy_score(y_test, prediction)))
print(pd.DataFrame(confusion_matrix(y_test,prediction)))

from sklearn.metrics import classification_report
model = classifier.fit(X_train_res, y_train_res)
prediction = model.predict(X_test_scaled)
classification_report(y_test, prediction)
```

Week 5

L5.1: Model the data: Pycaret

Automates the task of cleaning and modelling data - e.g. for Binary classification

```
import pycaret
from pycaret.datasets import get_data
index = get_data('index')

from pycaret.classification import *
```

```
clf1 = setup(data, target = 'Purchase', session_id=123, log_experiment=True, experiment_name='juice1', normalize = True, feature_selec

training_data = get_config(variable="X_train")

models()
best_model = compare_models()

rf = create_model('rf', fold = 5) #rf is random forest model
tuned_rf = tune_model(rf)

plot_model(tuned_rf)
plot_model(tuned_rf, plot = 'confusion_matrix')
plot_model(tuned_rf, plot = 'feature')
interpret_model(tuned_rf)
save_model(tuned_rf, model_name='best-model')
```

L5.2: Model the data: Clustering with Python

- KMeans clustering, after pre-processing using MinMaxScaler

```
# Good idea to standardize the features before k-Means
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
stockDataFeatures_scaled = scaler.fit_transform(stockData[features])
stockDataFeatures_scaled = pd.DataFrame(stockDataFeatures_scaled, columns=features)
stockDataFeatures_scaled.describe()

from sklearn.cluster import KMeans
kmeans = KMeans(7, n_jobs=-1)
clus = kmeans.fit_predict(stockDataFeatures_scaled)

clusterDesc = pd.DataFrame(stockData.iloc[:,2:].groupby('cluster').mean().round(3))
clusterDesc.insert(0,'size',stockData['cluster'].value_counts())
```

L5.3: Model the data: Image classification using Keras

https://www.youtube.com/watch?v=l81xLqR8tjg&list=PLZ2ps__7DhBZJ2q_hd8ZbDRgOJlB0CZLw&index=25

Google Colaboratory

https://colab.research.google.com/drive/1NzFigdeY2dCqqFArO6VDBShWsImAGabF

L5.4: Model the data: Image classification using Google AutoML

- Using Google cloud platform (GCP)
- Vision API - **AutoML Vision** used here

1. Ask for Data Set
2. Create a new Data set
    a. Single Label
    b. Multi Label
    c. Object Detection
3. Upload Images - folder structure with images - **zip form** - create new bucket
4. Location - Region - uscentral1(iowa) - Rest of the options as default
5. See images
6. Then go to train
    a. Train new model
    b. Precision and Recall

7. Full evaluation
    a. PR Curve
    b. Confusion Matrix
    c. Great Precision and Recall
8. Direct deploy option
9. No need to code, simple drag and drop tool built by GCP

Week 6

L6.1: Design the output

General

- Excel
- Google Data Studio
- Power BI
- Tableau

Specialised

- Excel VBA
- Flourish Studio
- Kumu.io
- QGIS

L6.2: Excel Forecasting Visualization

- Sparklines
- growth(knowny, knownx, new value)
- = growth (parameter, date in number format, next date)gf
- = stdev(parameter) / mean
- Correl (range1, range2)
- Correlation matrix - Data Analysis Toolpack

L6.3: Modern tools to simplify deep learning models: Sentiment Analysis with Excel

Azure ML

L6.4: Modern tools to simplify deep learning models: Text classification with Python

```
from textblob import TextBlob
data['TextBlob_Subjectivity'] = data['review'].apply(lambda x: TextBlob(x).sentiment.subjectivity)
data['TextBlob_Polarity'] = data['review'].apply(lambda x: TextBlob(x).sentiment.polarity)

from sklearn.metrics import classification_report
print(classification_report(data['sentiment'], data['TextBlob_Analysis']))
```

L6.5: Geospatial analysis with Excel

Starbucks and McDonald stores...

Data - Map data - Format as **Geography** - Excel - Latitude / Longitude and so on

Make Country, map type data

L6.6: Modern tools to simplify deep learning models: Geospatial Analysis with Python

```
import folium
import geopy.distance

#Compute distance of every store from city center
distances_km = []

for row in df.itertuples(index=False):
   distances_km.append(
       geopy.distance.distance(NY_coord, row.Coordinate).km
   )

df['Distance'] = distances_km
df.head(10)

#Empire State Building coordinates
m = folium.Map(location=[40.748488, -73.985238], zoom_start= 10)

#Place markers for the stores on the map
for i, row in df.iterrows():
  lat = df.at[i, 'lat']
  lng = df.at[i, 'lng']
  store = df.at[i, 'store']

  if store == 'McDonalds':
    color = 'blue'
  else:
    color = 'green'

  folium.Marker(location=[lat,lng], popup=store, icon= folium.Icon(color=color)).add_to(m)


#All stores at a distance greater/less than x kms
df[df['Distance'] > 10]
```

Week 7

L7.1: Getting Started

Downloading tableau

L7.2: Design your output - Getting Started with *Tableau*

- Import Data from tables
- Multiple tables can be used together also
- Select columns and rows for graphical output
- Lots of customisation options - labels, colours, size, text and all
- Seems to be like a tool made just for this
- Allows stuff like filters and all also
- Can directly add geographical data also - map - classify the geographical role (e.g. country)

L7.3: Design your output - Adding multiple data sources to *Tableau*

1. Go to Data Source Tab
2. Add a new connection (another csv file)
3. Then link both those files
4. Select a field which is equal to field in the other file
5. You can create your own variables in tableau too

L7.4: Design your output - Develop dynamic dashboard in *Tableau*

- New dashboard
- And from charts youve made
- Referencing them on one sheet
- Changing stuff in the main chart will change stuff for the dashboard also
- Tableau public doesnt allow saving files to our local repo - saved in the public repo - login required
- Tableau public online - access to everyone!

L7.5: Design your output- Tools for specialized visualizations- network of actors

Kumu.io

Really complicated stuff

CSR - Compressed Sparsed row format

```
!pip install scikit-network

import sknetwork.clustering
import sknetwork.utils
from scipy.sparse import csr_matrix

name = pd.read_csv('name.basics.tsv.gz', sep='\t', na_values='\\N', dtype={'birthYear': float}).set_index('nconst')[['primaryName', 'b

matrix = csr_matrix((data, (row, col)))
    square = matrix.T * matrix
    square.setdiag(0)
    square = square.tocoo()

algo = sknetwork.clustering.Louvain()
adjacency = sknetwork.utils.edgelist2adjacency(pairs_in)
labels = algo.fit_transform(adjacency)
clusters_in = pd.concat([
    cat_in.reset_index(),
    pd.Series(labels, name='cluster')], axis=1)

clusters_in = pd.concat([
    cat_in.reset_index(),
    pd.Series(labels, name='cluster'),
    pd.Series(clusters_in['index'].map(name_freq), name='freq'),
], axis=1)
clusters_in
```

L7.6: Modern tools to simplify deep learning models- Cluster the network of actors

Same as above!

L7.7: Design your output- Geospatial Analysis- Creating shapefiles with QGIS

- QGIS is a special tool downloadable from the internet
- Various Panels view
- Layers and Browser Panel

1. New Project
2. **Shape Files**
    a. diva-gis, free spatial data
    b. Geographical data
    c. Zip file - select only **.shp** files
3. These become different layers
4. You can add labels
5. Overlay shp file on world map
    a. **Add plug in**
    b. **Quick Map Services**
    c. **Web —> QuickMapServices —> OSM**

6. You can create your own shape file layer after selection geometric type, name and 2 variables for the same
    a. toggle editing in tools
    b. add polygons
    c. Left click like youre in Photoshop
    d. Save this shape file
    e. Others will be automatically created when you made this shape File

Week 8

- Numbers
- Visuals
- Text
- Illustrations

L8.2: Narrate your story : Narratives with excel

- Pivot tables
- vlookup
- if-else
- Sounds pretty much like a workaround

L8.3: Narrate your story : Smart Narratives with Power BI

- You can add dashboards and stuff here
- Add Smart Narratives
- **Right click on the graph and click Summarize**
- **Add dynamicaly changeable values (+value) - like questions**

L8.4: Narrate your story : Narratives with Quill on Tableau

- .trex file for add on into Tableau
- **Using Quill** - smart narratives
- Change chart type and stuff
- **objects -> Extension**
- Local files - > select the trex file
- Configure: Specify worksheet, fields and story (discrete, continuous, scatterplot..)
- narrative science- lots of extension settings too
- Select and deselect lines
- Story from Visualisations

L8.5: Narrate your story : Comic narratives with Google Sheets & Comicgen

This was neat

- **encodeURL** by adding stuff dynamically
- Add comics from comicgen: gramener

Week 10

Hosting and Deploying

https://github.com/rohithsrinivaas/streamlit-heroku

https://github.com/rohithsrinivaas/streamlit-heroku

Hosting Comparison

Heroku

- Platform as a Service
- Agile deployment for Ruby, Node.js, Clojure, Java, Python, Go and Scala.
- Run and scale any type of app.
- Total visibility across your entire app.
- Github integration
- Command Line Interface

Glitch

- User -friendly browser text editor
- Difficulty in scaling
- Low cache memory limit in runtime
- Github integration

Netlify

- Used for Static Web Hosting
- Global Network
- Instant Cache Validation
- continuous deployments
- HTML/CSS/JS backend
- Github Integration

Vercel

- Used for Static Web Hosting
- Low cache memory limit in runtime
- Continuous deployments
- Github Integration

Heroku

Easy to deploy because of various development languages like Python, Ruby, Scala etc ..

Scaling is extremely simple and straightforward

- Recommended for ready-to-go machine learning model deployment for real time model inference and visualizations

Netlify

Instant Cache Validations

Global Network

HTML/CSS/JS backend

- Recommended for making static websites

So basically speaking for our work, Heroku is best

***Credits*: Kautuk D aka @winterrolls**