Homework 2

Before you start: Read Chapter 3 Data Visualization and Chapter 4 Dimension Reduction in the textbook.

Note: Please make sure your plots are complete and presentable with a title, proper axis names, labels and legends if applicable.

Please enter the code along with your comments in the **TODO** sections.

Please refer to the **Hint** section if you do not know where to start.

Alternative solutions are welcomed.

Part 1: Advanced Data Visualization

Problem 1

Dataset: Mismanaged waste

Introduction: Jambeck et al. quantified municipal and plastic waste streams from coastal populations in 2010 with projections to the year 2025. The authors define mismanaged and inadequately managed waste as follows: "mismanaged waste is material that is either littered or inadequately disposed. Inadequately disposed waste is not formally managed and includes disposal in dumps or open, uncontrolled landfills, where it is not fully contained. Mismanaged waste could eventually enter the ocean via inland waterways, wastewater outflows, and transport by wind or tides. "

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
!pip install --upgrade openpyxl
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: openpyxl in /usr/local/lib/python3.8/dist-packages (3.0.10)
Collecting openpyxl
  Downloading openpyxl-3.1.0-py2.py3-none-any.whl (250 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 250.0/250.0 KB 3.3 MB/s eta 0:00:00
Requirement already satisfied: et-xmlfile in /usr/local/lib/python3.8/dist-packages (from openpyxl) (1.
Installing collected packages: openpyxl
  Attempting uninstall: openpyxl
    Found existing installation: openpyxl 3.0.10
    Uninstalling openpyxl-3.0.10:
      Successfully uninstalled openpyxl-3.0.10
Successfully installed openpyxl-3.1.0
```

```
#Load the dataset
from google.colab import files
file = files.upload()  #upload file into google colab session
```

```
df = pd.read_csv("mismanaged-waste-global-total.csv")
df.head()
```

Choose Files mismanage…bal-total.csv

- **mismanaged-waste-global-total.csv**(text/csv) - 4916 bytes, last modified: 2/9/2023 - 100% done

```
Saving mismanaged-waste-global-total.csv to mismanaged-waste-global-total.csv
```

	Entity	Code	Year	Mismanaged waste (% global total) (% of global total)
0	Albania	ALB	2010	0.0933
1	Algeria	DZA	2010	1.6347
2	Angola	AGO	2010	0.1964
3	Anguilla	AIA	2010	0.0002
4	Antigua and Barbuda	ATG	2010	0.0039

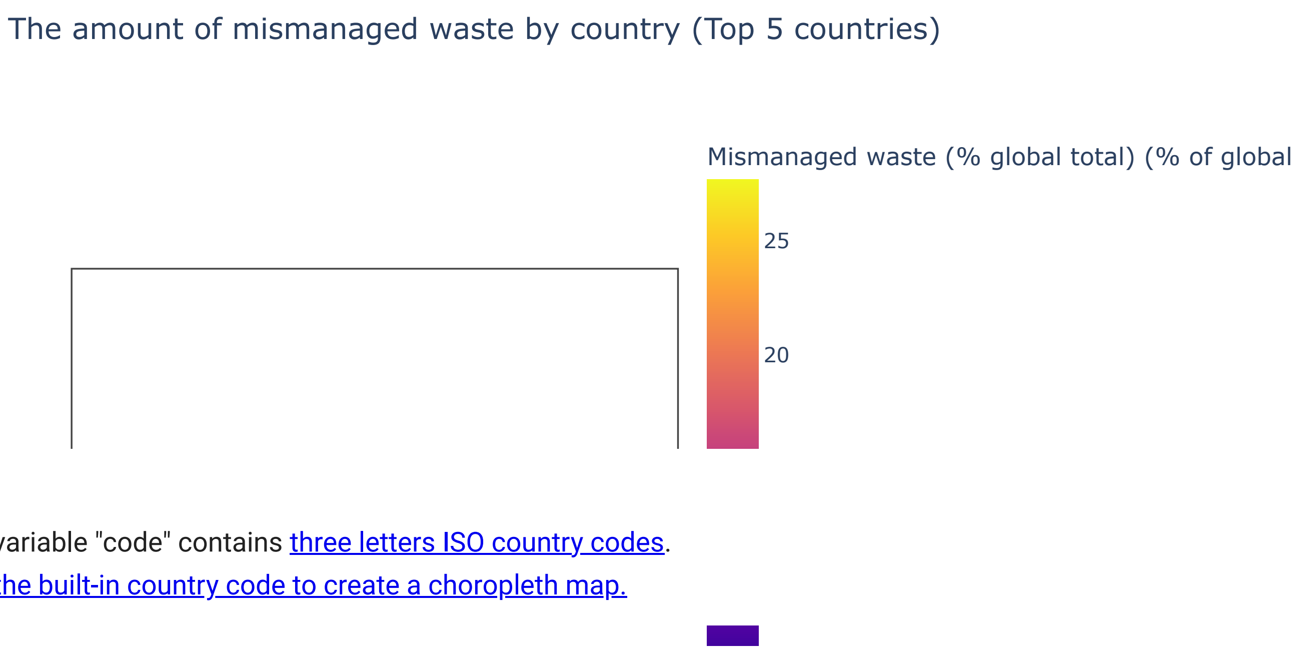
TODO1:

- Use a choropleth map to present the amount of mismanaged waste by country *(Highlight only the top 5 countries)*
- The label (hover) should include the country name and percentage of mismanaged waste
- Interpret your key findings from the map graph
- Considering the manufacturing volume of each country, is this graph misleading?

```
#highlighting top 5 countries with large amount of mismanaged waste using a choropleth map
df_top_5=df.sort_values('Mismanaged waste (% global total) (% of global total)', ascending=False).head(5)   #
df_top_5
```

	Entity	Code	Year	Mismanaged waste (% global total) (% of global total)
28	China	CHN	2010	27.6966
80	Indonesia	IDN	2010	10.1019
134	Philippines	PHL	2010	5.9153
184	Vietnam	VNM	2010	5.7588
161	Sri Lanka	LKA	2010	4.9968

```
#plotting the choropleth map of top countries with most mismanaged waste
fig = px.choropleth(df_top_5.head(5), locations='Code',color='Mismanaged waste (% global total) (% of global
fig.show()
```



Hint:

- The variable "code" contains three letters ISO country codes.
- Use the built-in country code to create a choropleth map.

Problem 2

Dataset: Plastic disposal dataset

Information: Plastic disposal dataset methods shows how has global plastic waste disposal method changed over time. In the chart we see the share of global plastic waste that is discarded, recycled or incinerated from 1980 through to 2015.

```
#Load the dataset
from google.colab import files
file = files.upload()  #upload file into google colab session
df = pd.read_excel("/content/activity.xlsx")
df.head()
```

Choose Files activity.xlsx

- **activity.xlsx**(application/vnd.openxmlformats-officedocument.spreadsheetml.sheet) - 10161 bytes, last modified: 2/9/2023 - 1

```
Saving activity.xlsx to activity.xlsx
```

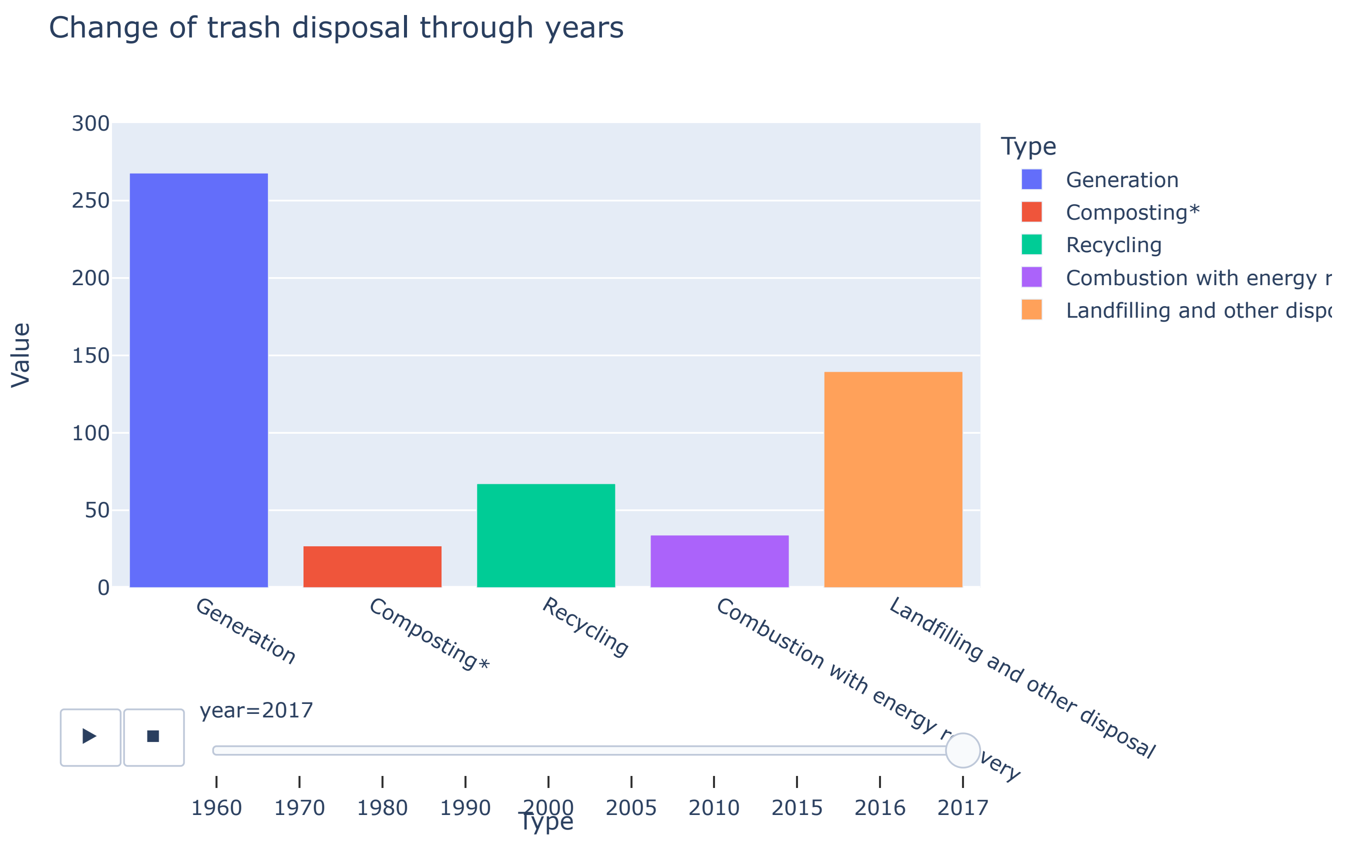
	year	Value	Type
0	1960	88.1	Generation
1	1960	NaN	Composting*
2	1960	5.6	Recycling
3	1960	0.0	Combustion with energy recovery
4	1960	82.5	Landfilling and other disposal

TODO1:

- Use an animated bar chart to indicate the change of trash disposal method through the years
- Interpret your key findings from the graph

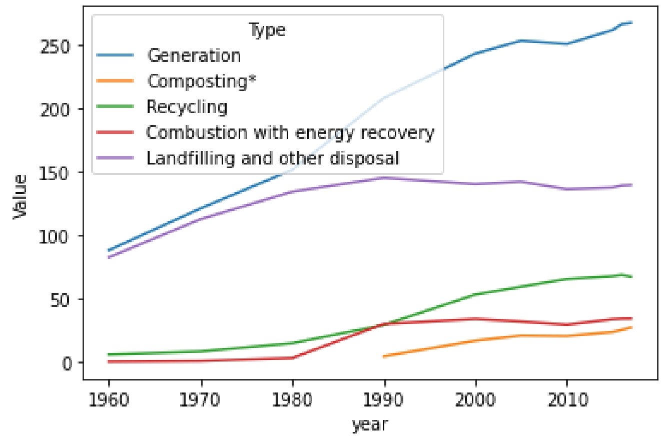
```
#using an animated bar chart to display change of trash disposal method through years
```

```
#using an animated bar chart to display change of trash disposal method through years
from matplotlib.animation import FuncAnimation          #importing FuncAnimation fom matplotlib.animation
fig = px.bar(df, x='Type', y="Value", color="Type",
  animation_frame="year", animation_group="Type",range_y=[0,300],     #selecting the animation frame, animati
             title='Change of trash disposal through years')
fig.show()
```



```
#TODO2: Alternate Visualization for animated bar we can also visualize line graph over series of time
sns.lineplot(data=df,x='year',y='Value',hue='Type')
# we can observe the trend among Type over the year(1960-2010) using line plot
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2a9b961640>
```



Hint: [Animated Bar Charts with Plotly Express]

TODO2:

- Suggest and show a better way to visualize the data (choose the most approporiate visualization for this use case)

Problem 3

Dataset: Global Fortune 500

Introduction: Fortune Global 500 List is a list of largest corporations worldwide which are measured by their total fiscal year revenues. Companies rankings sorted by total revenues for their respective fiscal years ended on or before March 31 of the relevant year.

```
#Upgrade the package "plotly" before you start to avoid future syntax error
#You only need to upgrade it once
!pip install plotly --upgrade
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: plotly in /usr/local/lib/python3.8/dist-packages (5.5.0)
Collecting plotly
  Downloading plotly-5.13.0-py2.py3-none-any.whl (15.2 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 15.2/15.2 MB 35.3 MB/s eta 0:00:00
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.8/dist-packages (from plotly)
Installing collected packages: plotly
  Attempting uninstall: plotly
    Found existing installation: plotly 5.5.0
    Uninstalling plotly-5.5.0:
      Successfully uninstalled plotly-5.5.0
Successfully installed plotly-5.13.0
WARNING: The following packages were previously imported in this runtime:
  [_plotly_utils,plotly]
You must restart the runtime in order to use newly installed versions.
```

RESTART RUNTIME

```
#Import packages
import pandas as pd
import numpy as np
import plotly.express as px

#Load the dataset
from google.colab import files
file = files.upload()  #upload file into google colab session
df = pd.read_csv("Global Fortune 500.csv")
df.head()
```

Choose Files Global Fortune 500.csv

- **Global Fortune 500.csv**(text/csv) - 37106 bytes, last modified: 2/7/2023 - 100% done

```
Saving Global Fortune 500.csv to Global Fortune 500.csv
```

	Rank	Company Name	Country	Number of Employees	Previous Rank	Revenues($millions)	Revenue Change	Profits($millions)	Prof Char
0	1	Walmart	USA	2,300,000	1	485873	0.80%	13643	-7.2
1	2	State Grid	China	926,067	2	315199	-4.40%	9571.3	-6.2
2	3	Sinopec Group	China	713,288	4	267518	-9.10%	1257.9	-65.0

TODO1:

- Build a treemap of the companies with "country" as the first hierarchy and "company" as the second hierarchy (Only show top 5 businesses for each country)
- The size of each block should indicate the corresponding company's revenue
- Interpret your key findings from the treemap

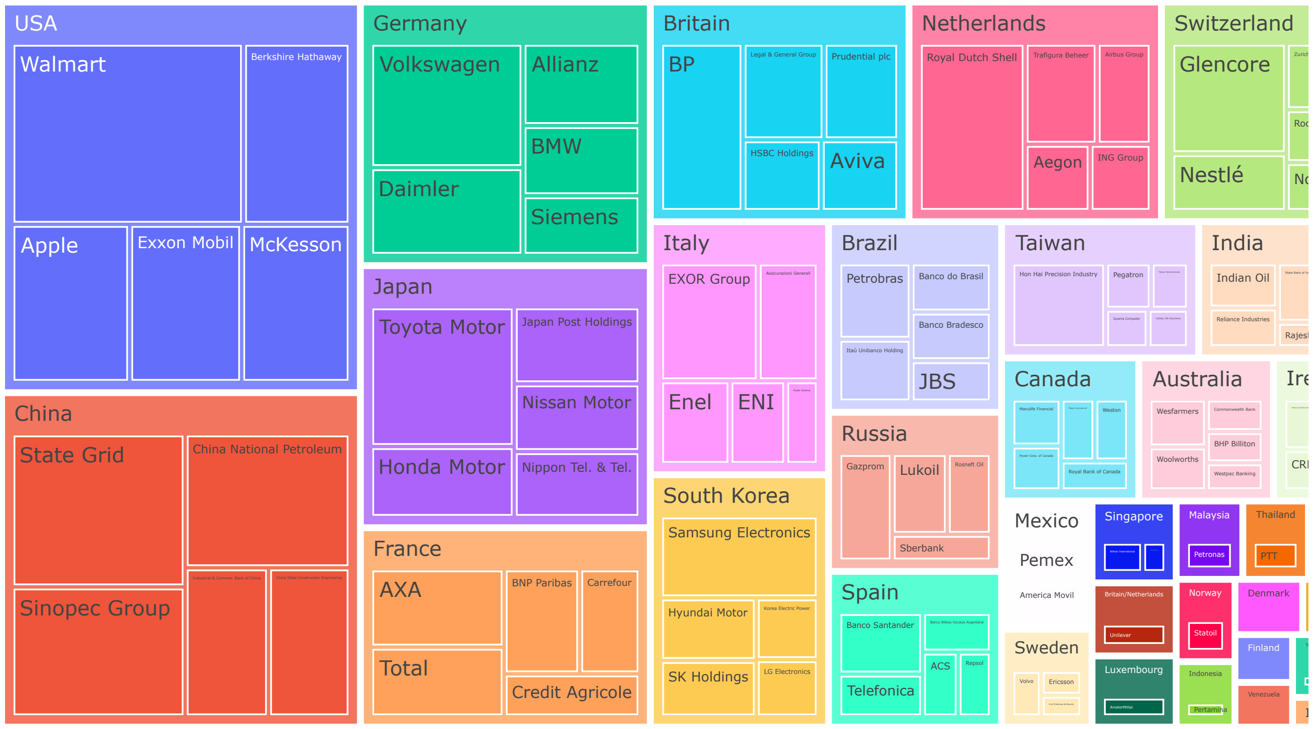
```
#grouping by countries and companies and showing only top 5 businesses for each country
df_revenue = df.groupby("Country").apply(lambda x: x.nlargest(5, "Revenues($millions)"))
df_revenue
```

		Rank	Company Name	Country	Number of Employees	Previous Rank	Revenues($millions)	Revenue Change	Profit
Country									
Australia	**197**	198	Wesfarmers	Australia	220,000	171	48003	-7.60%	
	217	218	Woolworths	Australia	205,000	176	43925	-13.20%	
	332	333	Commonwealth Bank	Australia	45,129	269	32287	-14.30%	
	349	350	BHP Billiton	Australia	26,827	168	30912	-40.90%	
	390	391	Westpac Banking	Australia	35,280	336	27704	-10.80%	
...	**...**	...	...	...	...	...	...	...	
USA	**7**	8	Berkshire Hathaway	USA	367,700	11	223604	6.10%	
	8	9	Apple	USA	116,000	9	215639	-7.70%	
	9	10	Exxon Mobil	USA	72,700	6	205004	-16.70%	
	10	11	McKesson	USA	64,500	12	198533	3.10%	
Venezuela	**441**	442	Mercantil Servicios Financieros	Venezuela	8,370	-	24403	50.30%	

109 rows × 12 columns

```
#plotting a treemap by mentioning path and values, mentioning margins
```

```
fig = px.treemap(df_revenue, path=["Country","Company Name"],
                 values='Revenues($millions)')
fig.update_layout(margin = dict(t=50, l=25, r=25, b=25))
fig.show()
```



Hint: Build a treemap with Plotly

Problem 4

Dataset: Air Quality

Introduction: The dataset contains 9358 instances of hourly averaged responses from an array of 5 metal oxide chemical sensors embedded in an Air Quality Chemical Multisensor Device. The device was located on the field in a significantly polluted area, at road level,within an Italian city. Data were recorded from March 2004 to February 2005 (one year)representing the longest freely available recordings of on field deployed air quality chemical sensor devices responses. Ground Truth hourly averaged concentrations for CO, Non Metanic Hydrocarbons, Benzene, Total Nitrogen Oxides (NOx) and Nitrogen Dioxide (NO2) and were provided by a co-located reference certified analyzer. Evidences of cross-sensitivities as well as both concept and sensor drifts are present as described in De Vito et al., Sens. And Act. B, Vol. 129,2,2008 (citation required) eventually affecting sensors concentration estimation capabilities. Missing values are tagged with -200 value.

```
#Import required libraries
import scipy.stats as stats
from sklearn import preprocessing
```

```
%matplotlib inline

#Load the dataset
from google.colab import files
file = files.upload()  #upload file into google colab session
df = pd.read_excel("Air Quality.xlsx")
df.head()
```

Choose Files Air Quality.xlsx

- **Air Quality.xlsx**(application/vnd.openxmlformats-officedocument.spreadsheetml.sheet) - 1183115 bytes, last modified: 2/9/2C

```
Saving Air Quality.xlsx to Air Quality.xlsx
```

	Date	Time	CO(GT)	PT08.S1(CO)	NMHC(GT)	C6H6(GT)	PT08.S2(NMHC)	NOx(GT)	PT08.S3(NOx)	NO2(G1
0	2004-03-10	18:00:00	2.6	1360.00	150	11.881723	1045.50	166.0	1056.25	113
1	2004-03-10	19:00:00	2.0	1292.25	112	9.397165	954.75	103.0	1173.75	92
2	2004-03-10	20:00:00	2.2	1402.00	88	8.997817	939.25	131.0	1140.00	114
3	2004-03-10	21:00:00	2.2	1375.50	80	9.228796	948.25	172.0	1092.00	122
4	2004-03-10	22:00:00	1.6	1272.25	51	6.518224	835.50	131.0	1205.00	116

TODO1:

- Plot a correlation heatmap for the Air Quality dataset
- Interpret your key findings from the correlation heatmap (value of correlation should be displayed in the heatmap)

```
#plotting a heatmap for the air quality dataset
import seaborn as sns
corr=df.corr()
sns.heatmap(data=corr,annot=True)
sns.set(rc={'figure.figsize':(15, 10)})
#key findings: finding the corelation between all the variables in the air quality dataset
```

Hint: Build a heatmap with Seaborn

Part 2: Dimension Reduction

Problem 5

please consider the **iris dataset**:

```
#Import the built-in dataset (Wine recognition) for this problem
import sklearn
from sklearn import datasets
from sklearn.decomposition import PCA
from sklearn.datasets import load_iris
print(sklearn.datasets.load_iris().DESCR)
```

```
     **Data Set Characteristics:**

         :Number of Instances: 150 (50 in each of three classes)
         :Number of Attributes: 4 numeric, predictive attributes and the class
         :Attribute Information:
             - sepal length in cm
             - sepal width in cm
             - petal length in cm
             - petal width in cm
             - class:
                     - Iris-Setosa
                     - Iris-Versicolour
                     - Iris-Virginica

         :Summary Statistics:

         ============== ==== ==== ======= ===== ====================
                         Min  Max   Mean    SD   Class Correlation
         ============== ==== ==== ======= ===== ====================
         sepal length:   4.3  7.9   5.84   0.83    0.7826
         sepal width:    2.0  4.4   3.05   0.43   -0.4194
         petal length:   1.0  6.9   3.76   1.76    0.9490  (high!)
         petal width:    0.1  2.5   1.20   0.76    0.9565  (high!)
         ============== ==== ==== ======= ===== ====================

         :Missing Attribute Values: None
         :Class Distribution: 33.3% for each of 3 classes.
         :Creator: R.A. Fisher
         :Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
         :Date: July, 1988

     The famous Iris database, first used by Sir R.A. Fisher. The dataset is taken
     from Fisher's paper. Note that it's the same as in R, but not as in the UCI
     Machine Learning Repository, which has two wrong data points.
```

```
    latter are NOT linearly separable from each other.

    .. topic:: References

       - Fisher, R.A. "The use of multiple measurements in taxonomic problems"
         Annual Eugenics, 7, Part II, 179-188 (1936); also in "Contributions to
         Mathematical Statistics" (John Wiley, NY, 1950).
       - Duda, R.O., & Hart, P.E. (1973) Pattern Classification and Scene Analysis.
         (Q327.D83) John Wiley & Sons.  ISBN 0-471-22361-1.  See page 218.
       - Dasarathy, B.V. (1980) "Nosing Around the Neighborhood: A New System
         Structure and Classification Rule for Recognition in Partially Exposed
         Environments".  IEEE Transactions on Pattern Analysis and Machine
         Intelligence, Vol. PAMI-2, No. 1, 67-71.
       - Gates, G.W. (1972) "The Reduced Nearest Neighbor Rule".  IEEE Transactions
         on Information Theory, May 1972, 431-433.
       - See also: 1988 MLC Proceedings, 54-64.  Cheeseman et al"s AUTOCLASS II
         conceptual clustering system finds 3 classes in the data.
       - Many, many more ...
```

```
#laod the dataset
import pandas as pd
import numpy as np
iris = datasets.load_iris()
X = pd.DataFrame(iris.data)
X.head()
```

	0	1	2	3
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

```
# laod the vector of target variable
y = iris.target
y
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

```
#to find the variance of the data points present in the dataset X
p = pd.DataFrame()
p[0]=X[0].apply(lambda x:x*x)
p[0].sum()/(X.shape[0]-1)
```

```
35.059395973154366
```

TODO1:

- Determine first two principal component scores for the data set with eigenvalues and eigenvectors on the RAW data (without standardization).

- Note that you are expected to perform matrix multipication, eigen value calculation **only** with the package Numpy, and sorted eigenvalues and eigenvectors in descending order (i.e. $\lambda_0 \geq \lambda_1 \geq \lambda 2 \geq \ldots \geq \lambda_n$ so do the eigenvectors).

```
# Perform matrix multiplication of raw data and the eigen vector matrix
from numpy import linalg as LA
import numpy as np
import pandas as pd


#converting the dataset X into array to easily compute the covariance matrix
data=np.array(X)
np.shape(data)
```

```
     (150, 4)
```

```
#Finding the covariance matrix
cov_matrix=np.cov(data,rowvar=False)
cov_matrix
```

```
     array([[ 0.68569351, -0.042434  ,  1.27431544,  0.51627069],
            [-0.042434  ,  0.18997942, -0.32965638, -0.12163937],
            [ 1.27431544, -0.32965638,  3.11627785,  1.2956094 ],
            [ 0.51627069, -0.12163937,  1.2956094 ,  0.58100626]])
```

```
#finding the eigen values and eigen vectors
w, v = LA.eig(cov_matrix)
w,v#w is eigen values and v is eigen vector
```

```
     (array([4.22824171, 0.24267075, 0.0782095 , 0.02383509]),
      array([[ 0.36138659, -0.65658877, -0.58202985,  0.31548719],
             [-0.08452251, -0.73016143,  0.59791083, -0.3197231 ],
             [ 0.85667061,  0.17337266,  0.07623608, -0.47983899],
             [ 0.3582892 ,  0.07548102,  0.54583143,  0.75365743]]))
```

```
w#eigen values-lambda
```

```
     array([4.22824171, 0.24267075, 0.0782095 , 0.02383509])
```

```
# Sort the eigenvalues in descending order
idx = np.argsort(v)[::-1]
eigenvalues = w[idx]
eigenvectors = v[:,idx]


w
#print(eigenvalues)
```

```
     array([4.22824171, 0.24267075, 0.0782095 , 0.02383509])
```

```
#print eigenvectors
v
```

```
     array([[ 0.36138659, -0.65658877, -0.58202985,  0.31548719],
            [-0.08452251, -0.73016143,  0.59791083, -0.3197231 ],
            [ 0.85667061,  0.17337266,  0.07623608, -0.47983899],
            [ 0.3582892 ,  0.07548102,  0.54583143,  0.75365743]])
```

```
# Select the top k eigenvectors
k = 2
eigenvectors = v[:, :k]
eigenvectors
```

```
     array([[ 0.36138659, -0.65658877],
            [-0.08452251, -0.73016143],
            [ 0.85667061,  0.17337266],
            [ 0.3582892 ,  0.07548102]])
```

```
#calculating the variance ratio
variance_ratio = w / np.sum(w)
variance_ratio
```

```
     array([0.92461872, 0.05306648, 0.01710261, 0.00521218])
```

```
result=data @ eigenvectors
#result
```

Hint:

1. [Eigen value calculation with Numpy]()
2. [Sorting a Numpy array]()

TODO2:

- Perform the data standardization on the data without using any buit-in functions.
- Determine the first two principal component scores with eigenvaues and eigenvectors on the standardized data by updataing your code form **TODO1**. (i.e. your code from **TODO** one should not be changed; you should only add a cell of code for standardization.)

```
#standardizing the dataset by finding the mean and standard deviation for each and every column and subtracti
new_s_data=(X-X.mean())/X.std()
new_s_data
```

	0	1	2	3
0	-0.897674	1.015602	-1.335752	-1.311052
1	-1.139200	-0.131539	-1.335752	-1.311052

```
#Finding the covariance matrix
cov_matrix=np.cov(new_s_data,rowvar=False)
cov_matrix

     array([[ 1.        , -0.11756978,  0.87175378,  0.81794113],
            [-0.11756978,  1.        , -0.4284401 , -0.36612593],
            [ 0.87175378, -0.4284401 ,  1.        ,  0.96286543],
            [ 0.81794113, -0.36612593,  0.96286543,  1.        ]])
```

```
w, v = LA.eig(cov_matrix)
w,v#w is eigen values and v is eigen vector

     (array([2.91849782, 0.91403047, 0.14675688, 0.02071484]),
      array([[ 0.52106591, -0.37741762, -0.71956635,  0.26128628],
             [-0.26934744, -0.92329566,  0.24438178, -0.12350962],
             [ 0.5804131 , -0.02449161,  0.14212637, -0.80144925],
             [ 0.56485654, -0.06694199,  0.63427274,  0.52359713]]))
```

```
w#eigen values

     array([2.91849782, 0.91403047, 0.14675688, 0.02071484])
```

```
v#eigen vectors

     array([[ 0.52106591, -0.37741762, -0.71956635,  0.26128628],
            [-0.26934744, -0.92329566,  0.24438178, -0.12350962],
            [ 0.5804131 , -0.02449161,  0.14212637, -0.80144925],
            [ 0.56485654, -0.06694199,  0.63427274,  0.52359713]])
```

```
#sort eigen values and eigen vectors
idx = np.argsort(v)[::-1]
eigenvalues = w[idx]
eigenvectors = v[:,idx]
```

```
#selecting first two components
k = 2
eigenvectors = v[:, :k]
eigenvectors

     array([[ 0.52106591, -0.37741762],
            [-0.26934744, -0.92329566],
            [ 0.5804131 , -0.02449161],
            [ 0.56485654, -0.06694199]])
```

```
#computing the pca matrix by mutiplying dataset and eigenvectors
result=new_s_data @ eigenvectors
result
```

	0	1
0	-2.257141	-0.478424
1	-2.074013	0.671883
2	-2.356335	0.340766
3	-2.291707	0.595400
4	-2.381863	-0.644676
...	...	...
145	1.864258	-0.385674
146	1.559356	0.893693
147	1.516091	-0.268171

TODO3:

- Use the built-in function called `PCA()` on the raw data to calculate the first two principal components.
- For each of the components, indentify the explained variance, proportion variance, and cummulative proportion of variance.
- Compare your result with **TODO1**, are they same? (You might expect that you will have the same result with **TODO1**. However, `PCA()` function automatically does `x-mean()` transformation. Therefore, do not worry that your result from this one is different than the result performed from **TODO1**.)

Hint:

- PCA Python

```
import numpy as np
from sklearn.decomposition import PCA
x=np.array(data)
pca=PCA(n_components=2)
pca.fit(x)
d=pca.transform(x)


#calculating vraiance, proportion variance and cumulative proportion of variance
explained_variance_ratio = pca.explained_variance_ratio_
variance = pca.explained_variance_
cumulative_proportion_of_variance = np.cumsum(explained_variance_ratio)
print("Variance: ", variance)
print("Proportion of Variance: ", explained_variance_ratio)
print("Cumulative Proportion of Variance: ", cumulative_proportion_of_variance)
```

```
    Variance:  [4.22824171 0.24267075]
    Proportion of Variance:  [0.92461872 0.05306648]
    Cumulative Proportion of Variance:  [0.92461872 0.97768521]
```

TODO4:

- Use the built-in function called `preprocessing.StandardScaler` and `PCA` to calculate the first two principal components.

- For each of the components, determine the explained variance, propotion variance, and cummulative proportion of variance.
- Compare your result with **TODO2**, are they same?

Hint: Data Standardization in Python

```
#using standard scalar function, scaling the data within the range -1 to 1
import numpy as np
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
data_scaled = scaler.fit_transform(data)
pca=PCA(n_components=2)           #selecting the first two components
pca.fit(data_scaled)
d2=pca.transform(data_scaled)
d2
```

```
#calculating variance, proportion of variance and cumulative proportion of variance for the scaled data
var = pca.explained_variance_
prop_var = var / np.sum(var)
cum_prop_var = pca.explained_variance_ratio_.cumsum()
print("Variance: ", var)
print("Proportion of Variance: ", prop_var)
print("Cumulative Proportion of Variance: ", cum_prop_var)
```

```
Variance:  [2.93808505 0.9201649 ]
Proportion of Variance:  [0.76150718 0.23849282]
Cumulative Proportion of Variance:  [0.72962445 0.95813207]
```

TODO5:

- Integrate from **TODO1** to **TODO4**, why is it important to standardize(normalize) variables before perfroming PCA
- Plot the records on 2D plane defined by the first two PCA components calcualted with standardized data and dffierentiate them using by **target** (i.e. y).
- Make some meaningful interpretation about the plot.

```
#To find the difference between standardised data and non standardised data, thus plottig the records on 2d c
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
original_data = result.copy()
scaler = StandardScaler()
standardized_data = scaler.fit_transform(data_scaled)
pca = PCA(n_components=2)
pca_results = pca.fit_transform(standardized_data)
#pca_results
```

```
#appending target variable Y to the dataframe
pca_df=pd.DataFrame(pca_results,columns=['z1','z2'])
#pca_df
```

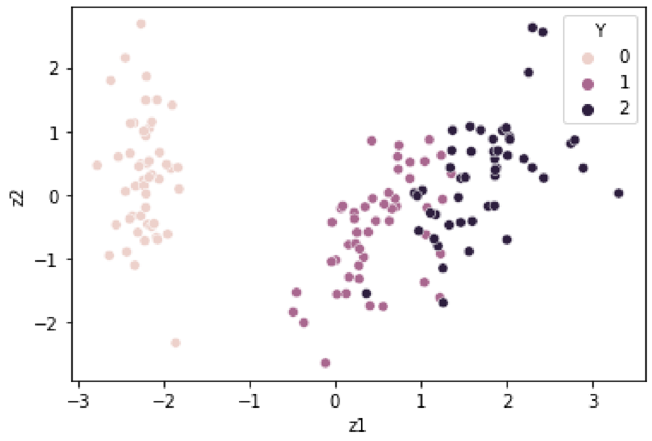
```
pca_df['Y']=y
pca_df
```

	z1	z2	Y
0	-2.264703	0.480027	0
1	-2.080961	-0.674134	0
2	-2.364229	-0.341908	0
3	-2.299384	-0.597395	0
4	-2.389842	0.646835	0
...	...	...	...
145	1.870503	0.386966	2
146	1.564580	-0.896687	2
147	1.521170	0.269069	2
148	1.372788	1.011254	2
149	0.960656	-0.024332	2

150 rows × 3 columns

```
#plotting a scatter plot between the component variables z1 and z2, hueing it with the target variable
sns.scatterplot(data=pca_df,x='z1',y='z2',hue='Y')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2a9aba3460>
```



Interpretation: From the above scatter plot, we can say that the principal components identify the most important underlying patterns in the data defined by the classes i.e 0, 1, 2. We can also observe that the datapoints lie in the range of (-3,3) as the data is standardized.

Problem 6

Dataset: Life Expectancy

Introduction: The above dataset gives life expectancy related data for 37 countries in2014.

Consider only the following variables in your analysis: 'GDP', 'Income composition of resources', 'Schooling', and 'Total expenditure'.

```
#Import useful package
from sklearn.manifold import MDS

#Load the dataset
from google.colab import files
file = files.upload()  #upload file into google colab session
df = pd.read_csv("Life Expectancy.csv")
df.head()
```

Choose Files Life Expectancy.csv

- **Life Expectancy.csv**(text/csv) - 4703 bytes, last modified: 2/9/2023 - 100% done

```
Saving Life Expectancy.csv to Life Expectancy.csv
```

	Country	Year	Status	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B	Measles
0	Afghanistan	2014	Developing	59.9	271	64	0.01	73.523582	62.0	492
1	Australia	2014	Developed	82.7	6	1	9.71	10769.363050	91.0	340
2	Austria	2014	Developed	81.4	66	0	12.32	8350.193523	98.0	117
3	Bangladesh	2014	Developing	71.4	132	98	0.01	10.446403	97.0	289
4	Belgium	2014	Developed	89.0	76	0	12.60	7163.348923	98.0	70

5 rows × 22 columns

TODO1:

- Standardize the numeric variables in the given data frame
- Run MDS() (Multi Dimensional Scaling) on the standardized data
    - **Hint:** n_components = 2
- Plot data points on a 2D plane defined by the first two components
- Use color to differentiate the statues of each country with legend
- Use text label to specify the country name for each point
- Comment your findings from the graph

Hint: MDS Python

```
#selecting only the required columns into a dataframe
new_df=df[['GDP', 'Income composition of resources', 'Schooling','Total expenditure']]
new_df
```

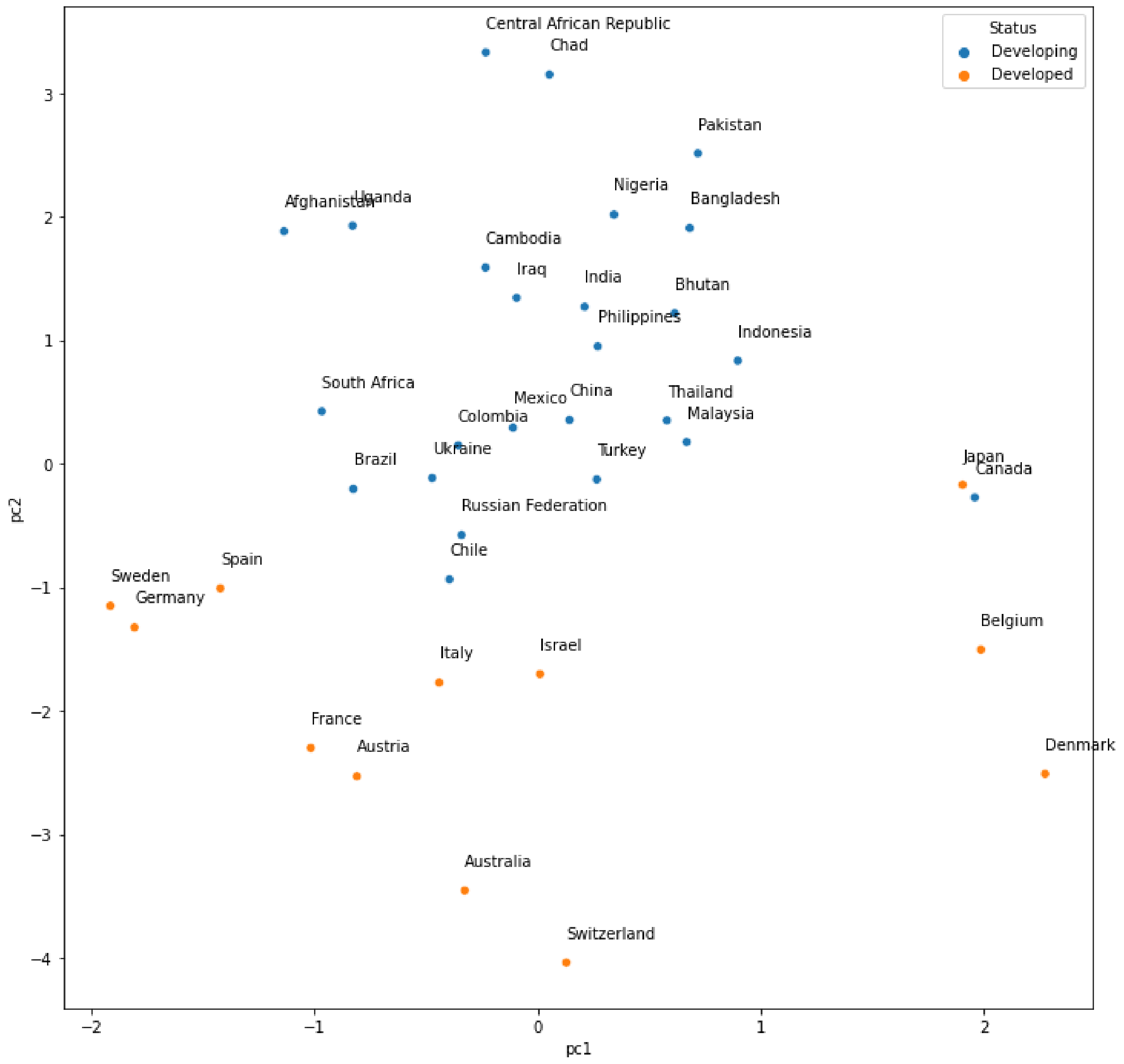
	GDP	Income composition of resources	Schooling	Total expenditure
0	612.696514	0.476	10.0	8.18
1	62214.691200	0.936	20.4	9.42
2	51322.639970	0.892	15.9	11.21
3	184.565430	0.570	10.0	2.82
4	47439.396840	0.890	16.3	1.59
5	2522.796800	0.596	12.5	3.57
6	1226.617310	0.747	15.2	8.32
7	198.687123	0.553	10.9	5.68
8	544.433760	0.912	15.9	1.45
9	377.132274	0.345	7.1	4.20
10	125.998515	0.390	7.3	3.62
11	14817.377780	0.841	16.2	7.79
12	7683.523800	0.723	13.1	5.55
13	7913.383432	0.720	13.6	7.20
14	62425.539200	0.926	19.2	1.80
15	42955.242870	0.890	16.2	11.54
16	4792.652880	0.920	17.0	11.30
17	1573.118890	0.607	11.6	4.69
18	3491.595887	0.682	12.9	2.85
19	673.747370	0.658	10.1	5.54
20	37582.846240	0.895	16.0	7.81
21	35396.665170	0.877	16.3	9.25
22	3896.211510	0.899	15.3	1.23
23	11183.961910	0.783	13.0	4.17
24	1452.277660	0.754	13.1	6.30
25	3221.678128	0.521	10.0	3.67
26	1316.989660	0.542	7.8	2.61
27	2842.938353	0.676	11.7	4.71
28	14125.961000	0.803	14.9	7.70
29	6479.625659	0.660	13.0	8.80
30	296.472250	0.877	17.6	9.30
31	5918.198980	0.906	15.8	11.93
32	85814.588570	0.936	15.9	11.66
33	5941.847100	0.737	13.6	4.12

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.manifold import MDS
# extract numeric columns only leaving character variables
df_numeric = new_df.select_dtypes(include=[np.number])
# standardize the numeric columns
df_standardized = StandardScaler().fit_transform(df_numeric)
# run MDS with 2 components
mds = MDS(n_components=2)
df_mds = mds.fit_transform(df_standardized)
df_mds
```

```
     array([[-1.13661336,  1.88286165],
            [-0.32619919, -3.45460636],
            [-0.80989262, -2.53069007],
            [ 0.68271314,  1.90867323],
            [ 1.98845178, -1.50478908],
            [ 0.61523451,  1.2204842 ],
            [-0.82597681, -0.20356233],
            [-0.23302263,  1.58934663],
            [ 1.96119818, -0.27195505],
            [-0.23183859,  3.33161017],
            [ 0.05324368,  3.1517462 ],
            [-0.39464004, -0.93531171],
            [ 0.14385298,  0.35465831],
            [-0.35537526,  0.14976824],
            [ 2.27617541, -2.51044799],
            [-1.01640652, -2.30038543],
            [-1.80700516, -1.3245783 ],
            [ 0.21166848,  1.27204526],
            [ 0.89843885,  0.83515649],
            [-0.09374905,  1.34422735],
            [ 0.01119949, -1.70267589],
            [-0.44056139, -1.77160405],
            [ 1.90678749, -0.16997066],
            [ 0.66877393,  0.17660246],
            [-0.10957935,  0.29228618],
            [ 0.34338586,  2.01751227],
            [ 0.71940382,  2.51365072],
            [ 0.27037087,  0.95096644],
            [-0.34047238, -0.57708918],
            [-0.96712577,  0.425222  ],
            [-1.42217101, -1.00895419],
            [-1.91504428, -1.15041502],
            [ 0.129562  , -4.038383  ],
            [ 0.58034251,  0.35158471],
            [ 0.26606686, -0.1261997 ],
            [-0.82906362,  1.92783389],
            [-0.47213281, -0.11461836]])
```

```
#creating a new dataframe with columns pc1 and pc2
new=pd.DataFrame(df_mds,columns=['pc1','pc2'])
```

```
#plotting a scatterplot to disply countries based on developed and developing statuses
import seaborn as sns
import matplotlib.pyplot as plt
plt.rcParams["figure.figsize"] = (12,12)
sns.scatterplot(x = "pc1", y = "pc2", hue = df["Status"], data = new)
for i, txt in enumerate(df['Country']):
  plt.annotate(txt, (new['pc1'][i], new['pc2'][i]+0.20), fontsize=10)
```



Problem 7

Dataset: Game of thrones Books

Introduction: If you haven't heard of Game of Thrones, then you must be really good at hiding. Game of Thrones is the hugely popular television series by HBO based on the (also) hugely popular book series A Song of Ice and Fire by George R.R. Martin. You need to analyze the co-occurrence network of the characters in the Game of Thrones books. Here, two characters are considered to co-occur if their names appear in the vicinity of 15 words from one another in the books.

This dataset (5 files attached in zip file) constitutes a network and is given as a text file describing the edges between characters, with some attributes attached to each edge.

Loading the required libraries

```
!pip install pyvis
import pyvis
import networkx as nx
from pyvis.network import Network
!pip install decorator==5.0.9
!pip install --user networkx==2.3
```

```
    Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
    Collecting pyvis
      Downloading pyvis-0.3.1.tar.gz (748 kB)
         ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 748.9/748.9 KB 10.8 MB/s eta 0:00:00
```

TODO1:

- Load data for the all the books of Game of Thrones (Merge all files into one and create a dataframe)
- Select only the rows which has weight more than 10

```
         ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 40.5/40.5 KB 4.1 MB/s eta 0:00:00
```

```
#uploading all the data files
import pandas as pd
from google.colab import files
file = files.upload()
```

Choose Files 5 files

- **book5.csv**(text/csv) - 31515 bytes, last modified: 2/9/2023 - 100% done
- **book4.csv**(text/csv) - 28973 bytes, last modified: 2/9/2023 - 100% done
- **book3.csv**(text/csv) - 41174 bytes, last modified: 2/9/2023 - 100% done
- **book2.csv**(text/csv) - 31768 bytes, last modified: 2/9/2023 - 100% done
- **book1.csv**(text/csv) - 28344 bytes, last modified: 2/9/2023 - 100% done

```
Saving book5.csv to book5.csv
Saving book4.csv to book4.csv
Saving book3.csv to book3.csv
Saving book2.csv to book2.csv
Saving book1.csv to book1.csv
  Building wheel for pyvis (setup.py) ... done
```

```
file1=pd.read_csv('/content/book1.csv')
file2=pd.read_csv('/content/book2.csv')
file3=pd.read_csv('/content/book3.csv')
file4=pd.read_csv('/content/book4.csv')
file5=pd.read_csv('/content/book5.csv')
files = ['/content/book1.csv','/content/book2.csv','/content/book3.csv','/content/book4.csv','/content/book5.
dataframe = [pd.read_csv(file) for file in files]
merged_df = pd.concat(dataframe, axis=0, ignore_index=True)
```

```
      Found existing installation: decorator 4.4.2
```

```
#merging all the 5 datasets into one
merged_df
```

```
Source          Target     Type  weight  book
```

```
df_filtered = merged_df[merged_df['weight'] > 10]
df_filtered
#selecting the rows whose weight is greater than 10
```

	Source	Target	Type	weight	book
8	Aemon-Targaryen-(Maester-Aemon)	Jeor-Mormont	Undirected	13	1.0
9	Aemon-Targaryen-(Maester-Aemon)	Jon-Snow	Undirected	34	1.0
16	Aerys-II-Targaryen	Robert-Baratheon	Undirected	12	1.0
17	Aggo	Daenerys-Targaryen	Undirected	11	1.0
30	Alliser-Thorne	Jon-Snow	Undirected	32	1.0
...	...	...	...	...	...
3900	Tyrion-Lannister	Tywin-Lannister	undirected	18	5.0
3902	Tyrion-Lannister	Yandry	undirected	15	5.0
3903	Tyrion-Lannister	Yezzan-zo-Qaggaz	undirected	17	5.0
3904	Tyrion-Lannister	Ysilla	undirected	11	5.0
3908	Yandry	Ysilla	undirected	14	5.0

747 rows × 5 columns

TODO2: Load the dataframe as networkx graph

Hint: Network analysis in python

```
#importing the network package
import networkx as nx
G = nx.Graph()
```

```
#network graph for data whose weights are greater than 10
import pandas as pd
import networkx as nx
mg = nx.from_pandas_edgelist(df_filtered, source='Source', target='Target')
plt.figure(figsize=(70,20))
nx.draw(mg,node_size=30,font_size=12,font_weight='normal',with_labels=True)
```

TODO3:

Create viz network Hint: Use Pyvis

```
#plotting a pyvis network graph with the source and target
import time
from pyvis.network import Network
import networkx as nx
net = Network(notebook=False)
g=Network(height=800,width=800)
g.toggle_hide_edges_on_drag(False)
g.barnes_hut()
#Plotting the network graph
for index, row in df_filtered.iterrows():
  net.add_node(row['Source'], label=row['Source'])
  net.add_node(row['Target'], label=row['Target'])
  net.add_edge(row['Source'], row['Target'], value=row['weight'], title=row['weight'])
net.show("ex2.html")
```

```
!jupyter nbconvert --to html Group_25_Homework2.ipynb
```

```
[NbConvertApp] Converting notebook Group_25_Homework2.ipynb to html
/usr/local/lib/python3.8/dist-packages/nbconvert/filters/datatypefilter.py:39: UserWarning: Your elemen
  warn("Your element with mimetype(s) {mimetypes}"
[NbConvertApp] Writing 1576852 bytes to Group_25_Homework2.html
```

✓ 2s completed at 23:33