ASSIGNMENT - 2

Name : Nazrin Thanikattil Rafeeque

Student ID : 21031314

In [ ]:
```
import pandas as pd
import numpy as np
import wbgapi as wb
import matplotlib.pyplot as plt
import scipy.stats as stats
%matplotlib inline
import seaborn as sns
```

Defining Functions

In [ ]:
```
"""Defines a function which takes a filename as argument, reads a dataframe in World bank format and returns
two dataframes: one with years as columns and one with countries as columns"""

def wb_dataframe(Filename,country_list):
    df1 = wb.data.DataFrame(Filename,country_codes,time=range(2010,2019),columns='economy')
    df2= wb.data.DataFrame(Filename,country_codes,time=range(2010,2019))

    return df1,df2
```

In [ ]:
```
"""Defines a function to provide Xlabel ,Y label and Titler for the Plots"""
def plot_labels(x_label,y_label,title):
    plt.xticks( rotation='vertical')
    plt.xlabel(x_label)
    plt.ylabel(y_label)
    plt.title(title,fontsize= 15, fontweight='bold')
```

In [ ]:
```
"""Function to print the Different summary statitics of dataframe Attributes"""
def Display_stats(Attributes):
    print("average:        ", np.average(Attributes))
    print("std. deviations:", np.std(Attributes))
    print("maximum :       ", np.max(Attributes))
    print("minimum :       ", np.min(Attributes))
    print("skewness:       ", stats.skew(Attributes))
    print("kurtosis:       ", stats.kurtosis(Attributes))

    return
```

In [ ]:
```
"""Defining Function for the Subplots"""


def subplot(data_1,data_2,data_3,data_4):
    plt.style.use('fivethirtyeight')
    plt.figure(figsize=(15,8), constrained_layout=True)

    plt.subplot(4,1,1)
    plt.plot(data_1)
    plot_labels('Years','values','Agricultural land (% of land area)')


    plt.subplot(4,1,2)
    plt.plot(data_2)
    plot_labels('Years','values','Forest Area')

    plt.subplot(4,1,3)
    plt.plot(data_3)
    plot_labels('Years','values','% of total territorial area')

    plt.subplot(4,1,4)
    plt.plot(data_4)
    plot_labels('Years','values','value added (% of GDP)')
```

In [ ]:
```
"""Function defining the Pie Plots Dimensions"""

def Region_wise(df1,df2,subtitle):
    fig, axes = plt.subplots(1, 2, figsize=(3,2),dpi=144)
    plt.suptitle(subtitle,size=10)
    for ax, df in zip(axes, (df1,df2)):
        ax.pie(df, labels=df.index,shadow=False,startangle=90, autopct="%1.1f%%"
                    ,textprops={'fontsize': 5})
```

In [ ]:
```
"""Prints Pearon's and Spearmans correction coefficients between the different attributes of the dataframe"""

def print_corr(df):
    print("Pearson's correlation coefficients")
    r, p = stats.spearmanr(df["Agricultural land (% of land area)"], df["Forest Area"])
    print("\n Agricultural land (% of land area) vs. Forest Area coefficient ", r, "probability", p)
    r, p = stats.spearmanr(df["Forest Area"], df["% of total territorial area"])
    print("\n Forest Area vs. % of total territorial area coefficient", r, "probability", p)
    r, p = stats.spearmanr(df["% of total territorial area"], df["value added (% of GDP)"])
    print("\n % of total territorial area vs. value added (% of GDP) coefficient", r, "probability", p)
    r, p = stats.spearmanr(df["Agricultural land (% of land area)"], df["value added (% of GDP)"])
    print("\n Agricultural land (% of land area) vs. value added (% of GDP) coefficient", r, "probability", p)
```

Defining and Creating DataFrame

In [ ]:
```
#List containing the country list,indicators from World Bank API and Country groups are assigned

country_codes  =  ['GBR','NLD','ESP','JPN','KOR','IND','LKA','USA','CAN']
indicator_list =  [' AG.LND.AGRI.ZS','AG.LND.ARBL.ZS','AG.LND.FRST.ZS','NV.AGR.TOTL.ZS','ER.PTD.TOTL.ZS']
country_groups =  ({'GBR':'Europe','NLD':'Europe',
                   'ESP':'Europe','JPN':'East Asia','KOR':'East Asia',
                   'IND':'South Asia','LKA':'South Asia',
                   'USA':'North America','CAN':'North America'})
```

In [ ]:
```
idx=pd.IndexSlice
#Defining Multiindex for Dataframe 2
df1,df2 = wb_dataframe(indicator_list,country_codes)

df2
index1_ = pd.MultiIndex.from_product([['Agricultural land (% of land area)','Forest Area','value added (% of GDP)'
                                       ,'% of total territorial area'],['GBR','NLD','ESP','JPN','KOR','IND','LKA','USA',
                                       ,names=['Attributes','Country'])
df2.index = index1_
```

In [ ]:
```
#Defining Multiindex for Dataframe 1

index_ = pd.MultiIndex.from_product([['Agricultural land (% of land area)','Forest Area',
       'value added (% of GDP)','% of total territorial area'],['YR2010','YR2011','YR2012','YR2013','YR2014','YR2015',
                                                                 'YR2016','YR2017','YR2018']],names=['Attributes','year']
df1.index=index_
```

Data Cleaning

In [ ]:
```
# Checks for the null values in both the dataframes
print(df1.isna().sum())
print(df1.info())

print(df2.isna().sum())
print(df2.info())
```

In [ ]:
```
# Filling the Nan Values with backward fill method

df2.fillna(method='bfill',inplace=True)
df1.fillna(method='bfill',inplace=True)
```

In [ ]:
```
#Checking wheather the missing data is Filled

print(df1.isna().sum())
print(df2.isna().sum())
```

In [ ]:
```
#Display both the Dataframe
print(df1,df2)
```

Data Preperation

In [ ]:
```
#Normalises the Dataframe for more accuracy in plotting to have small discerte change in values

def normalise(dataframe_new):
    '''this function takes the values and normalises all the coloum values in [0,1]'''
    dataframe = dataframe_new.copy()

    for col in range(0, len(dataframe.columns)):
        max_value = dataframe.iloc[:,col].max()
        min_value = dataframe.iloc[:,col].min()


        for row in range(0, len(dataframe.index)):
            dataframe.iloc[row,col] = (dataframe.iloc[row,col]-min_value)/(max_value-min_value)
    return dataframe
```

In [ ]:
```
# calling the Normalise Function
df_1 = normalise(df1)
df_1
```

Plotting and Analysis

Analysing and Plotting the Averge Variations in the Agricultural,Tesrrestial,Forest And Value added GDP across the world

In [ ]:
```
print()
print("Agricultural land (% of land area)")
Display_stats(df1.loc[idx['Agricultural land (% of land area)', ("YR2014","YR2015","YR2016","YR2017","YR2018")], :])

print()
print("Forest Area")
Display_stats(df1.loc[idx['Forest Area', ("YR2014","YR2015","YR2016","YR2017","YR2018")], :])


print()
print("value added (% of GDP)")
Display_stats(df1.loc[idx['value added (% of GDP)', ("YR2014","YR2015","YR2016","YR2017","YR2018")], :])


print()
print("% of total territorial area")
Display_stats(df1.loc[idx['% of total territorial area', ("YR2014","YR2015","YR2016","YR2017","YR2018")], :])
```

In [ ]:
```
"""THe average values of different constrains during the year of consideration assigning as dataframe dd"""

dd=pd.DataFrame((df2.groupby('Attributes')).mean())
ddd=dd.T
ddd.index
```

In [ ]:
```
import seaborn as sns

# Declaring the cm variable by the color palette from seaborn
cm = sns.light_palette("orange", as_cmap=True)

# Visualizing the DataFrame with set precision
ddd.style.background_gradient(cmap=cm)
```

In [ ]:
```
#Changing the index of ddd dataframe to get a Time Seies plot(DD/MM/YYYY) format
ddd.index = pd.to_datetime(ddd.index.astype(str).str[2:])
ddd.index = pd.to_datetime(ddd.index)
print(ddd)

#Plots the Dataframe
plt.figure(figsize=(15,8))
ddd.plot().legend(bbox_to_anchor= (1.02, 1));
plt.title('Attributes across the Globe')
```

Analysing and Plotting the Attributes for selected Countries(GBR,USA,IND,JPN) in Different regions of world

In [ ]:
```
#Defing the datas for the individual plots of Country BRITAN
data=df_1.loc[idx['Agricultural land (% of land area)','GBR']]
data1=df_1.loc[idx['Forest Area','GBR']]*.4
data2=df_1.loc[idx['% of total territorial area','GBR']]
data3=df_1.loc[idx['value added (% of GDP)','GBR']]
print(data3)

#Function Call for Subplotting
subplot(data,data1,data2,data3)

#Display its Statistics as well
print()
print("GBR")
pd.DataFrame(Display_stats(df1.loc[idx['Agricultural land (% of land area)', 'GBR']]))
```

In [ ]:
```
#Defing the datas for the individual plots of Country USA
idx=pd.IndexSlice
data=df_1.loc[idx['Agricultural land (% of land area)','USA']]
data1=df_1.loc[idx['Forest Area','USA']]*.4
data2=df_1.loc[idx['% of total territorial area','USA']]
data3=df_1.loc[idx['value added (% of GDP)','USA']]

#Function Call for Subplotting
subplot(data,data1,data2,data3)

#Display its Statistics as well
print()
print("USA")
pd.DataFrame(Display_stats(df1.loc[idx['Agricultural land (% of land area)', 'USA']]))
```

In [ ]:
```
idx=pd.IndexSlice
#Defing the datas for the individual plots of Country INDIA
data=df_1.loc[idx['Agricultural land (% of land area)','IND']]
data1=df_1.loc[idx['Forest Area','IND']]*.4
data2=df_1.loc[idx['% of total territorial area','IND']]
data3=df_1.loc[idx['value added (% of GDP)','IND']]

#Function Call for Subplotting
subplot(data,data1,data2,data3)

#Display its Statistics as well
print()
print("IND")
pd.DataFrame(Display_stats(df1.loc[idx['Agricultural land (% of land area)', 'IND']]))
```

In [ ]:
```
#Defing the datas for the individual plots of Country JAPAN
idx=pd.IndexSlice
data=df_1.loc[idx['Agricultural land (% of land area)','JPN']]
data1=df_1.loc[idx['Forest Area','JPN']]*.4
data2=df_1.loc[idx['% of total territorial area','JPN']]
data3=df_1.loc[idx['value added (% of GDP)','JPN']]

#Function Call for Subplotting
subplot(data,data1,data2,data3)

#Display its Statistics as well
print()
print("JPN")
pd.DataFrame(Display_stats(df1.loc[idx['Agricultural land (% of land area)', 'JPN']]))
```

In [ ]:
```
#Defining an dAnalysing the Variation in GDP for the above defined Countries

d1=df_1.loc[idx['value added (% of GDP)',['USA','GBR','KOR','JPN']]]
print(d1)


d1.index=['2010','2011','2012','2013','2014','2015','2016','2017','2018']
d1.index = pd.to_datetime(d1.index)
print(d1)

d1.plot(subplots=True, figsize=(12, 15))
```

Analysing and plotting variations in attribute Across different Regions of World(Europe,North America,South and East Asia)

In [ ]:
```
# Mapping the Existing Datframe with the Country_groups Dictionary to have a detailed Analysis.


df2['Region'] = df2.index.get_level_values(1).map(country_groups)
df2.groupby('Region')

# Defining the Slices For Plotting

agri=df2.loc[idx[['Agricultural land (% of land area)'],:]].groupby('Region').mean()
forest=df2.loc[idx[['Forest Area'],:]].groupby('Region').mean()
terrestial=df2.loc[idx[['% of total territorial area'],:]].groupby('Region').mean()

# Locating the Slices For Desired years
agri_a = agri.loc[:, "YR2015":"YR2018"].mean(axis=1)
agri_b= agri.loc[:, "YR2010":"YR2014"].mean(axis=1)

forest_a = forest.loc[:, "YR2015":"YR2018"].mean(axis=1)
forest_b= forest.loc[:, "YR2010":"YR2014"].mean(axis=1)


Terrestial_a = terrestial.loc[:, "YR2015":"YR2018"].mean(axis=1)
Terrestial_b= terrestial.loc[:, "YR2010":"YR2014"].mean(axis=1)

# Plotting with the Function Region_wise
Region_wise(agri_b,agri_a,'Agricultural land before and after 2015')
Region_wise(forest_b,forest_a,'Forest Area before and after 2015')
Region_wise(Terrestial_b,Terrestial_a,'Total territorial area before and after 2015')
```

In [ ]:
```
# Printing the Correlation between Attributes

print_corr(ddd)
```

In [ ]:
```
# plotting the Correlation using Heatmap

sns.heatmap(ddd.corr(), annot = True,cmap="YlOrRd_r")
```

END