```
#!/usr/bin/env python
# coding: utf-8

# In[3]:


import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
get_ipython().run_line_magic('matplotlib', 'inline')
# scaling
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler
# linear regression
from sklearn import linear_model


# In[4]:


Pov_data = pd.read_csv("Poverty_LifeExp.csv")
# quick view of columns and values
Pov_data.head()
# how many columns and rows in dataframe
Pov_data.shape
Pov_data.isnull().sum()
# are there duplicate values?
format(len(Pov_data[Pov_data.duplicated()]))
# standard statistical measures
Pov_data.describe(percentiles = [.25, .5, .75, .90 ,.95, .99])


# In[5]:


plt.figure(figsize=(12,5))
plt.title("Child Mortality: Death of children under 5 years of age per 1000 live births")
ax = sns.histplot(Pov_data["child_mort"])  #纵坐标表示国家总数，countries


# In[6]:


# pearson
plt.figure(figsize=(15,10))
sns.heatmap(Pov_data.corr(method='pearson', min_periods=1),annot=True)


# In[7]:


Pov_data.corr()
pd.plotting.scatter_matrix(Pov_data,figsize=[20,20])
plt.show()


# In[8]:
```

```
# TO DO 1:
# if the child_mort increase, the total_far will increase as well,
        # they are positive associational relationship, and strong
correlative(0.85).
# if income increase, the gdpp will increase,
        # they are positive associational relationship, and strong
correlative(0.9).
# there is a negative correlative relation beween income and child_mort, the degree
of linear association is -0.52.
# the relationship between exports and inflation is negative and weak(-0.11)


# In[9]:


# 1.3 Scaling


# In[10]:


Pov_data_Drop = Pov_data.drop(['country'], axis =1)
Pov_data_Drop.head()

# eliminate the column. Save the new dataset as Pov_data_Drop
# so you have a backup of  original dataset just in case!


# In[11]:


# Columns argument ==> we'll use later to create a new datafarame with the rescaled
data
columns = Pov_data_Drop.columns
scaler = MinMaxScaler() # fot the rescaling
# 'fit' function is to find the x_min and the x_man
#'transform' function applies formula to all elements of data

normalised_dataset = scaler.fit_transform(Pov_data_Drop)
normalised_dataset

My_normalised_df = pd.DataFrame(data = normalised_dataset, columns = columns )
My_normalised_df


# In[12]:


from sklearn.preprocessing import StandardScaler


# In[13]:


scaler = StandardScaler()
```

```
# In[14]:


print(scaler.fit(Pov_data_Drop))


# In[15]:


print(scaler.mean_)


# In[16]:


print(scaler.transform(Pov_data_Drop))


# In[17]:


# 2 Linear Regression Model


# In[18]:


# Perform step 1:3 first:

# 1) Import data and save it as ‘mpi_ds’
# 2) Observe the features
# 3) Drop 'ISO','Headcount Ratio Urban','Intensity of Deprivation
    # Urban','Headcount Ratio Rural','Intensity of Deprivation Rural' columns
    # and save the new dataset as ‘my_mpi_ds’

mpi_ds =  pd.read_csv ('MPI_Dataset.csv')


# In[19]:


mpi_ds


# In[20]:


my_mpi_ds = mpi_ds.drop(['ISO','Headcount Ratio Urban','Intensity of Deprivation
Urban',
                         'Headcount Ratio Rural','Intensity of Deprivation
Rural'],axis =1)


# In[21]:


#Rename the column heading as below
my_mpi_ds.rename(
columns = {'Country':'country','MPI Urban':'mpi_urban','MPI
```

```
Rural':'mpi_rural'},inplace = True)


# In[22]:


my_mpi_ds.head(3)


# In[23]:


combined = pd.merge(Pov_data,my_mpi_ds,on='country',how='inner')
combined.head()


# In[24]:


# TO DO 2
# Perform correlation analysis on your new dataset (‘combined’). Provide the
correlation
# matrix output and explain your finding
#there is a negative correlative relation between child_mort and life_expec.
#the correlation R between mpi_urban and mpi_rural is close to 1.

# Is there any multicollinearity within the features?
# yes, the total fertility is increasing mpi_urban and mpi_rural is increasing


# In[25]:


#w5
plt.figure(figsize= (10,5))
sns.heatmap(combined.corr(method='pearson', min_periods=1), annot= True)


# In[26]:


Pov_data.corr()
pd.plotting.scatter_matrix(combined,figsize=[20,20])
plt.show()


# In[27]:


reg = linear_model.LinearRegression()#linear regression class object
import statsmodels.api as sm
from statsmodels.formula.api import ols # libraries for plotting of residual plots


# In[28]:


#fit simple linear regression model
model = ols('mpi_urban ~ child_mort', data=combined).fit()
```

```
#print model summary
print(model.summary())
#adjust figure size
fig = plt.figure(figsize=(12,8))
#generate regression plots
fig = sm.graphics.plot_regress_exog(model, 'child_mort', fig=fig)


# In[29]:


# To Do 3: Create the model for remaining predictors and provide the results only.
Your result report
# must be well formatted and readable.


# In[30]:


#fit simple linear regression model
model = ols('life_expec ~ child_mort', data=combined).fit()
#print model summary
print(model.summary())
#adjust figure size
fig = plt.figure(figsize=(12,8))
#generate regression plots
fig = sm.graphics.plot_regress_exog(model, 'child_mort', fig=fig)


# In[31]:


# Lab 5


# In[32]:


# create linear regression class object
reg = linear_model.LinearRegression()
# libraries for plotting of residual plots
import statsmodels.api as sm
from statsmodels.formula.api import ols


# In[33]:


#fit simple linear regression model
model = ols('mpi_urban ~ child_mort', data=combined).fit()
#view model summary
print(model.summary())
#define figure size
fig = plt.figure(figsize=(12,8))
#produce regression plots
fig = sm.graphics.plot_regress_exog(model, 'child_mort', fig=fig)


# In[ ]:
```

```
# In[34]:


# 2 Multiple Independent Variables


# In[43]:


reg.fit(combined[['child_mort','exports','health','imports','income','inflation','l
ife_expec','total_fer','gdpp']],
        combined.mpi_urban)


# In[44]:


reg.score(combined[['child_mort','exports','health','imports','income','inflation',
'life_expec','total_fer','gdpp']],combined.mpi_urban)


# In[45]:


1- (1-
reg.score(combined[['child_mort','exports','health','imports','income','inflation',
'life_expec','total_fer','gdpp']],combined.mpi_urban))*(len
(combined.mpi_urban)-1)/(len(combined.mpi_urban)-
combined[['child_mort','exports','health','imports','income','inflation','life_expe
c','total_fer','gdpp']].shape[1]-1)


# In[38]:


Model1 = ols('mpi_urban
~child_mort+exports+health+imports+income+inflation+life_expec+total_fer+gdpp',
data=combined).fit()
print(Model1.summary())


# In[39]:


# Figure 1: Result of Model1: fitting all available independent variables to
predict ‘mpi_urban’


# In[40]:


# To Do 1: Create a second model without features with multicollinearity and
heteroscedasticity.
# Provide the code and complete Table1.
```

```
# In[46]:


# pearson
plt.figure(figsize= (10,5))
sns.heatmap(combined.corr(method='pearson', min_periods=1), annot= True)


# In[42]:


# remove 相关性低的数据


# In[48]:


Model1 = ols('mpi_urban ~child_mort+life_expec+total_fer', data=combined).fit()
print(Model1.summary())


# In[82]:


# To do 2
#


# In[49]:


Model1 = ols('mpi_urban ~total_fer+mpi_rural+child_mort', data=combined).fit()
print(Model1.summary())


# In[ ]:


# TO DO 3


# In[ ]:


#              R^2.       Adj R^2
# Model 1    0,835        0.813
# Model 2    0.805.       0.797
# Model 4    0.886        0.881


# In[ ]:





# In[ ]:
```