Data Science Project on GDP Analysis with Python

In [1]:

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import seaborn as sns
from matplotlib import pyplot as plt

 from sklearn.preprocessing import LabelEncoder
 from sklearn.model_selection import train_test_split
 from sklearn.linear_model import LinearRegression
 from sklearn.tree import DecisionTreeRegressor
 from sklearn.ensemble import RandomForestRegressor
 from sklearn.metrics import mean_squared_error, mean_squared_log_error
```

In [2]:

```python
data = pd.read_csv('C:/Users/DELL/Downloads/world.csv',decimal=',')
print('number of missing data:')
print(data.isnull().sum())
data.describe(include='all')
```

```
number of missing data:
Country                                0
Region                                 0
Population                             0
Area (sq. mi.)                         0
Pop. Density (per sq. mi.)             0
Coastline (coast/area ratio)           0
Net migration                          3
Infant mortality (per 1000 births)     3
GDP ($ per capita)                     1
Literacy (%)                          18
Phones (per 1000)                      4
Arable (%)                             2
Crops (%)                              2
Other (%)                              2
Climate                               22
Birthrate                              3
Deathrate                              4
Agriculture                           15
Industry                              16
Service                               15
dtype: int64 Out[2]:
```

	Country	Region	Population	Area (sq. mi.)	Pop. Density (per sq. mi.)	Coastline (coast/area ratio)	Net migration
count	227	227	2.270000e+02	2.270000e+02	227.000000	227.000000	224.000000
unique	227	11	NaN	NaN	NaN	NaN	NaN
top	Jordan	SUB-SAHARAN AFRICA	NaN	NaN	NaN	NaN	NaN
freq	1	51	NaN	NaN	NaN	NaN	NaN
mean	NaN	NaN	2.874028e+07	5.982270e+05	379.047137	21.165330	0.038125
std	NaN	NaN	1.178913e+08	1.790282e+06	1660.185825	72.286863	4.889269
min	NaN	NaN	7.026000e+03	2.000000e+00	0.000000	0.000000	-20.990000
25%	NaN	NaN	4.376240e+05	4.647500e+03	29.150000	0.100000	-0.927500

50%	NaN	NaN	4.786994e+06	8.660000e+04	78.800000	0.730000	0.000000
75%	NaN	NaN	1.749777e+07	4.418110e+05	190.150000	10.345000	0.997500
max	NaN	NaN	1.313974e+09	1.707520e+07	16271.500000	870.660000	23.060000

In [3]:

```python
data.groupby('Region')[['GDP ($ per capita)','Literacy (%)','Agriculture']].median()
```

Out[3]:

	GDP ($ per capita)	Literacy (%)	Agriculture
Region			
ASIA (EX. NEAR EAST)	3450.0	90.60	0.1610
BALTICS	11400.0	99.80	0.0400
C.W. OF IND. STATES	3450.0	99.05	0.1980
EASTERN EUROPE	9100.0	98.60	0.0815
LATIN AMER. & CARIB	6300.0	94.05	0.0700
NEAR EAST	9250.0	83.00	0.0350
NORTHERN AFRICA	6000.0	70.00	0.1320
NORTHERN AMERICA	29800.0	97.50	0.0100
OCEANIA	5000.0	95.00	0.1505
SUB-SAHARAN AFRICA	1300.0	62.95	0.2760
WESTERN EUROPE	27200.0	99.00	0.0220

In [4]:

```python
for col in data.columns.values:
    if data[col].isnull().sum() == 0:
        continue
    if col == 'Climate':
        guess_values = data.groupby('Region')['Climate'].apply(lambda x: x.mode().max()
    else:
        guess_values = data.groupby('Region')[col].median()
    for region in data['Region'].unique():
        data[col].loc[(data[col].isnull())&(data['Region']==region)] = guess_values[reg
```

```
C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\indexing.py:1637: Set
tingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/
stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pand
```

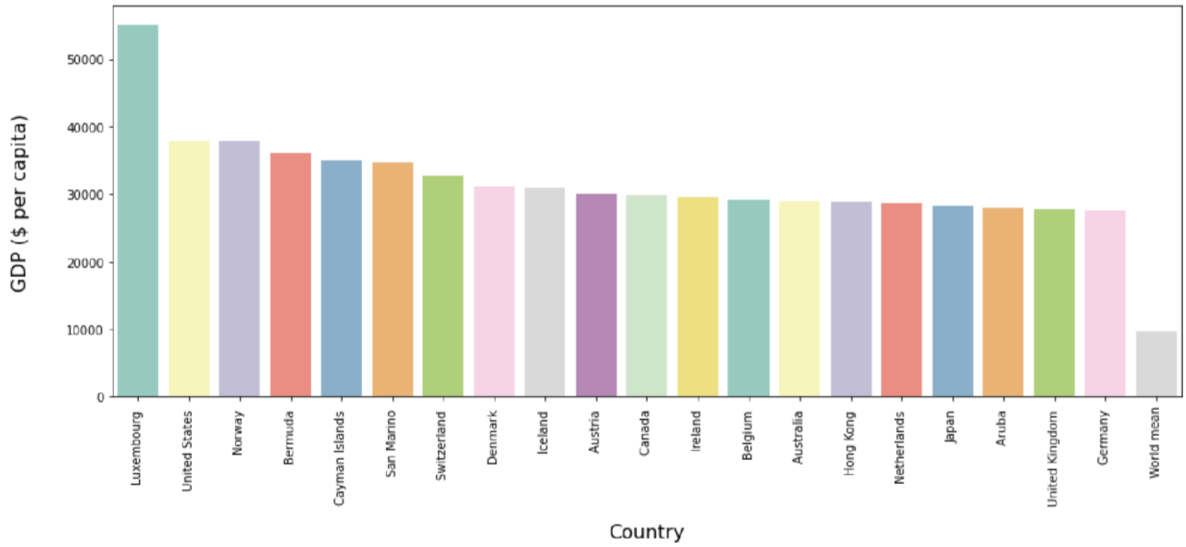
```
as.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v ersus-a-copy)
  self._setitem_single_block(indexer, value, name)
```

Data Exploration

Top Countries with highest GDP per capita

In [5]:

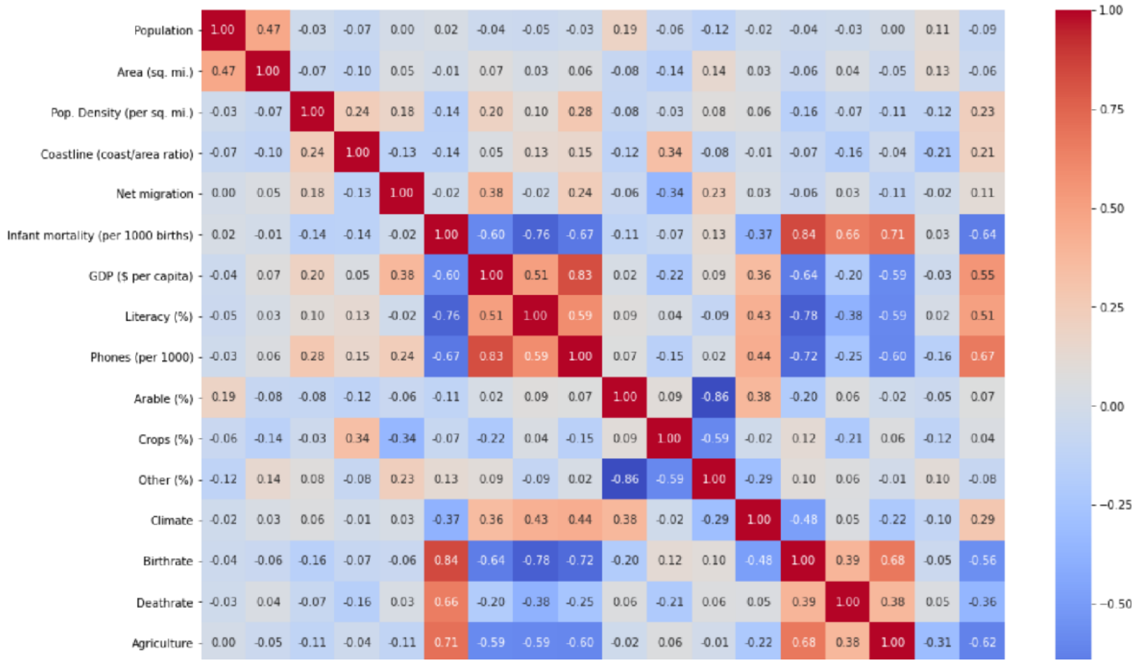
```
fig, ax = plt.subplots(figsize=(16,6))
#ax = fig.add_subplot(111)
top_gdp_countries = data.sort_values('GDP ($ per capita)',ascending=False).head(20)
mean = pd.DataFrame({'Country':['World mean'], 'GDP ($ per capita)':[data['GDP ($ per c
gdps = pd.concat([top_gdp_countries[['Country','GDP ($ per capita)']],mean],ignore_inde

sns.barplot(x='Country',y='GDP ($ per capita)',data=gdps, palette='Set3')
ax.set_xlabel(ax.get_xlabel(),labelpad=15)
ax.set_ylabel(ax.get_ylabel(),labelpad=30)
ax.xaxis.label.set_fontsize(16)
ax.yaxis.label.set_fontsize(16)
plt.xticks(rotation=90)
plt.show()
```



Correlation between Variables

In [6]:

```
plt.figure(figsize=(16,12))
sns.heatmap(data=data.iloc[:,2:].corr(),annot=True,fmt='.2f',cmap='coolwarm')
plt.show()
```

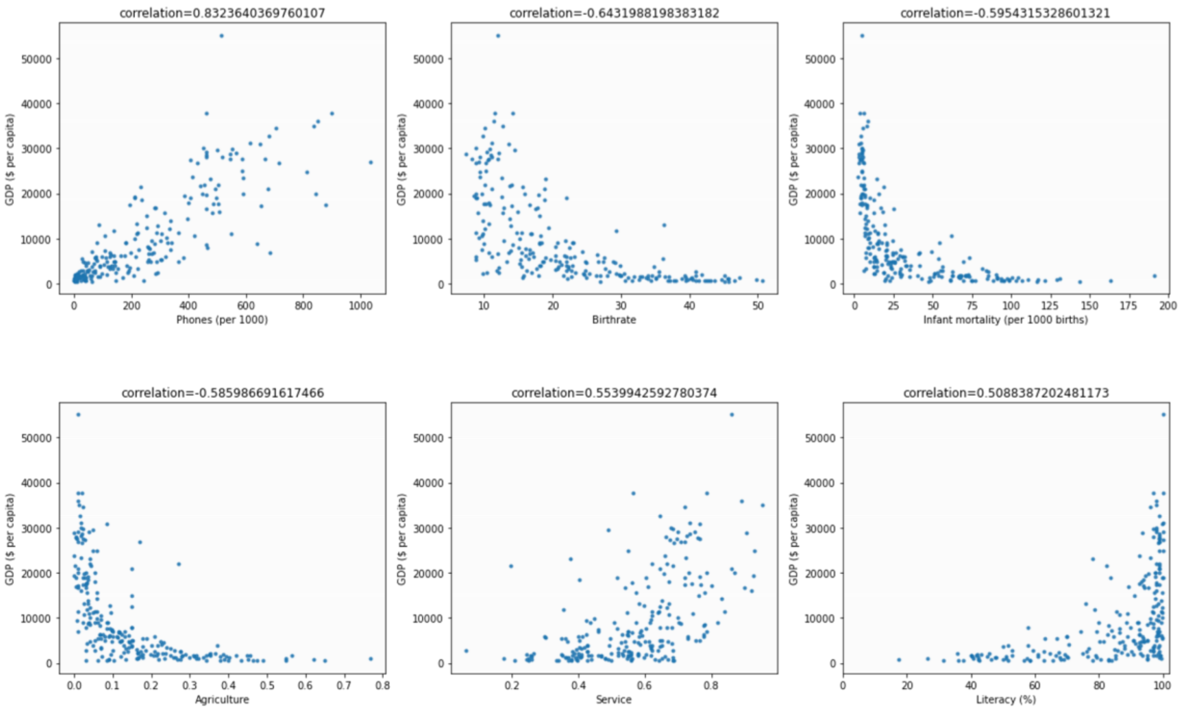


Top Factors affecting GDP per capita

In [7]:

```
fig, axes = plt.subplots(nrows=2, ncols=3, figsize=(20,12))
plt.subplots_adjust(hspace=0.4)

        corr_to_gdp = pd.Series()
        for col in data.columns.values[2:]:
        if ((col!='GDP ($ per capita)')&(col!='Climate')):
        corr_to_gdp[col] = data['GDP ($ per capita)'].corr(data[col])
        abs_corr_to_gdp = corr_to_gdp.abs().sort_values(ascending=False) corr_to_gdp
        = corr_to_gdp.loc[abs_corr_to_gdp.index]

                    for i in range(2):
                    for j in range(3):
                    sns.regplot(x=corr_to_gdp.index.values[i*3+j], y='GDP ($ per
                    capita)', data=dat
                    ax=axes[i,j], fit_reg=False, marker='.')
                    title = 'correlation='+str(corr_to_gdp[i*3+j])
                    axes[i,j].set_title(title)
                    axes[1,2].set_xlim(0,102)
                    plt.show()
```

```
<ipython-input-7-6c0fb2867ba9>:4: DeprecationWarning: The default dtype for
empty Series will be 'object' instead of 'float64' in a future version. Spec
ify a dtype explicitly to silence this warning.  corr_to_gdp = pd.Series()
```



In [8]:

```
data.loc[(data['Birthrate']<14)&(data['GDP ($ per capita)']<10000)]
```

Out[8]:

	Country	Region	Population	Area (sq. mi.)	Pop. Density (per sq. mi.)	Coastline (coast/area ratio)	Net migration	Infant mortality (per 1000 births)
9	Armenia	C.W. OF IND. STATES	2976372	29800	99.9	0.00	-6.47	23.28
18	Belarus	C.W. OF IND. STATES	10293011	207600	49.6	0.00	2.54	13.37
25	Bosnia & Herzegovina	EASTERN EUROPE	4498976	51129	88.0	0.04	0.31	21.05
30	Bulgaria	EASTERN EUROPE	7385367	110910	66.6	0.32	-4.58	20.55
42	China	ASIA (EX. NEAR EAST)	1313973713	9596960	136.9	0.15	-0.40	24.18
51	Cuba	LATIN AMER. & CARIB	11382820	110860	102.7	3.37	-1.58	6.33
75	Georgia	C.W. OF IND. STATES	4661473	69700	66.9	0.44	-4.70	18.59
123	Macedonia	EASTERN EUROPE	2050554	25333	80.9	0.00	-1.45	10.09
168	Romania	EASTERN EUROPE	22303552	237500	93.9	0.09	-0.13	26.43
169	Russia	C.W. OF IND. STATES	142893540	17075200	8.4	0.22	1.02	15.39
171	Saint Helena	SUB-SAHARAN AFRICA	7502	413	18.2	14.53	0.00	19.00
174	St Pierre & Miquelon	NORTHERN AMERICA	7026	242	29.0	49.59	-4.86	7.54
181	Serbia	EASTERN EUROPE	9396411	88361	106.3	0.00	-1.33	12.89

201	Thailand	ASIA (EX. NEAR EAST)	64631595	514000	125.7	0.63	0.00	20.48
204	Trinidad & Tobago	LATIN AMER. & CARIB	1065842	5128	207.9	7.06	-10.83	24.31
211	Ukraine	C.W. OF IND. STATES	46710816	603700	77.4	0.46	-0.39	20.34

Training and Testing

In [9]:

```
LE = LabelEncoder()
data['Region_label'] = LE.fit_transform(data['Region'])
data['Climate_label'] = LE.fit_transform(data['Climate'])
data.head()
```

Out[9]:

	Country	Region	Population	Area (sq. mi.)	Pop. Density (per sq. mi.)	Coastline (coast/area ratio)	Net migration	Infant mortality (per 1000 births)	GD cap
0	Afghanistan	ASIA (EX. NEAR EAST)	31056997	647500	48.0	0.00	23.06	163.07	70
1	Albania	EASTERN EUROPE	3581655	28748	124.6	1.26	-4.93	21.52	450
2	Algeria	NORTHERN AFRICA	32930091	2381740	13.8	0.04	-0.39	31.00	600
3	American Samoa	OCEANIA	57794	199	290.4	58.29	-20.71	9.27	800
4	Andorra	WESTERN EUROPE	71201	468	152.1	0.00	6.60	4.05	1900

5 rows × 22 columns

In [10]:

```
        train, test = train_test_split(data, test_size=0.3, shuffle=True)
        training_features = ['Population', 'Area (sq. mi.)',
```

```
        'Pop. Density (per sq. mi.)', 'Coastline (coast/area ratio)',
        'Net migration', 'Infant mortality (per 1000 births)',
        'Literacy (%)', 'Phones (per 1000)',
        'Arable (%)', 'Crops (%)', 'Other (%)', 'Birthrate',
        'Deathrate', 'Agriculture', 'Industry', 'Service', 'Region_label',
        'Climate_label','Service']
        target = 'GDP ($ per capita)'
        train_X = train[training_features]
        train_Y = train[target]
        test_X = test[training_features]
        test_Y = test[target]
```

In [11]:

```
model = LinearRegression()
model.fit(train_X, train_Y)
train_pred_Y = model.predict(train_X)
test_pred_Y = model.predict(test_X)
train_pred_Y = pd.Series(train_pred_Y.clip(0, train_pred_Y.max()), index=train_Y.index)
test_pred_Y = pd.Series(test_pred_Y.clip(0, test_pred_Y.max()), index=test_Y.index)

rmse_train = np.sqrt(mean_squared_error(train_pred_Y, train_Y))
msle_train = mean_squared_log_error(train_pred_Y, train_Y)
rmse_test = np.sqrt(mean_squared_error(test_pred_Y, test_Y))
msle_test = mean_squared_log_error(test_pred_Y, test_Y)

print('rmse_train:',rmse_train,'msle_train:',msle_train)
print('rmse_test:',rmse_test,'msle_test:',msle_test)
```

```
rmse_train: 4627.28148511499 msle_train: 5.03226574977708 rmse_test:
5244.298101394129 msle_test: 4.794097282847199
```

In [12]:

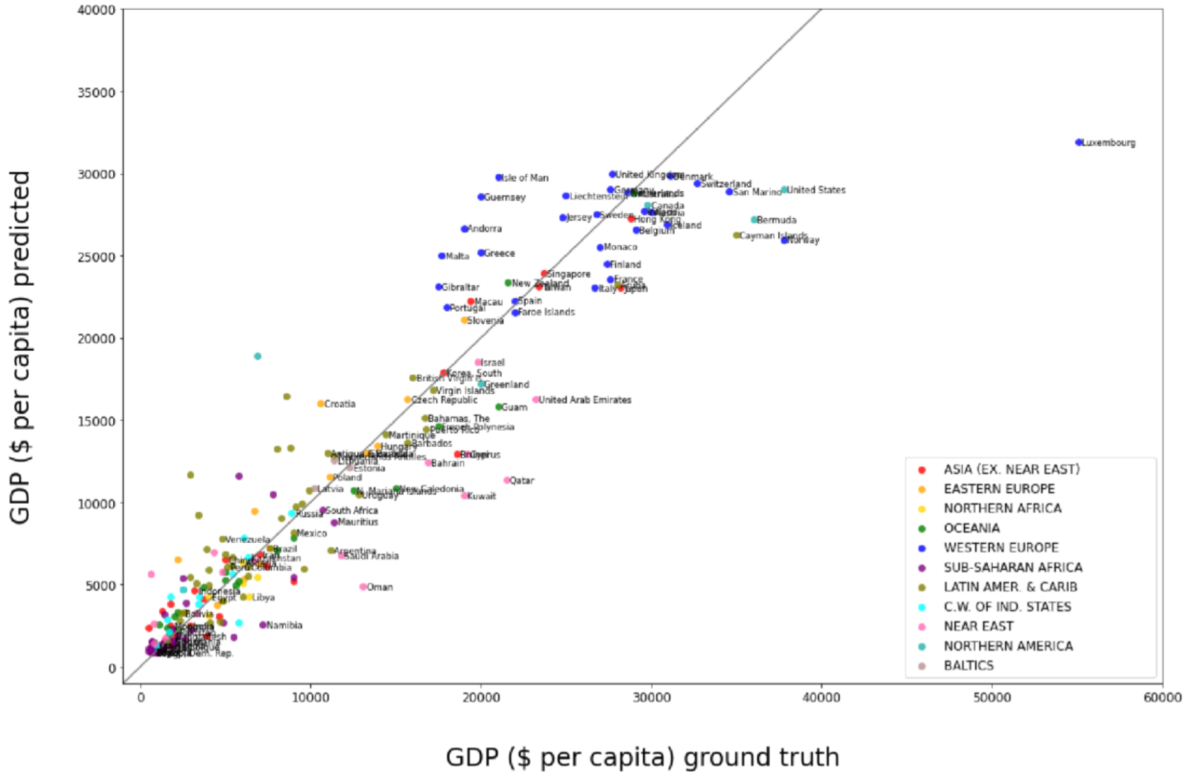
```
model = RandomForestRegressor(n_estimators = 50,
max_depth = 6,
min_weight_fraction_leaf = 0.05,
max_features = 0.8,
random_state = 42)
model.fit(train_X, train_Y)
train_pred_Y = model.predict(train_X)
test_pred_Y = model.predict(test_X)
train_pred_Y = pd.Series(train_pred_Y.clip(0, train_pred_Y.max()),
index=train_Y.index)
test_pred_Y = pd.Series(test_pred_Y.clip(0, test_pred_Y.max()),
index=test_Y.index)

rmse_train = np.sqrt(mean_squared_error(train_pred_Y, train_Y))
msle_train = mean_squared_log_error(train_pred_Y, train_Y)
rmse_test = np.sqrt(mean_squared_error(test_pred_Y, test_Y))
msle_test = mean_squared_log_error(test_pred_Y, test_Y)

print('rmse_train:',rmse_train,'msle_train:',msle_train)
print('rmse_test:',rmse_test,'msle_test:',msle_test)
```

```
rmse_train: 3196.8973340924636 msle_train: 0.1617009331866535 rmse_test:
3997.795644303798 msle_test: 0.22654287905590084
```

Visualization of Results

In [13]:

```python
plt.figure(figsize=(18,12))

train_test_Y = train_Y.append(test_Y)
train_test_pred_Y = train_pred_Y.append(test_pred_Y)

data_shuffled = data.loc[train_test_Y.index]
label = data_shuffled['Country']

colors = {'ASIA (EX. NEAR EAST)         ':'red',
          'EASTERN EUROPE                     ':'orange',
          'NORTHERN AFRICA                    ':'gold',
          'OCEANIA                            ':'green',
          'WESTERN EUROPE                     ':'blue',
          'SUB-SAHARAN AFRICA                 ':'purple',
          'LATIN AMER. & CARIB    ':'olive',
          'C.W. OF IND. STATES ':'cyan',
          'NEAR EAST                          ':'hotpink',
          'NORTHERN AMERICA                   ':'lightseagreen',
          'BALTICS                            ':'rosybrown'}

for region, color in colors.items():
    X = train_test_Y.loc[data_shuffled['Region']==region]
    Y = train_test_pred_Y.loc[data_shuffled['Region']==region]
    ax = sns.regplot(x=X, y=Y, marker='.', fit_reg=False, color=color, scatter_kws={'s'
plt.legend(loc=4,prop={'size': 12})

ax.set_xlabel('GDP ($ per capita) ground truth',labelpad=40)
ax.set_ylabel('GDP ($ per capita) predicted',labelpad=40)
ax.xaxis.label.set_fontsize(24)
ax.yaxis.label.set_fontsize(24)
ax.tick_params(labelsize=12)

x = np.linspace(-1000,50000,100) # 100 linearly spaced numbers
y = x
plt.plot(x,y,c='gray')

plt.xlim(-1000,60000)
plt.ylim(-1000,40000)

for i in range(0,train_test_Y.shape[0]):
    if((data_shuffled['Area (sq. mi.)'].iloc[i]>8e5) |
       (data_shuffled['Population'].iloc[i]>1e8) |
```



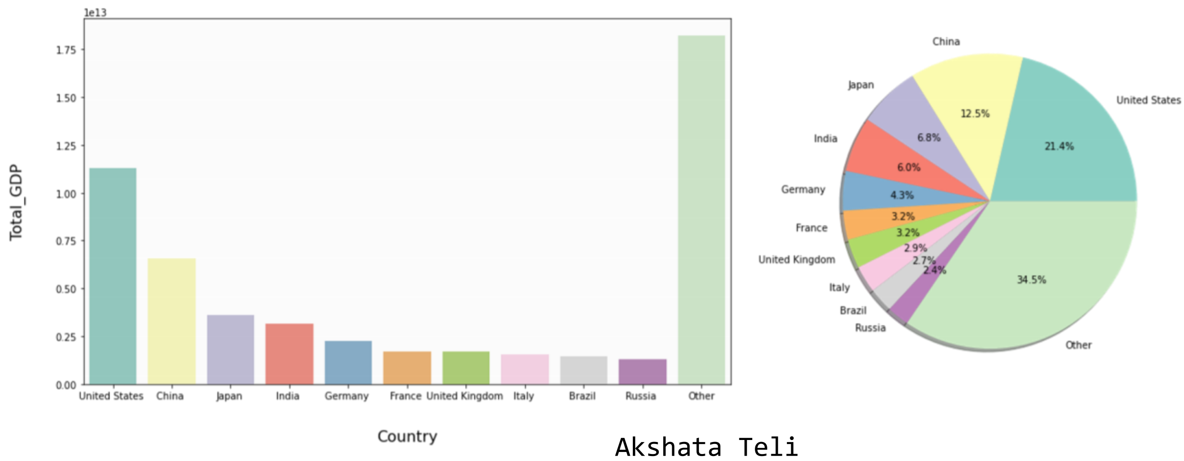

```
        (data_shuffled['GDP ($ per capita)'].iloc[i]>10000)):
        plt.text(train_test_Y.iloc[i]+200, train_test_pred_Y.iloc[i]-200, label.iloc[i]
```

Total GDP

Let's compare the above ten countries' rank in total GDP and GDP per capita.6

In [14]:

```
data['Total_GDP ($)'] = data['GDP ($ per capita)'] * data['Population']
#plt.figure(figsize=(16,6))
top_gdp_countries = data.sort_values('Total_GDP ($)',ascending=False).head(10)
other = pd.DataFrame({'Country':['Other'], 'Total_GDP ($)':[data['Total_GDP ($)'].sum(
gdps = pd.concat([top_gdp_countries[['Country','Total_GDP ($)']],other],ignore_index=Tr

fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(20,7),gridspec_kw = {'width_ratios
sns.barplot(x='Country',y='Total_GDP ($)',data=gdps,ax=axes[0],palette='Set3')
axes[0].set_xlabel('Country',labelpad=30,fontsize=16)
axes[0].set_ylabel('Total_GDP',labelpad=30,fontsize=16)

colors = sns.color_palette("Set3", gdps.shape[0]).as_hex()
axes[1].pie(gdps['Total_GDP ($)'], labels=gdps['Country'],colors=colors,autopct='%1.1f%
axes[1].axis('equal')
plt.show()
```



In [15]:

```
Rank1 = data[['Country','Total_GDP ($)']].sort_values('Total_GDP ($)', ascending=False)
Rank2 = data[['Country','GDP ($ per capita)']].sort_values('GDP ($ per capita)', ascend
 Rank1 = pd.Series(Rank1.index.values+1, index=Rank1.Country)
 Rank2 = pd.Series(Rank2.index.values+1, index=Rank2.Country)
 Rank_change = (Rank2-Rank1).sort_values(ascending=False)
 print('rank of total GDP - rank of GDP per capita:')
Rank_change.loc[top_gdp_countries.Country] rank of total GDP - rank of GDP per capita:
```

Out[15]:

```
Country
United States        1
China              118
Japan               14
India              146
Germany             15
France              15
United Kingdom      12
Italy               17
Brazil              84 Russia
75 dtype: int64
```

Factors affecting Total GDP

In [16]:

```
    corr_to_gdp = pd.Series()
    for col in data.columns.values[2:]:
    if ((col!='Total_GDP ($)')&(col!='Climate')&(col!='GDP ($ per capita)')):
    corr_to_gdp[col] = data['Total_GDP ($)'].corr(data[col]) abs_corr_to_gdp =
    corr_to_gdp.abs().sort_values(ascending=False)
corr_to_gdp = corr_to_gdp.loc[abs_corr_to_gdp.index]
print(corr_to_gdp)
```

```
Population                            0.639528
Area (sq. mi.)                        0.556396
Phones (per 1000)                     0.233484
Birthrate                            -0.166889
Agriculture                          -0.139516
Arable (%)                            0.129928
Climate_label                         0.125791
Infant mortality (per 1000 births)   -0.122076
Literacy (%)                          0.099417
Service                               0.085096
Region_label                         -0.079745
Crops (%)                            -0.077078
Coastline (coast/area ratio)         -0.065211
```

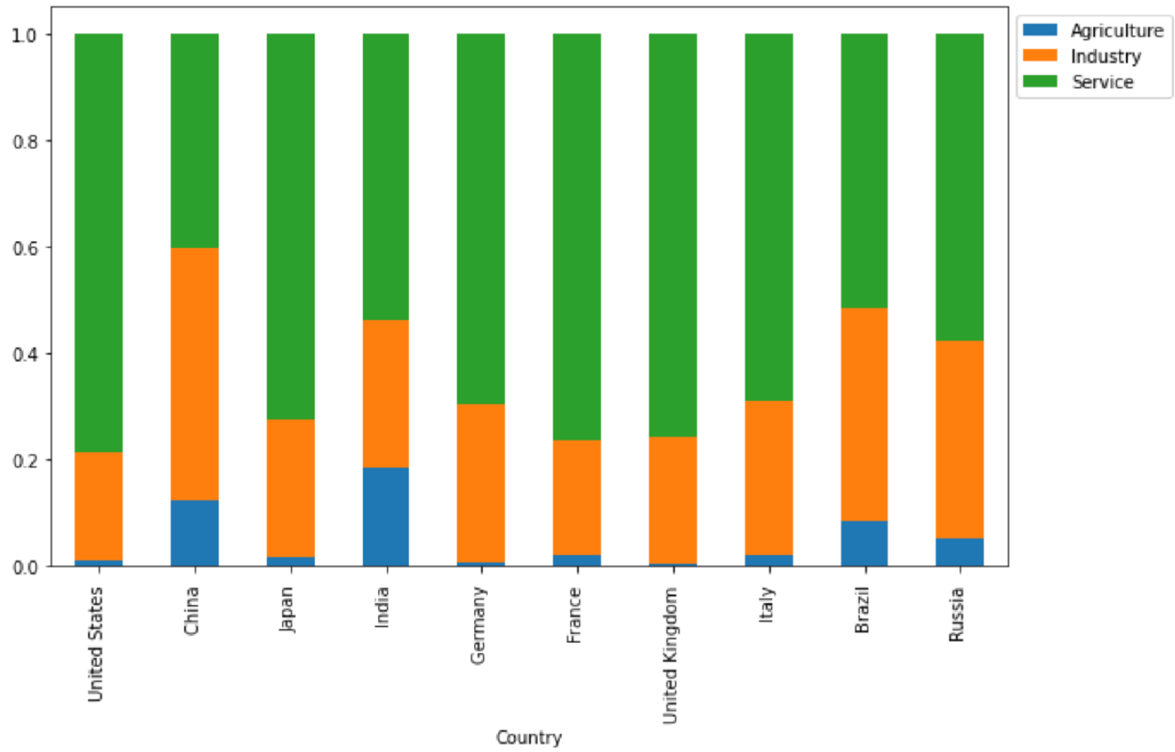
```
Other (%)                           -0.064882
Net migration                        0.054632
Industry                             0.050399
Deathrate                           -0.035820 Pop.
Density (per sq. mi.)          -0.028487 dtype:
float64

<ipython-input-16-97518c8494d5>:1: DeprecationWarning: The default dtype for
empty Series will be 'object' instead of 'float64' in a future version. Spec
ify a dtype explicitly to silence this warning.  corr_to_gdp = pd.Series()
```

Comparison of the Top 101

Finally, let us do a comparison of the economy structure for the ten countries with highest total GDP.

In [17]:

```python
plot_data = top_gdp_countries.head(10)[['Country','Agriculture', 'Industry', 'Service']]
plot_data = plot_data.set_index('Country')
ax = plot_data.plot.bar(stacked=True,figsize=(10,6))
ax.legend(bbox_to_anchor=(1, 1))
plt.show()
```



As well as their land usage:

In [18]:

```
plot_data = top_gdp_countries[['Country','Arable (%)', 'Crops (%)', 'Other (%)']]
plot_data = plot_data.set_index('Country')
ax = plot_data.plot.bar(stacked=True,figsize=(10,6))
ax.legend(bbox_to_anchor=(1, 1))
plt.show()
```