DATA SCIENCE PROJECT ON GDP ANALYSIS WITH PYTHON

Khushi Kuldeep
UG Student, Department of Mathematics and bio-informatics,
Maulana Azad National Institute of Technology,
Madhya Pradesh, India.

Abstract- *With technology seeping into every aspect of our lives, the country's economy is no exception. The Data Science process involves analyzing, visualizing, extracting, managing, and storing data to generate insights from analytics. Data Science is rapidly becoming one of the most widely used technologies, and it is assisting governments in maintaining transparency, becoming far more efficient, and boosting the economy and productivity. Using statistics, data preparation, predictive modeling, and machine learning, it attempts to resolve various problems within different areas and the economy on the whole. As technology continues to progress, the influence that data science has on the world around us will grow.*

Keywords: Gross Domestic Product (GDP), Data Science, economic statistics.

1.INTRODUCTION

Gross Domestic Product (GDP) is the final value of all the economic goods and services produced within the country's geographic boundaries during a specified period of time. GDP growth rate is the major indicator of a country's economic performance. Broadly speaking, the primary sector (agricultural), the secondary sector (industry), and the tertiary sector (services) all contribute to GDP by producing goods and services (services). Gross Domestic Product (GDP) is a key tool that guides investors, policymakers, and businesses in strategic decision-making. Per capita GDP is a global indicator of a country's economy that economists use in combination with GDP to assess a country's wealth based on its economic growth. The formula of GDP per capita is:

GDP per capita = Gross Domestic Product (GDP) / Population

1.1 Abbreviations and Acronyms

GDP- Gross Domestic Product

IDE- Integrated Development Environment

2. OBJECTIVE

The primary goal of this project is to investigate the dataset "Countries of the World" and to focus on the elements that are influencing a Country's GDP per capita.

3. METHODOLOGY

3.1 Installation

1. Install python 3.3 or greater, or Python 2.7.
2. Select an IDE (Integrated Development Environment) as the working environment.
   For example, Jupyter Notebook.
3. Install Jupyter Notebook using Anaconda or pip.
4. Jupyter Notebook in Anaconda comes pre-installed. The main advantage of anaconda is that it has over 720 packages to access, that can be easily installed with Anaconda's Conda package.
5. You only need to make sure you have the newest version of pip if you don't want to install Anaconda. If you have installed Python, you will typically already have pip.
6. Open command prompt and Upgrade pip using:
   pip install --upgrade pip
7. Install the Jupyter Notebook using:
   pip install jupyter
8. Then launch Jupyter notebook using:
   jupyter notebook
9. Instructions to install the python libraries are:
   pip install NumPy
   pip install pandas
   pip install seaborn
   pip install matplotlib
   pip install SciPy
   pip install sklearn

3.2 Implementation

Although every data science job is different, here is one way to visualize data science:

1. Data Collection

   Data can come from a variety of sources like from our local machine, query SQL servers, or the internet.
   For this project, the data titled "Countries of the World" is imported from Kaggle.

2. Data Cleaning

   Data cleaning is also referred to as data preparation, is a vital step that comprises reformatting the data, making data corrections, and merging data sets to enhance the data.

3. Data Exploration

   Data Exploration is used to explore and visualize data to derive insights from the start or identify patterns to dig deeper.

4. Training and Testing

   A training set is used to develop a model in a dataset, whereas a test (or validation) set is used to validate the model built.

4. EXPERIMENTAL WORK AND RESULTS

Importing the required python libraries

```
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
```

Data

```
In [2]: data = pd.read_csv('world.csv',decimal=',')
        print('number of missing data:')
        print(data.isnull().sum())
        data.describe(include='all')
```

Output:

```
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
```

	Country	Region	Population	Area (sq. mi.)	Pop. Density (per sq. mi.)	Coastline (coast/area ratio)	Net migration	Infant mortality (per 1000 births)	GDP ($ per capita)	Literacy (%)	Phones (per 1000)
count	227	227	2.270000e+02	2.270000e+02	227.000000	227.000000	224.000000	224.000000	226.000000	209.000000	223.000000
unique	227	11	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
top	Haiti	SUB-SAHARAN AFRICA	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
freq	1	51	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
mean	NaN	NaN	2.874028e+07	5.982270e+05	379.047137	21.165330	0.038125	35.506964	9689.823009	82.838278	236.061435
std	NaN	NaN	1.178913e+08	1.790282e+06	1660.185825	72.286863	4.889269	35.389899	10049.138513	19.722173	227.991829
min	NaN	NaN	7.026000e+03	2.000000e+00	0.000000	0.000000	-20.990000	2.290000	500.000000	17.600000	0.200000
25%	NaN	NaN	4.376240e+05	4.647500e+03	29.150000	0.100000	-0.927500	8.150000	1900.000000	70.600000	37.800000
50%	NaN	NaN	4.786994e+06	8.660000e+04	78.800000	0.730000	0.000000	21.000000	5550.000000	92.500000	176.200000
75%	NaN	NaN	1.749777e+07	4.418110e+05	190.150000	10.345000	0.997500	55.705000	15700.000000	98.000000	389.650000
max	NaN	NaN	1.313974e+09	1.707520e+07	16271.500000	870.660000	23.060000	191.190000	55100.000000	100.000000	1035.600000

Data Preparation

The missing data in table is filled in by using the median of the region to which a country belongs because geologically close countries are often similar in many aspects.

In [3]:
```
data.groupby('Region')[['GDP ($ per capita)','Literacy (%)','Agriculture']].median()
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[region]
```

Data Exploration

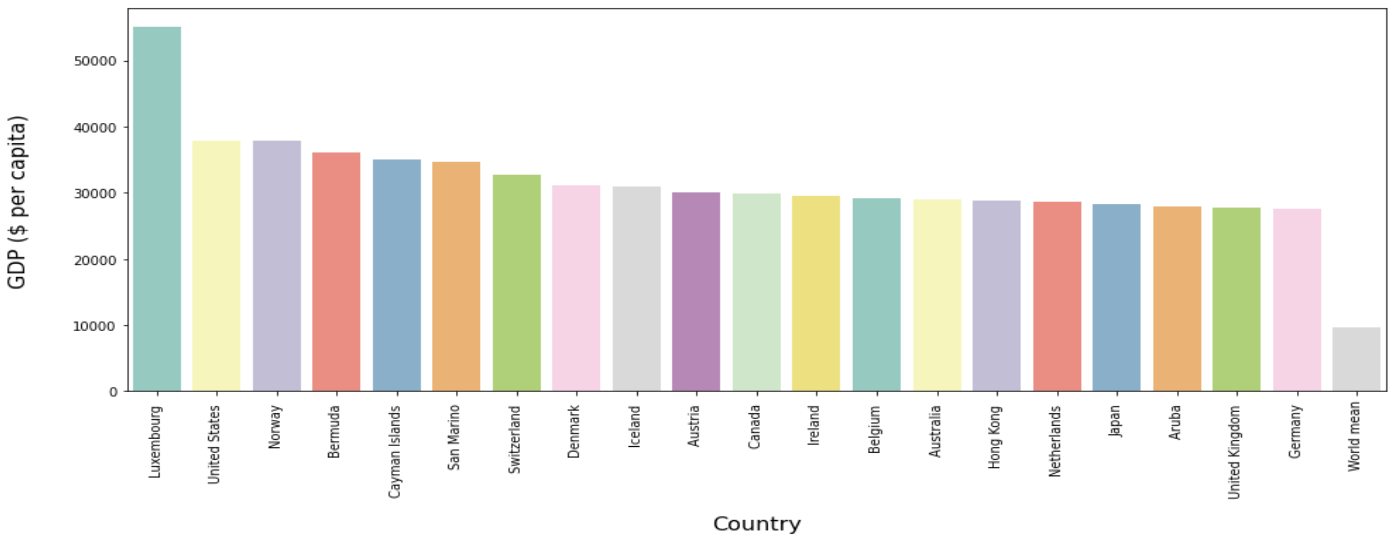
Top Countries with highest GDP per capita

The bar graph of the top countries with the highest GDP per capita is built.

In [4]:
```
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 capita)'].mean()]})
gdps = pd.concat([top_gdp_countries[['Country','GDP ($ per capita)']],mean],ignore_index=True)

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()
```

Output:

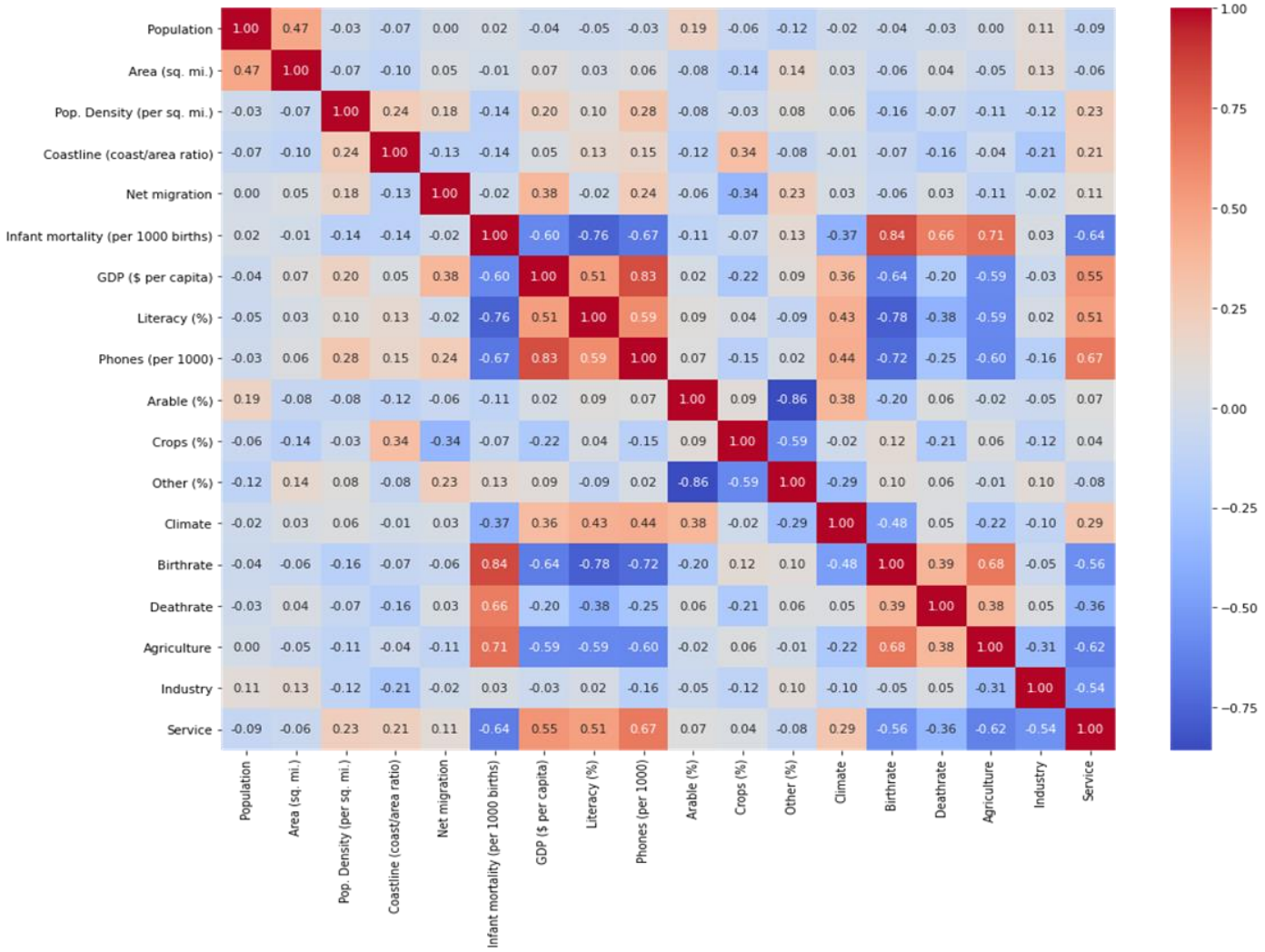


Correlation between Variables

The heatmap that depicts the correlation between the numerical columns is constructed.

In [5]:
```
plt.figure(figsize=(16,12))
sns.heatmap(data=data.iloc[:,2:].corr(),annot=True,fmt='.2f',cmap='coolwarm')
plt.show()
```

Output:


Population 1.00 0.47 -0.03 -0.07 0.00 0.02 -0.04 -0.05 -0.03 0.19 -0.06 -0.12 -0.02 -0.04 -0.03 0.00 0.11 -0.09
Area (sq. mi.) 0.47 1.00 -0.07 -0.10 0.05 -0.01 0.07 0.03 0.06 -0.08 -0.14 0.14 0.03 -0.06 0.04 -0.05 0.13 -0.06
Pop. Density (per sq. mi.) -0.03 -0.07 1.00 0.24 0.18 -0.14 0.20 0.10 0.28 -0.08 -0.03 0.08 0.06 -0.16 -0.07 -0.11 -0.12 0.23
Coastline (coast/area ratio) -0.07 -0.10 0.24 1.00 -0.13 -0.14 0.05 0.13 0.15 -0.12 0.34 -0.08 -0.01 -0.07 -0.16 -0.04 -0.21 0.21
Net migration 0.00 0.05 0.18 -0.13 1.00 -0.02 0.38 -0.02 0.24 -0.06 -0.34 0.23 0.03 -0.06 0.03 -0.11 -0.02 0.11
Infant mortality (per 1000 births) 0.02 -0.01 -0.14 -0.14 -0.02 1.00 -0.60 -0.76 -0.67 -0.11 -0.07 0.13 -0.37 0.84 0.66 0.71 0.03 -0.64
GDP ($ per capita) -0.04 0.07 0.20 0.05 0.38 -0.60 1.00 0.51 0.83 0.02 -0.22 0.09 0.36 -0.64 -0.20 -0.59 -0.03 0.55
Literacy (%) -0.05 0.03 0.10 0.13 -0.02 -0.76 0.51 1.00 0.59 0.09 0.04 -0.09 0.43 -0.78 -0.38 -0.59 0.02 0.51
Phones (per 1000) -0.03 0.06 0.28 0.15 0.24 -0.67 0.83 0.59 1.00 0.07 -0.15 0.02 0.44 -0.72 -0.25 -0.60 -0.16 0.67
Arable (%) 0.19 -0.08 -0.08 -0.12 -0.06 -0.11 0.02 0.09 0.07 1.00 0.09 -0.86 0.38 -0.20 0.06 -0.02 -0.05 0.07
Crops (%) -0.06 -0.14 -0.03 0.34 -0.34 -0.07 -0.22 0.04 -0.15 0.09 1.00 -0.59 -0.02 0.12 -0.21 0.06 -0.12 0.04
Other (%) -0.12 0.14 0.08 -0.08 0.23 0.13 0.09 -0.09 0.02 -0.86 -0.59 1.00 -0.29 0.10 0.06 -0.01 0.10 -0.08
Climate -0.02 0.03 0.06 -0.01 0.03 -0.37 0.36 0.43 0.44 0.38 -0.02 -0.29 1.00 -0.48 0.05 -0.22 -0.10 0.29
Birthrate -0.04 -0.06 -0.16 -0.07 -0.06 0.84 -0.64 -0.78 -0.72 -0.20 0.12 0.10 -0.48 1.00 0.39 0.68 -0.05 -0.56
Deathrate -0.03 0.04 -0.07 -0.16 0.03 0.66 -0.20 -0.38 -0.25 0.06 -0.21 0.06 0.05 0.39 1.00 0.38 0.05 -0.36
Agriculture 0.00 -0.05 -0.11 -0.04 -0.11 0.71 -0.59 -0.59 -0.60 -0.02 0.06 -0.01 -0.22 0.68 0.38 1.00 -0.31 -0.62
Industry 0.11 0.13 -0.12 -0.21 -0.02 0.03 -0.03 0.02 -0.16 -0.05 -0.12 0.10 -0.10 -0.05 0.05 -0.31 1.00 -0.54
Service -0.09 -0.06 0.23 0.21 0.11 -0.64 0.55 0.51 0.67 0.07 0.04 -0.08 0.29 -0.56 -0.36 -0.62 -0.54 1.00
Population
Area (sq. mi.)
Pop. Density (per sq. mi.)
Coastline (coast/area ratio)
Net migration
Infant mortality (per 1000 births)
GDP ($ per capita)
Literacy (%)
Phones (per 1000)
Arable (%)
Crops (%)
Other (%)
Climate
Birthrate
Deathrate
Agriculture
Industry
Service
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75

Modeling

Training and Testing

All the features given in the data set are used without further feature engineering.

In [8]:
```
LE = LabelEncoder()
data['Region_label'] = LE.fit_transform(data['Region'])
data['Climate_label'] = LE.fit_transform(data['Climate'])
data.head()
```

Output:

	Country	Region	Population	Area (sq. mi.)	Pop. Density (per sq. mi.)	Coastline (coast/area ratio)	Net migration	Infant mortality (per 1000 births)	GDP ($ per capita)	Literacy (%)	...	Crops (%)	Other (%)	Climate	Birthrate	Deathrate
0	Afghanistan	ASIA (EX. NEAR EAST)	31056997	647500	48.0	0.00	23.06	163.07	700.0	36.0	...	0.22	87.65	1.0	46.60	20.34
1	Albania	EASTERN EUROPE	3581655	28748	124.6	1.26	-4.93	21.52	4500.0	86.5	...	4.42	74.49	3.0	15.11	5.22
2	Algeria	NORTHERN AFRICA	32930091	2381740	13.8	0.04	-0.39	31.00	6000.0	70.0	...	0.25	96.53	1.0	17.14	4.61
3	American Samoa	OCEANIA	57794	199	290.4	58.29	-20.71	9.27	8000.0	97.0	...	15.00	75.00	2.0	22.46	3.27
4	Andorra	WESTERN EUROPE	71201	468	152.1	0.00	6.60	4.05	19000.0	100.0	...	0.00	97.78	3.0	8.71	6.25

5 rows × 22 columns

In [9]:
```
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]
```

First, the linear regression model is used. Both root mean squared error and mean squared log error were assessed, as for metric.

In [10]:
```
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)
```

Output:

```
rmse_train: 4581.872820201069 msle_train: 5.837694423374635

rmse_test: 5052.774470843632 msle_test: 6.403340342863427
```

As we know, the target isn't linear and has a lot of features, it's worth experimenting with nonlinear models like the random forest model.

In [11]:
```
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)
```

Output:

```
rmse_train: 2901.328964468395 msle_train: 0.1922395819755205

rmse_test: 4678.589822380622 msle_test: 0.26151057307179615
```

Visualization of Results

To see how well the model works, a scatter plot of prediction against ground truth is generated. The model gives a reasonable prediction as the data points gather around the line y=x.

In [12]:
```
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':200, 'linewidths':0}, label=region)
plt.legend(loc=4,prop={'size': 12})
```

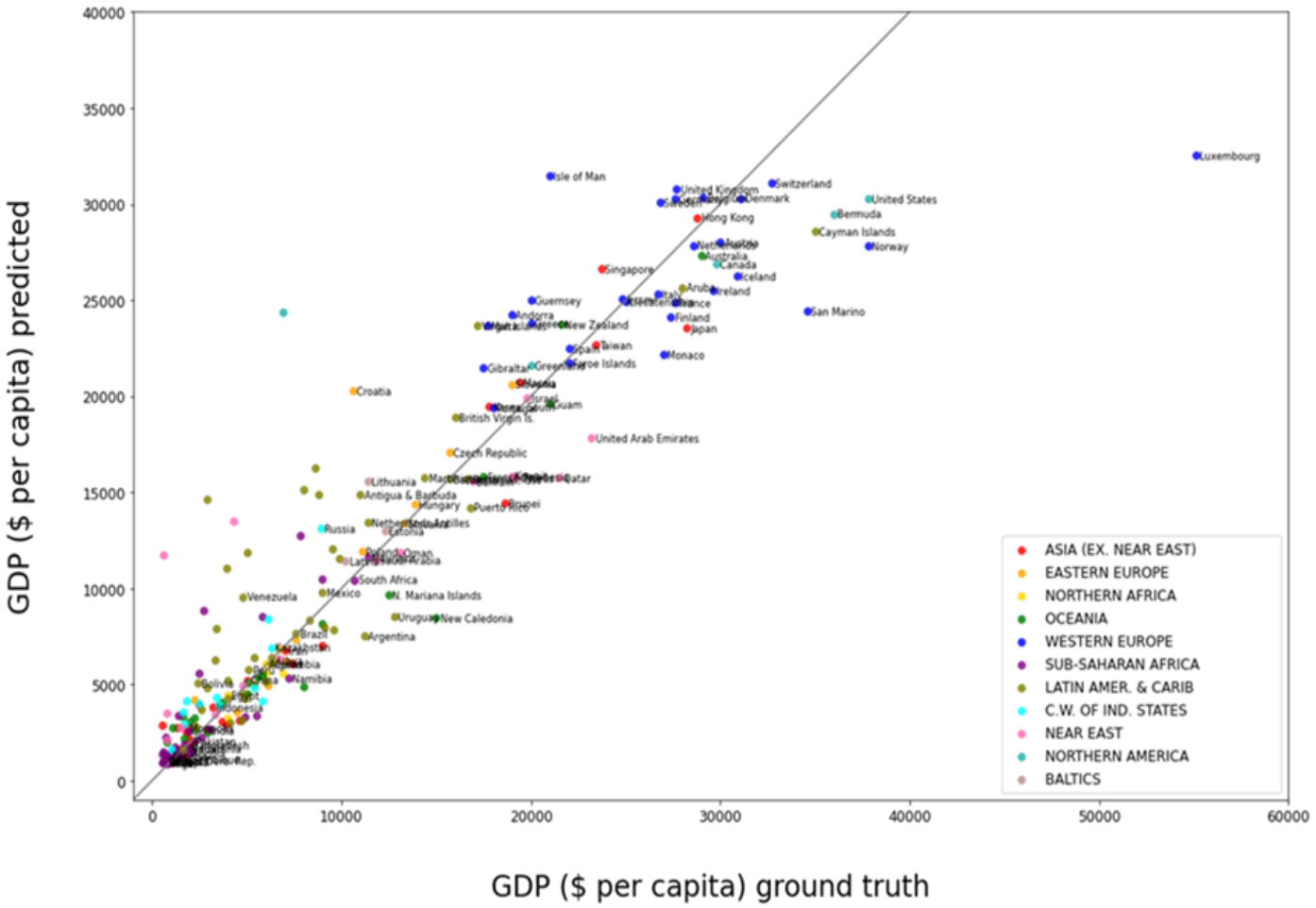
```
ax.set_xlabel('GDP ($ per capita) ground truth',labelpad=40)
ax.set_ylabel('GDP ($ per capita) predicted',labelpad=40)
ax.xaxis.label.set_fontsize(24)
ax.yaxis.label.set_fontsize(24)
ax.tick_params(labelsize=12)

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], size='small')
```

Output:



5. CONCLUSIONS

As technology is used in every aspect of our lives, the country's economy is no exception. Data science deals with massive amounts of data using modern tools and techniques and enables better decision making, predictive analysis, and pattern discovery. Using data science in GDP analysis enables us to know the factors that are affecting the GDP per capita of various countries. This helps to focus on the areas that help to foster economic development.