Forecasting Economic Prosperity: Leveraging machine learning for GDP per Capita Predicition

Mini Project documentation Submitted to:

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY , HYDERBAD

In partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING (DATA SCIENCE)

Submitted by

THEEGALA UJWALA	21UK1A67B6
SD OWEZ SHARIEF	22UK5A6715
JANGA PREETHI	21UK1A67B3
POOJARI SAI KRISHNA	21UK1A6781

under the guidance of

Mr. T. Sanath Kumar

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VAAGDEVI ENGINEERING COLLEGE

Affiliated to JNTUH, HYDERABAD BOLLIKUNTA, WARANGAL (T.S) – 506005

DEPARTMENT OF

COMPUTER SCIENCE AND ENGINEERING (DATA SCIENCE)

VAAGDEVI ENGINEERING COLLEGE(WARANGAL)

CERTIFICATE OF COMPLETION

INDUSTRY ORIENTED MINI PROJECT

This is to certify that the UG Project Phase-1 entitled " Leveraging Machine Learning for GDP Per Capita Prediction" is being submitted By THEEGALA UJWALA (21UK1A67B6), SD OWEZ SHARIEF (22UK5A6715), JANGA PREETHI (21UK1A67B3), POOJARI SAI KRISHNA (21UK1A6781) in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science & Engineering to Jawaharlal Nehru Technological University Hyderabad during the academic year 2023- 2024

PROJECT GUIDE	HOD
Mr. T. Sanath Kumar	Dr. K. Sharmila Reddy
(Assistant Professor)	(Professor)

External

ACKNOWLEDGEMENT

We wish to take this opportunity to express our sincere gratitude and deep sense of respect to our beloved **Dr. P. PRASAD RAO**, Principal, Vaagdevi Engineering College for making us available all the required assistance and for his support and inspiration to carry out this UG Project Phase-1 in the institute.

We extend our heartfelt thanks to **Dr. K. SHARMILA**, Head of the Department of CSD, Vaagdevi Engineering College for providing us necessary infrastructure and thereby giving us freedom to carry out the UG Project Phase-1

We express heartfelt thanks to Smart Bridge Educational Services Private Limited, for their constant supervision as well as for providing necessary information regarding the UG Project Phase-1 and for their support in completing the UG Project Phase-1.

We express heartfelt thanks to the guide, **T. SANATH KUMAR,** Assistant professor, Department of CSD for his constant support and giving necessary guidance for completion of this UG Project Phase-1.

Finally, we express our sincere thanks and gratitude to my family members, friends for their encouragement and outpouring their knowledge and experience throughout the thesis.

THEEGALA UJWALA **21UK1A67B6**

SD OWEZ SHARIEF **22UK5A6715**

JANGA PREETHI **21UK1A67B3**

POOJARI SAI KRISHNA **21UK1A6781**

ABSTRACT

This project aims to explore the application of machine learning techniques in forecasting GDP per capita, a crucial indicator of economic prosperity. By leveraging historical data encompassing various economic, social, and demographic factors, we seek to develop robust predictive models. The methodologies include data preprocessing, feature engineering, and model selection to optimize accuracy and reliability. Through this research, we endeavor to contribute to the field of economic forecasting, offering insights into the factors driving economic growth and providing policymakers and stakeholders with valuable tools for informed decision-making.

TABLE OF CONTENTS: -

1.INTRODUCTION

1.1 OVERVIEW

GDP per capita is a critical indicator of economic health and living standards, pivotal for decision-making in policy, economics, and business. Traditional forecasting models rely on historical trends and macroeconomic variables. However, leveraging machine learning and extensive datasets offers an opportunity to enhance accuracy and uncover new insights.

This project uses machine learning to predict GDP per capita by analyzing historical data across economic, social, and demographic factors influencing economic growth. Through rigorous data preprocessing, feature engineering, and model selection, our aim is to build robust predictive models. These models not only forecast GDP per capita but also provide insights into the drivers of economic prosperity.

This research aims to improve forecasting methods by integrating a broader range of variables and employing advanced modeling techniques. The results will inform policymakers, aiding in the formulation of strategies for sustainable economic growth and improved living standards.

1.2. PURPOSE

The purpose of this project is twofold:

1. Predictive Accuracy: Utilize machine learning techniques to improve the accuracy of GDP per capita forecasts beyond traditional economic models. By incorporating a diverse set of economic, social, and demographic variables, the aim is to capture more nuanced relationships and dynamics affecting economic growth.

2. Insight Generation: Uncover actionable insights into the drivers of economic prosperity. Through detailed analysis of model outputs and feature importance, identify which factors significantly influence GDP per capita variations. This information can inform policymakers, economists, and businesses in making informed decisions and shaping effective strategies.

By achieving these purposes, this research seeks to contribute to the advancement of economic forecasting methodologies and provide valuable tools for stakeholders navigating economic landscapes.

2.LITERATURE SURVEY

2.1 EXISTING PROBLEM

Despite advancements in economic forecasting, several challenges persist in accurately predicting GDP per capita:

1.Data Quality and Availability: Economic data, especially from developing countries or regions, may be incomplete, inconsistent, or outdated. This can hinder the construction of reliable forecasting models and lead to biased or unreliable predictions.

2. Complexity of Economic Systems: Economic growth is influenced by a multitude of interconnected factors, including political stability, technological advancements, global trade dynamics, and environmental conditions. Modeling these complexities accurately requires sophisticated methodologies that traditional models may struggle to accommodate.

3.Non-linear Relationships: Economic variables often exhibit non-linear relationships with GDP per capita, making it challenging to capture these nuances using linear regression models or simplistic approaches. Machine learning techniques offer potential solutions by allowing for the exploration of non-linear patterns in data

4. Model Interpretability: While machine learning models can offer high predictive accuracy, they often sacrifice interpretability. Understanding how and why specific variables influence GDP per capita predictions is crucial for policymakers seeking actionable insights.

5. External Shocks and Uncertainties: Economic forecasts are vulnerable to unexpected events such as financial crises, natural disasters, or geopolitical tensions. These external shocks can disrupt traditional forecasting models and necessitate adaptable and resilient forecasting frameworks.

2.2 PROPOSED SOLLUTION

1. Enhanced Data Quality: Improve data collection and quality assurance processes, collaborating with international organizations and governments.

2.Advanced Modeling Techniques: Implement advanced machine learning algorithms and techniques like ensemble methods and deep learning to capture complex economic relationships.

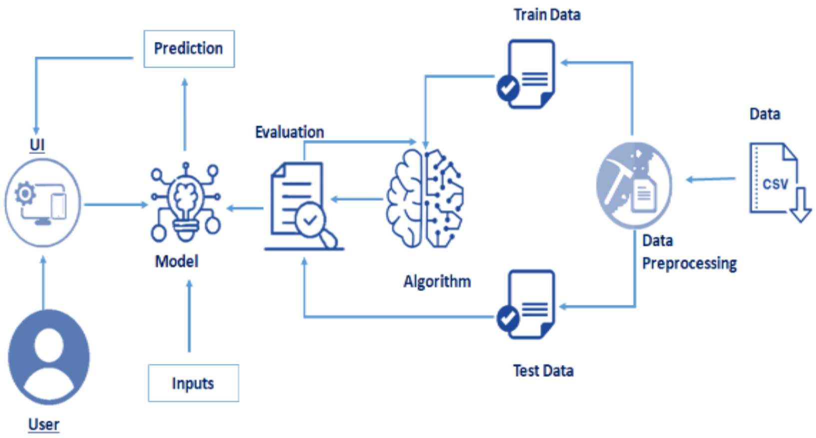
3.Integration of Diverse Variables: Expand variables beyond traditional economic indicators to include social, demographic, and environmental factors, using advanced analytics to analyze interactions.

4.Interpretability and Transparency: Develop hybrid models for balanced accuracy and interpretability, employing model-agnostic methods to explain predictions.

5.Scenario Analysis and Resilience: Incorporate scenario analysis and adaptive frameworks to assess and adjust forecasts in response to external shocks and uncertainties.

3.THEORITICAL ANALYSIS

3.1. BLOCK DIAGRAM



3.2 SOFTWARE DESIGNING

The following is the Software required to complete this project:

➢ **Google Colab:** Google Colab will serve as the development and execution environment for your predictive modeling, data preprocessing, and model training tasks. It provides a cloud-based Jupyter Notebook environment with access to Python libraries and hardware acceleration.

➢ **Dataset (CSV File):** The dataset in CSV format is essential for training and testing your predictive model. It should include historical air quality data, weather information, pollutant levels, and other relevant features.

➢ **Data Preprocessing Tools:** Python libraries like NumPy, Pandas, and Scikit-learn will be used to preprocess the dataset. This includes handling missing data, feature scaling and data cleaning.

➢ **Feature Selection/Drop:** Feature selection or dropping unnecessary features from the dataset can be done using Scikit-learn or custom Python code to enhance the model's efficiency.

➢ **Model Training Tools:** Machine learning libraries such as Scikit-learn, TensorFlow, or PyTorch will be used to develop, train, and fine-tune the predictive model. Regression or classification models can be considered, depending on the nature of the AQI prediction task.

➢ **Model Accuracy Evaluation:** After model training, accuracy and performance evaluation tools, such as Scikit-learn metrics or custom validation scripts, will assess the model's predictive capabilities. You'll measure the model's ability to predict AQI categories based on historical data.

➢ **UI Based on Flask Environment:** Flask, a Python web framework, will be used to develop the user interface (UI) for the system. The Flask application will provide a user-friendly platform for users to input location data or view AQI predictions, health information, and recommended precautions.

➢ Google Colab will be the central hub for model development and training, while Flask will facilitate user interaction and data presentation. The dataset, along with data preprocessing, will ensure the quality of the training data, and feature selection will optimize the model. Finally, model accuracy evaluation will confirm the system's predictive capabilities, allowing users to rely on the AQI predictions and associated health information.

To complete this project, you must required the following software's, concepts and packages
Visual studio

Python packages:

- Open Jupyter in vs code then install packages
- Type "pip install numpy" and click enter
- Type "pip install pandas" and click enter.
- Type "pip install seaborn" and click enter.
- Type "pip install matplotlib" and click enter.
- Type "pip install pickle" and click enter.
- Type "pip install Flask" and click enter.

4.EXPERIMENTAL INVESTIGATIONS

The project involves comprehensive experimental investigations to assess the efficacy of machine learning models in predicting GDP per capita. By employing diverse datasets encompassing economic, social, and demographic variables, the experiments aim to evaluate model performance across different methodologies. These investigations include rigorous analysis of predictive accuracy, model interpretability, and robustness in capturing complex economic relationships. Furthermore, the research delves into conducting sensitivity analyses to understand the influence of key variables on GDP per capita predictions, thereby uncovering critical insights into the drivers of economic growth.

Project Flow:

- User interacts with the UI to enter the input.
- Entered input is analyzed by the model which is integrated.
- Once model analyses the input the prediction is showcased on the UI

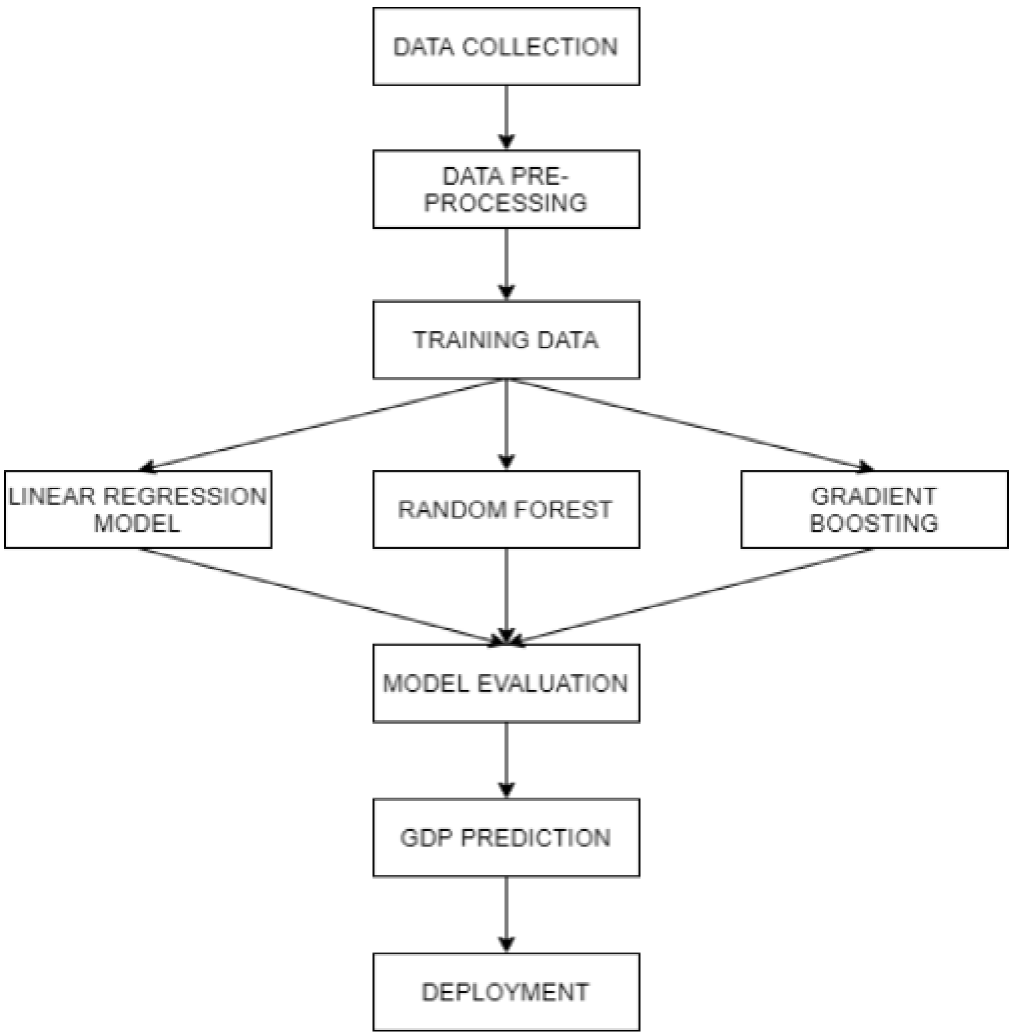
To accomplish this, we have to complete all the activities listed below,

- **Data collection**
  - Collect the dataset or create the dataset
- **Visualizing and analyzing data**
  - Univariate analysis
  - Bivariate analysis
  - Multivariate analysis
  - Descriptive analysis
- **Data pre-processing**
  - Checking for null values
  - Handling outlier

- Handling categorical data
- Splitting data into train and test

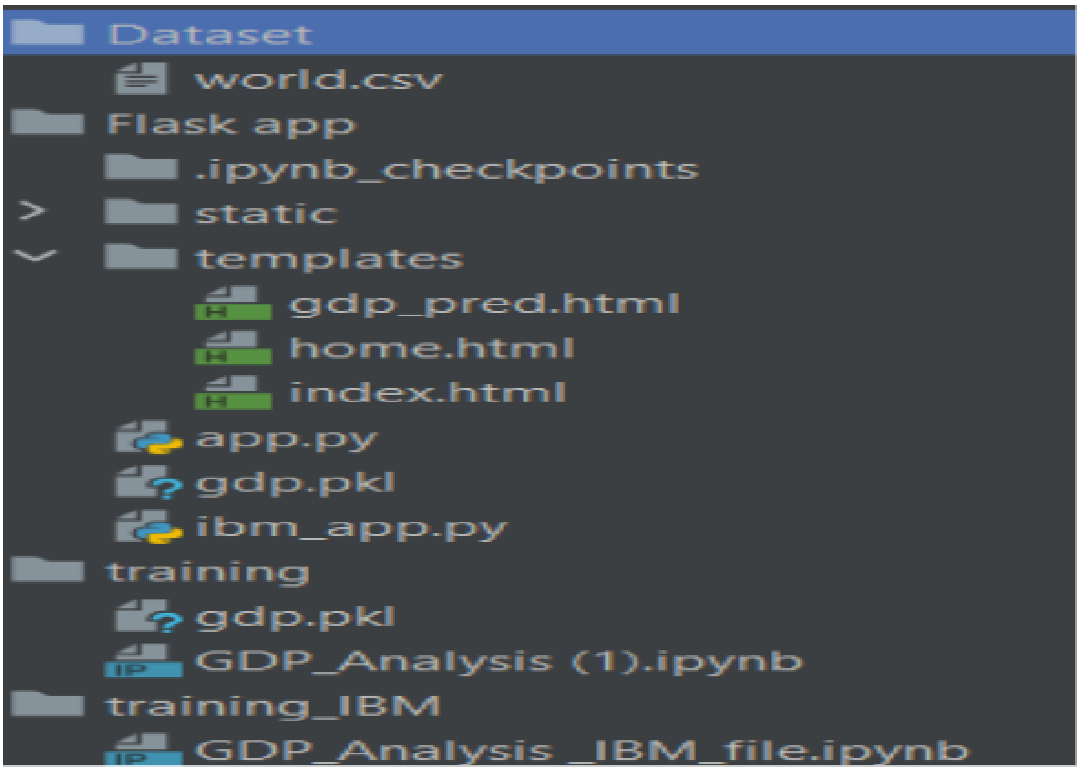
- **Model building**
  - Import the model building libraries
  - Initializing the model
  - Training and testing the model
  - Evaluating performance of model
  - Save the model
- **Application Building**
  - Create an HTML file and Build python code

5. FLOWCHART



Project Structure

Create a Project folder that contains files as shown below



We are building a Flask application that needs HTML pages stored in the templates folder and a Python script app.py for scripting.
For IBM deployment ibm_app.py file is used.
Model.pkl is our saved model. Further, we will use this model for flask integration.
The training folder contains model training files and the training ibm folder contains IBM model training files.

6.ADVANTAGES:

Enhanced Accuracy: Machine learning models capture complex relationships for more accurate GDP per capita predictions.

Flexibility and Scalability: Models adapt to diverse data and can scale for multiple countries, facilitating broader economic analysis.

Automation and Efficiency: Automated forecasting reduces manual effort and enhances efficiency in economic prediction.

Insightful Analysis: Provides insights into key economic drivers, aiding policymakers and researchers in decision-making.

Continuous Improvement: Iterative model refinement ensures predictions align with evolving economic conditions.

Interdisciplinary Collaboration: Bridges economics and data science, fostering innovative approaches to economic forecasting.

7.DISADVANTAGES:

Complexity and Interpretability: Machine learning models can be complex, making it challenging to interpret results, especially for non-experts.

Data Dependency: Effective models require large, high-quality datasets, which may not be available or reliable in all regions.

Overfitting Risk: Models may overfit to historical data, leading to inaccurate predictions when faced with new economic scenarios.

8.APPLICATIONS

Forecasting: Machine learning models can predict future GDP per capita trends based on historical data and economic indicators.

Policy Analysis: They provide insights into the impact of policy changes on economic growth and per capita income.

Risk Assessment: Models assess economic risks and vulnerabilities that could affect GDP per capita, helping to inform risk management strategies.

9.CONCLUSION

In this project, we used countries_of_the_world dataset to build a GDP predictor. 3 different learning regressors Linear Regression, Random Forest, and Gradiant Boosting were tested, and we have acheived the best prediction performance using Gradient Boosting, followed by Random Forest, and then Linear Regression . The best prediction performance was acheived using Gradient Boosting regressor, using all features in the dataset, and resulted in the following metrics: Mean Absolute Error (MAE): 2280.46 Root mean squared error (RMSE): 3413.63 R-squared Score (R2_Score): 0.85

10.FUTURE SCOPE

Advanced Modeling Techniques: Future advancements may focus on developing more sophisticated machine learning models that can better capture complex economic relationships and non-linear dynamics.

Integration of Big Data: Incorporating larger and more diverse datasets, including real-time economic indicators and alternative data sources, could enhance prediction accuracy and timeliness.

Interdisciplinary Approaches: Further integration of economics, data science, and domain expertise can lead to innovative approaches for economic forecasting and policy analysis.

Ethical and Regulatory Considerations: Future research will likely address ethical issues such as fairness, transparency, and privacy concerns associated with the use of predictive models in economic decision-making.

11. Model Building

Data Collection

ML depends heavily on data, it is the most crucial aspect that makes algorithm training possible. So this section allows you to download the required dataset

Download Dataset

You can collect datasets from different open sources like kaggle.com, data.gov, UCI machine learning repository, etc.

Download the dataset from the link below.

Link

https://www.kaggle.com/fernandol/countries-of-the-world

Visualizing And Analysing The Data

1.Import Data and Data Cleaning

For collecting the data of the countries of the world for gdp prediction we have used Kaggle.com. Using this website we have downloaded the csv file of the countries of the world. To read the data from the file we have used pandas data reader

```
In [1]: import numpy as np
        import pandas as pd
        import seaborn as sns
        import matplotlib.pyplot as plt
        %matplotlib inline
        from sklearn.model_selection import train_test_split
        from sklearn.preprocessing import StandardScaler
        from sklearn import metrics
        from sklearn.model_selection import GridSearchCV
        from sklearn.linear_model import LinearRegression
        from sklearn.svm import SVR
        from sklearn.ensemble import RandomForestRegressor
        from sklearn.ensemble import GradientBoostingRegressor
        from sklearn.model_selection import cross_val_score
```

```
In [2]: data = pd.read_csv('countries of the world.csv')
```

```
In [3]: data.head(3)
```

Out[3]:

	Country	Region	Population	Area (sq. mi.)	Pop. Density (per sq. mi.)	Coastline (coast/area ratio)	Net migration	Inf morta (1 birt
0	Afghanistan	ASIA (EX. NEAR EAST)	31056997	647500	48,0	0,00	23,06	16
1	Albania	EASTERN EUROPE	3581655	28748	124,6	1,26	-4,93	2
2	Algeria	NORTHERN AFRICA	32930091	2381740	13,8	0,04	-0,39	

As we are using EDA here which means:

Exploratory Data Analysis (EDA) is an approach to analyze the data using visual techniques. It is used to discover trends, patterns, or to check assumptions with the help of statistical summary and graphical representations. For this we will perform following tasks on our dataset: • Getting insights about the dataset • Handling missing values • Data Visualization Now we will see how our data attributes looks like such as the name of the attributes and its data type:

```
data.service = data.service.astype(str)
data.service = data.service.str.replace(",",".").astype(float)
```

In [7]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 227 entries, 0 to 226
Data columns (total 20 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   country             227 non-null    category
 1   region              227 non-null    category
 2   population          227 non-null    int64
 3   area                227 non-null    int64
 4   density             227 non-null    float64
 5   coastline_area_ratio 227 non-null   float64
 6   net_migration       224 non-null    float64
 7   infant_mortality    224 non-null    float64
 8   gdp_per_capita      226 non-null    float64
 9   literacy            209 non-null    float64
 10  phones              223 non-null    float64
 11  arable              225 non-null    float64
 12  crops               225 non-null    float64
 13  other               225 non-null    float64
 14  climate             205 non-null    float64
 15  birthrate           224 non-null    float64
 16  deathrate           223 non-null    float64
 17  agriculture         212 non-null    float64
 18  industry            211 non-null    float64
 19  service             212 non-null    float64
dtypes: category(2), float64(16), int64(2)
memory usage: 43.0 KB
```

In [6]:
```
data.country = data.country.astype('category')

data.region = data.region.astype('category')

data.density = data.density.astype(str)
data.density = data.density.str.replace(",",".").astype(float)

data.coastline_area_ratio = data.coastline_area_ratio.astype(str)
data.coastline_area_ratio = data.coastline_area_ratio.str.replace(",",".").astype(float)

data.net_migration = data.net_migration.astype(str)
data.net_migration = data.net_migration.str.replace(",",".").astype(float)

data.infant_mortality = data.infant_mortality.astype(str)
data.infant_mortality = data.infant_mortality.str.replace(",",".").astype(float)

data.literacy = data.literacy.astype(str)
data.literacy = data.literacy.str.replace(",",".").astype(float)

data.phones = data.phones.astype(str)
data.phones = data.phones.str.replace(",",".").astype(float)

data.arable = data.arable.astype(str)
data.arable = data.arable.str.replace(",",".").astype(float)

data.crops = data.crops.astype(str)
data.crops = data.crops.str.replace(",",".").astype(float)

data.other = data.other.astype(str)
data.other = data.other.str.replace(",",".").astype(float)

data.climate = data.climate.astype(str)
data.climate = data.climate.str.replace(",",".").astype(float)
```

In [8]:
```
data.describe()
```

Out[8]:

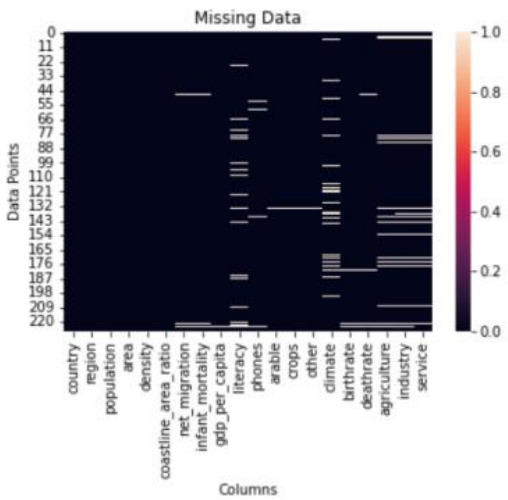
	population	area	density	coastline_area_ratio	net_migration	infant_mortality	gdp_per_capita	literacy	phones	arable
count	2.270000e+02	2.270000e+02	227.000000	227.000000	224.000000	224.000000	226.000000	209.000000	223.000000	225.000000
mean	2.874028e+07	5.982270e+05	379.047137	21.165330	0.038125	35.506964	9689.823009	82.838278	236.061435	13.797111
std	1.178913e+08	1.790282e+06	1660.185825	72.286863	4.889269	35.389899	10049.138513	19.722173	227.991829	13.040402
min	7.026000e+03	2.000000e+00	0.000000	0.000000	-20.990000	2.290000	500.000000	17.600000	0.200000	0.000000
25%	4.376240e+05	4.647500e+03	29.150000	0.100000	-0.927500	8.150000	1900.000000	70.600000	37.800000	3.220000
50%	4.786994e+06	8.660000e+04	78.800000	0.730000	0.000000	21.000000	5550.000000	92.500000	176.200000	10.420000
75%	1.749777e+07	4.418110e+05	190.150000	10.345000	0.997500	55.705000	15700.000000	98.000000	389.650000	20.000000
max	1.313974e+09	1.707520e+07	16271.500000	870.660000	23.060000	191.190000	55100.000000	100.000000	1035.600000	62.110000

In [9]:
```
print(data.isnull().sum())
```

```
country                 0
region                  0
population              0
area                    0
density                 0
coastline_area_ratio    0
net_migration           3
infant_mortality        3
gdp_per_capita          1
literacy               18
phones                  4
arable                  2
crops                   2
other                   2
climate                22
birthrate               3
deathrate               4
agriculture            15
industry               16
service                15
dtype: int64
```

We will see the missing data values using heat map:

In [11]:
```
sns.heatmap(data.isnull()).set(title = 'Missing Data', xlabel = 'Columns', ylabel ='Data Points');
```



```
In [20]: data['net_migration'].fillna(0, inplace=True)
data['infant_mortality'].fillna(0, inplace=True)
data['gdp_per_capita'].fillna(2500, inplace=True)
data['literacy'].fillna(data.groupby('region')['literacy'].transform('mean'), inplace= True)
data['phones'].fillna(data.groupby('region')['phones'].transform('mean'), inplace= True)
data['arable'].fillna(0, inplace=True)
data['crops'].fillna(0, inplace=True)
data['other'].fillna(0, inplace=True)
data['climate'].fillna(0, inplace=True)
data['birthrate'].fillna(data.groupby('region')['birthrate'].transform('mean'), inplace= True)
data['deathrate'].fillna(data.groupby('region')['deathrate'].transform('mean'), inplace= True)
data['agriculture'].fillna(0.17, inplace=True)
data['service'].fillna(0.8, inplace=True)
data['industry'].fillna((1 - data['agriculture'] - data['service']), inplace= True)
```

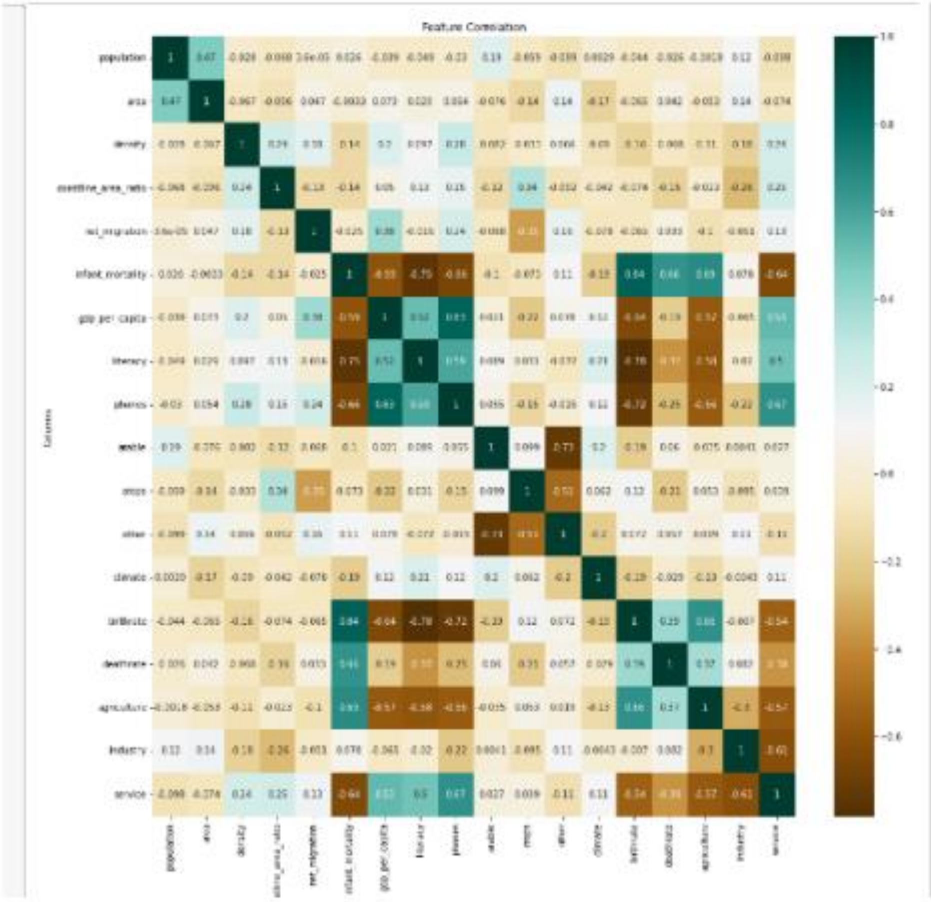
Now check if any null value exist:

```
In [21]: print(data.isnull().sum())

country                0
region                 0
population             0
area                   0
density                0
coastline_area_ratio   0
net_migration          0
infant_mortality       0
gdp_per_capita         0
literacy               0
phones                 0
arable                 0
crops                  0
other                  0
climate                0
birthrate              0
deathrate              0
agriculture            0
industry               0
service                0
dtype: int64
```

Now our dataset contains no Null values

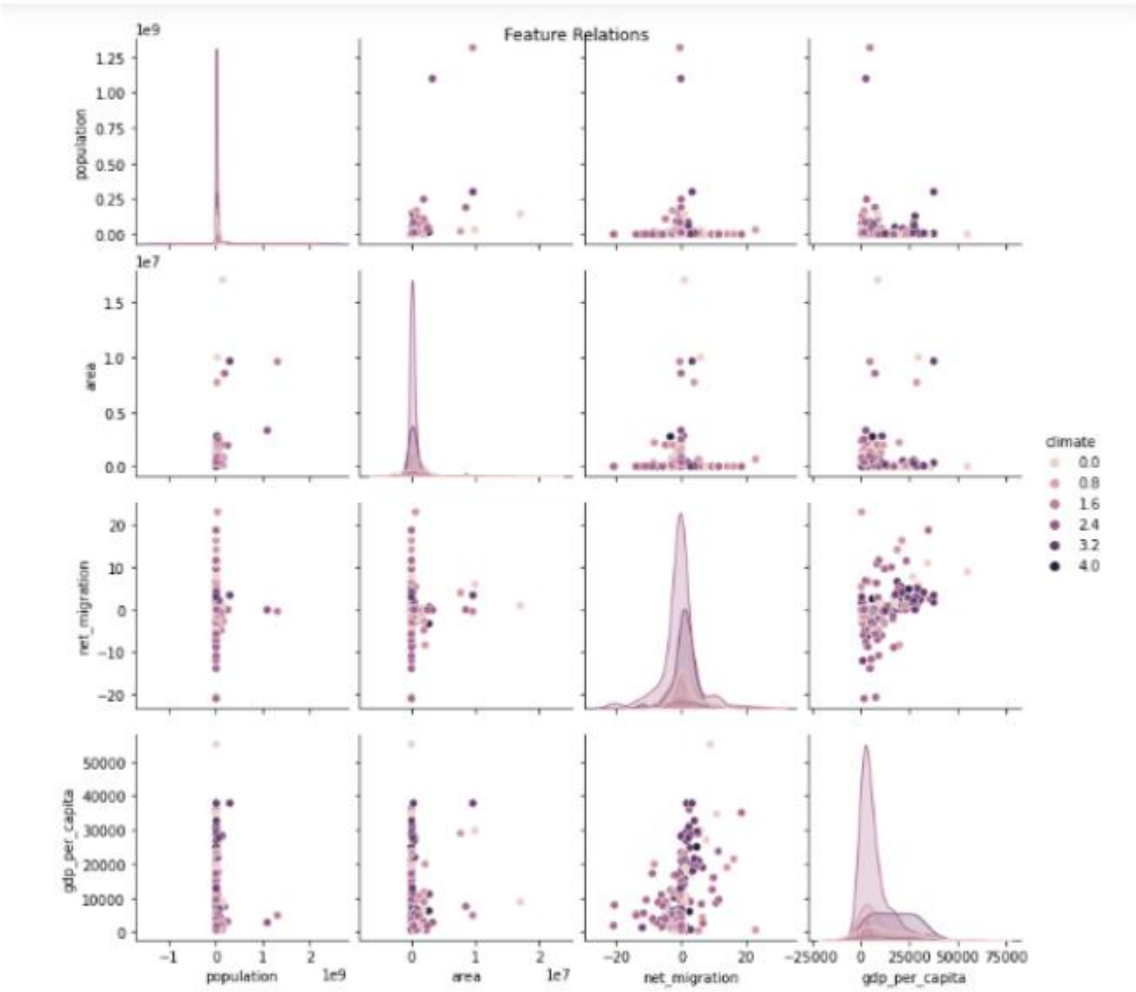
```
In [22]: fig, ax = plt.subplots(figsize=(16,16))
sns.heatmap(data.corr(), annot=True, ax=ax, cmap='BrBG').set(
    title = 'Feature Correlation', xlabel = 'Columns', ylabel = 'Columns')
plt.show()
```

Our correlation heatmap looks like this :



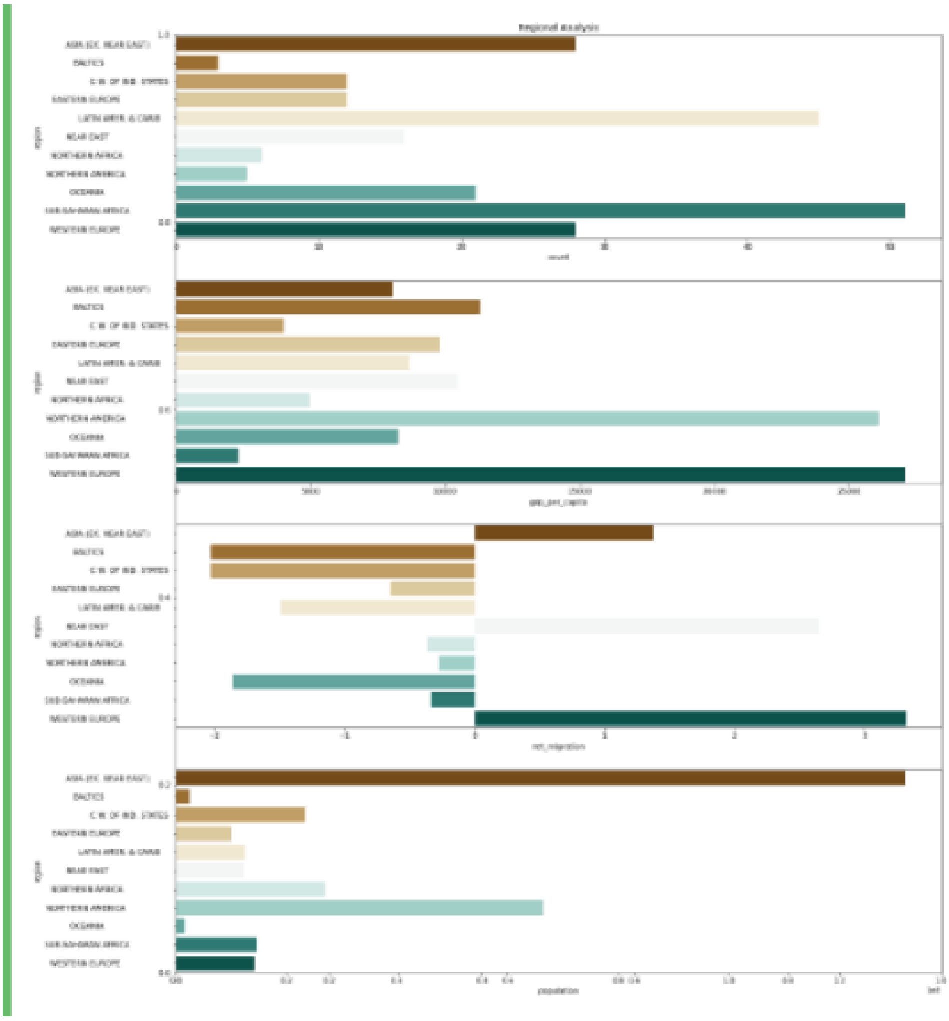
Lets now show some correlations among a few of features

```
In [23]: g = sns.pairplot(data[['population', 'area', 'net_migration', 'gdp_per_capita', 'climate']], hue='climate')
         g.fig.suptitle('Feature Relations')
         plt.show()
```



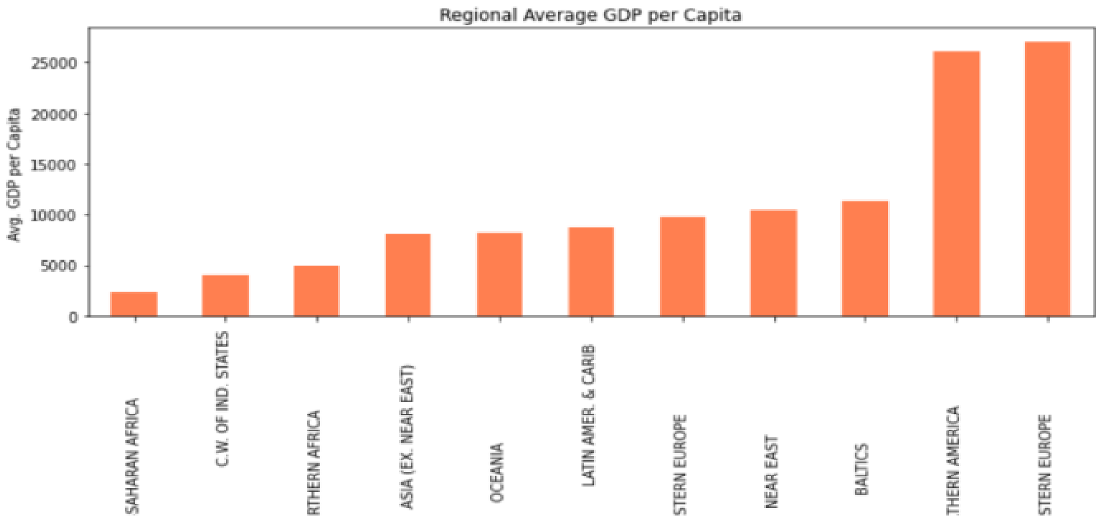
Now we do regional analysis

```
In [24]: fig = plt.figure(figsize=(18, 24))
         plt.title('Regional Analysis')
         ax1 = fig.add_subplot(4, 1, 1)
         ax2 = fig.add_subplot(4, 1, 2)
         ax3 = fig.add_subplot(4, 1, 3)
         ax4 = fig.add_subplot(4, 1, 4)
         sns.countplot(data= data, y= 'region', ax= ax1, palette='BrBG')
         sns.barplot(data= data, y= 'region', x= 'gdp_per_capita', ax= ax2, palette='BrBG', ci= None)
         sns.barplot(data= data, y= 'region', x= 'net_migration', ax= ax3, palette='BrBG', ci= None)
         sns.barplot(data= data, y= 'region', x= 'population', ax= ax4, palette='BrBG', ci= None)
         plt.show()
```



Now we will perform GDP analysis

```
In [25]: fig = plt.figure(figsize=(12, 4))
         data.groupby('region')['gdp_per_capita'].mean().sort_values().plot(kind='bar', color='coral')
         plt.title('Regional Average GDP per Capita')
         plt.xlabel("Region")
         plt.ylabel('Avg. GDP per Capita')
         plt.show()
```



Preprocess the data train and test

In [29]:
```
data_final = pd.concat([data,pd.get_dummies(data['region'], prefix='region')], axis=1).drop(['region'],axis=1)
print(data_final.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 227 entries, 0 to 226
Data columns (total 30 columns):
 #   Column                           Non-Null Count  Dtype
---  ------                           --------------  -----
 0   country                          227 non-null    category
 1   population                       227 non-null    int64
 2   area                             227 non-null    int64
 3   density                          227 non-null    float64
 4   coastline_area_ratio             227 non-null    float64
 5   net_migration                    227 non-null    float64
 6   infant_mortality                 227 non-null    float64
 7   gdp_per_capita                   227 non-null    float64
 8   literacy                         227 non-null    float64
 9   phones                           227 non-null    float64
 10  arable                           227 non-null    float64
 11  crops                            227 non-null    float64
 12  other                            227 non-null    float64
 13  climate                          227 non-null    float64
 14  birthrate                        227 non-null    float64
 15  deathrate                        227 non-null    float64
 16  agriculture                      227 non-null    float64
 17  industry                         227 non-null    float64
 18  service                          227 non-null    float64
 19  region_ASIA (EX. NEAR EAST)      227 non-null    uint8
 20  region_BALTICS                   227 non-null    uint8
 21  region_C.W. OF IND. STATES       227 non-null    uint8
 22  region_EASTERN EUROPE            227 non-null    uint8
 23  region_LATIN AMER. & CARIB       227 non-null    uint8
 24  region_NEAR EAST                 227 non-null    uint8
 25  region_NORTHERN AFRICA           227 non-null    uint8
```

n [32]:
```
y = data_final['gdp_per_capita']
X = data_final.drop(['gdp_per_capita','country'], axis=1)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=101)
```

```
In [34]: sc_X = StandardScaler()

X2_train = sc_X.fit_transform(X_train)
X2_test = sc_X.fit_transform(X_test)
y2_train = y_train
y2_test = y_test
```

```
In [35]: y3 = y
X3 = data_final.drop(['gdp_per_capita','country','population', 'area', 'coastline_area_ratio', 'arable',
                      'crops', 'other', 'climate', 'deathrate', 'industry'], axis=1)

X3_train, X3_test, y3_train, y3_test = train_test_split(X3, y3, test_size=0.2, random_state=101)
```

```
In [36]: sc_X4 = StandardScaler()

X4_train = sc_X4.fit_transform(X3_train)
X4_test = sc_X4.fit_transform(X3_test)
y4_train = y3_train
y4_test = y3_test
```

Now Our Data is ready for applying machine learning algorithms :

1. Linear Regression Model Training

Model Training

```
In [37]: lm1 = LinearRegression()
lm1.fit(X_train,y_train)

lm2 = LinearRegression()
lm2.fit(X2_train,y2_train)

lm3 = LinearRegression()
lm3.fit(X3_train,y3_train)

lm4 = LinearRegression()
lm4.fit(X4_train,y4_train)

Out[37]: LinearRegression()
```

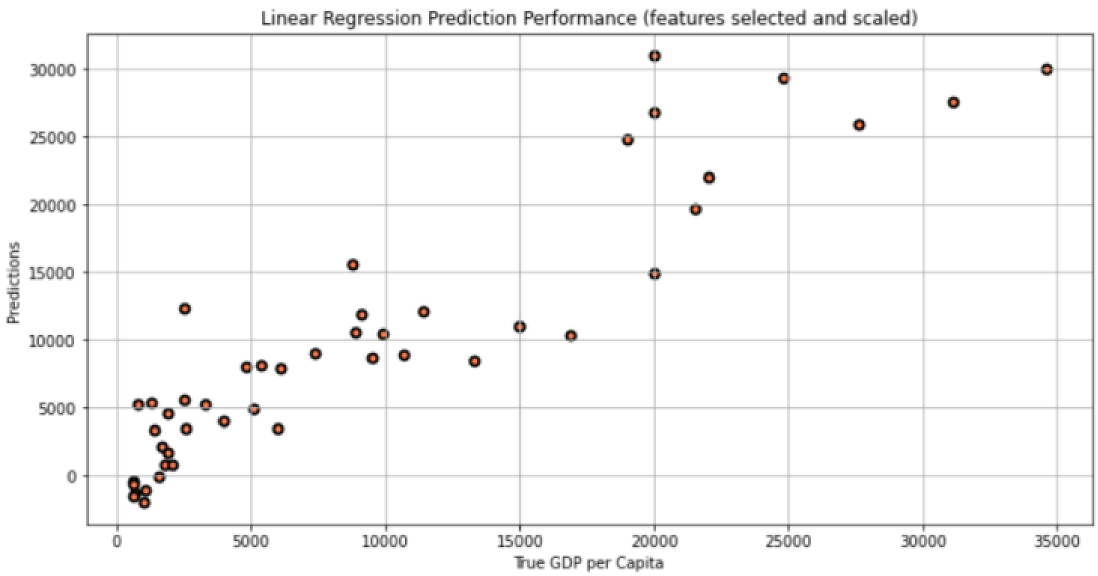
Predictions

```
In [38]: lm1_pred = lm1.predict(X_test)
         lm2_pred = lm2.predict(X2_test)
         lm3_pred = lm3.predict(X3_test)
         lm4_pred = lm4.predict(X4_test)
```

Evaluation

```
In [39]: print('Linear Regression Performance:')

         print('\nall features, No scaling:')
         print('MAE:', metrics.mean_absolute_error(y_test, lm1_pred))
         print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, lm1_pred)))
         print('R2_Score: ', metrics.r2_score(y_test, lm1_pred))

         print('\nall features, with scaling:')
         print('MAE:', metrics.mean_absolute_error(y2_test, lm2_pred))
         print('RMSE:', np.sqrt(metrics.mean_squared_error(y2_test, lm2_pred)))
         print('R2_Score: ', metrics.r2_score(y2_test, lm2_pred))

         print('\nselected features, No scaling:')
         print('MAE:', metrics.mean_absolute_error(y3_test, lm3_pred))
         print('RMSE:', np.sqrt(metrics.mean_squared_error(y3_test, lm3_pred)))
         print('R2_Score: ', metrics.r2_score(y3_test, lm3_pred))

         print('\nselected features, with scaling:')
         print('MAE:', metrics.mean_absolute_error(y4_test, lm4_pred))
         print('RMSE:', np.sqrt(metrics.mean_squared_error(y4_test, lm4_pred)))
         print('R2_Score: ', metrics.r2_score(y4_test, lm4_pred))

         fig = plt.figure(figsize=(12, 6))
         plt.scatter(y4_test,lm4_pred,color='coral', linewidths=2, edgecolors='k')
         plt.xlabel('True GDP per Capita')
         plt.ylabel('Predictions')
         plt.title('Linear Regression Prediction Performance (features selected and scaled)')
         plt.grid()
         plt.show()
```

```
Linear Regression Performance:

all features, No scaling:
MAE: 330350.8586600643
RMSE: 1570337.5456386511
R2_Score:  -29843.120383337

all features, with scaling:
MAE: 569019.4687587288
RMSE: 1283170.8219650008
R2_Score:  -19925.99011845563

selected features, No scaling:
MAE: 2965.9357229398815
RMSE: 4088.7945802479585
R2_Score:  0.7976685756858989

selected features, with scaling:
MAE: 2879.5213243944386
RMSE: 3756.436588502965
R2_Score:  0.8292247702712091
```



2.RANDOM FOREST

Model Training

```
In [40]: rf1 = RandomForestRegressor(random_state=101, n_estimators=200)
         rf3 = RandomForestRegressor(random_state=101, n_estimators=200)

         rf1.fit(X_train, y_train)
         rf3.fit(X3_train, y3_train)

Out[40]: RandomForestRegressor(n_estimators=200, random_state=101)
```

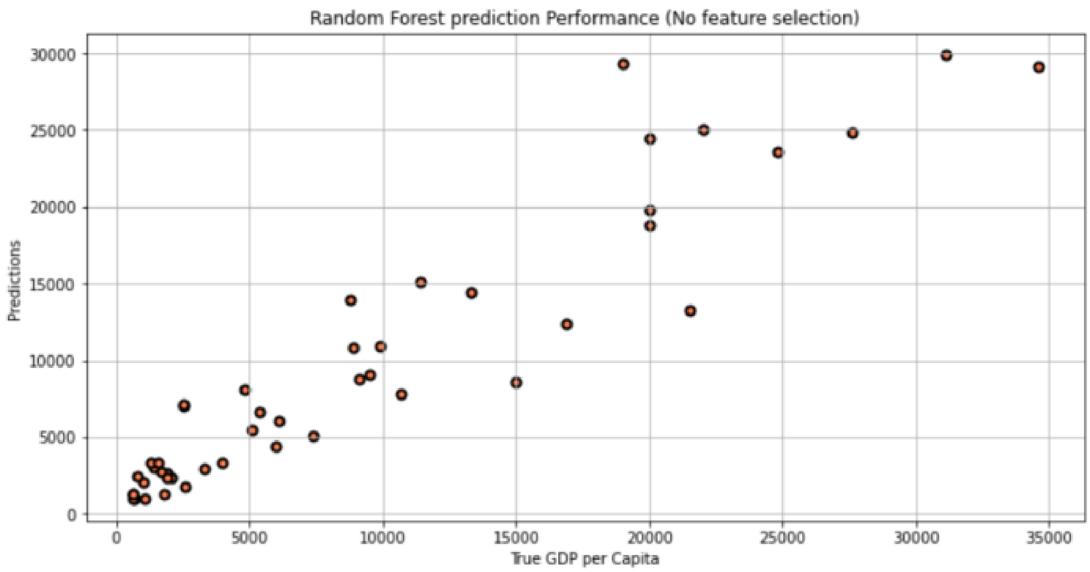
Predictions

```
In [41]: rf1_pred = rf1.predict(X_test)
         rf3_pred = rf3.predict(X3_test)
```

Evaluation

```
In [42]: print('Random Forest Performance:')

         print('\nall features, No scaling:')
         print('MAE:', metrics.mean_absolute_error(y_test, rf1_pred))
         print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, rf1_pred)))
         print('R2_Score: ', metrics.r2_score(y_test, rf1_pred))

         print('\nselected features, No scaling:')
         print('MAE:', metrics.mean_absolute_error(y3_test, rf3_pred))
         print('RMSE:', np.sqrt(metrics.mean_squared_error(y3_test, rf3_pred)))
         print('R2_Score: ', metrics.r2_score(y3_test, rf3_pred))

         fig = plt.figure(figsize=(12, 6))
         plt.scatter(y_test,rf1_pred,color='coral', linewidths=2, edgecolors='k')
         plt.xlabel('True GDP per Capita')
         plt.ylabel('Predictions')
         plt.title('Random Forest prediction Performance (No feature selection)')
         plt.grid()
         plt.show()
```

```
Random Forest Performance:

all features, No scaling:
MAE: 2142.1304347826085
RMSE: 3097.1944738255706
R2_Score:  0.8839060185534444

selected features, No scaling:
MAE: 2416.0652173913045
RMSE: 3533.590316058036
R2_Score:  0.8488858452472634
```



3. Gradient Boosting Regressor

Model Training

```
In [37]: gbm1 = GradientBoostingRegressor(learning_rate=0.1, n_estimators=100, min_samples_split=2, min_samples_leaf=1, max_depth=3,
                                          subsample=1.0, max_features= None, random_state=101)
         gbm3 = GradientBoostingRegressor(learning_rate=0.1, n_estimators=100, min_samples_split=2, min_samples_leaf=1, max_depth=3,
                                          subsample=1.0, max_features= None, random_state=101)

         gbm1.fit(X_train, y_train)
         gbm3.fit(X3_train, y3_train)

Out[37]: GradientBoostingRegressor(random_state=101)
```

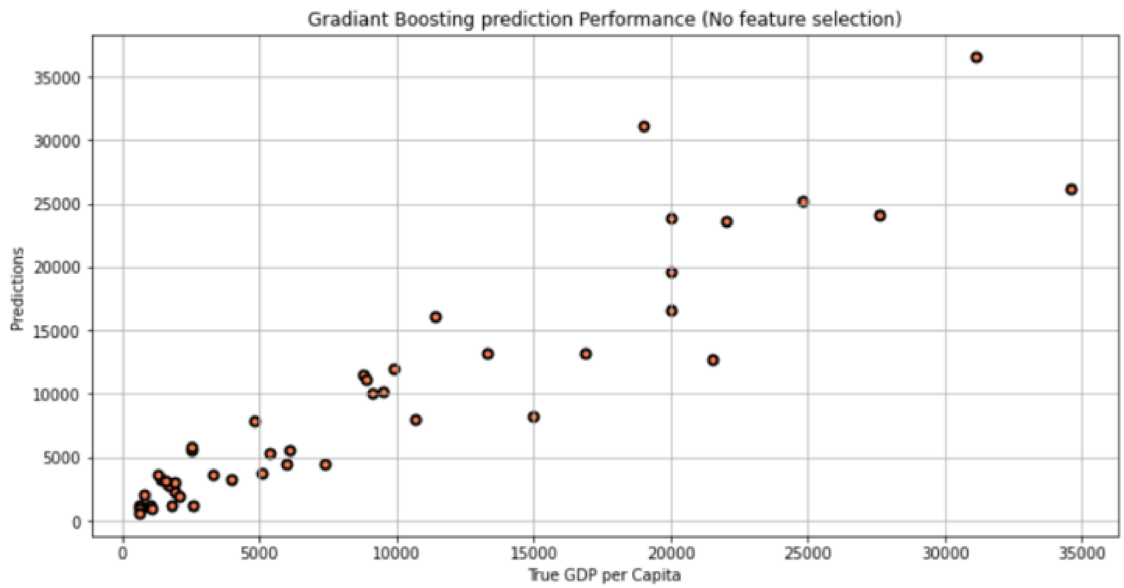
Prediction

```
In [38]: gbm1_pred = gbm1.predict(X_test)
         gbm3_pred = gbm3.predict(X3_test)
```

Evaluation

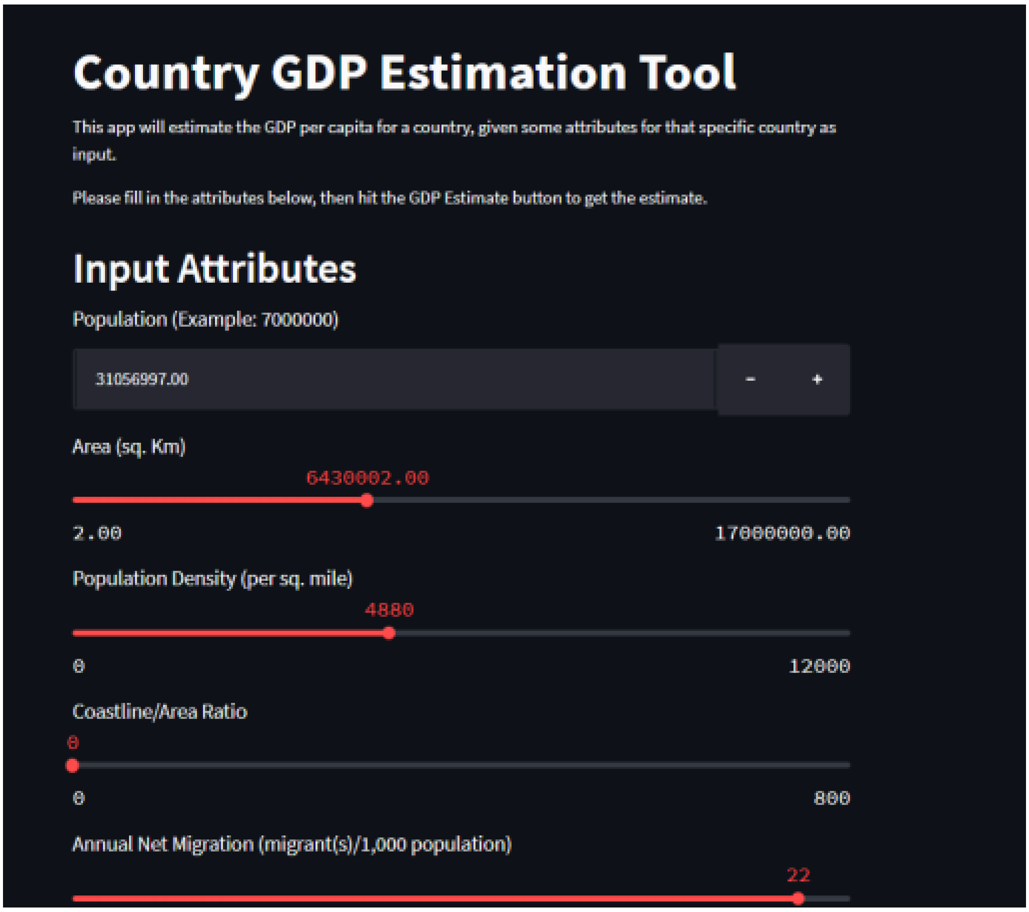
```
In [39]: print('Gradiant Boosting Performance:')

         print('\nall features, No scaling:')
         print('MAE:', metrics.mean_absolute_error(y_test, gbm1_pred))
         print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, gbm1_pred)))
         print('R2_Score: ', metrics.r2_score(y_test, gbm1_pred))

         print('\nselected features, No scaling:')
         print('MAE:', metrics.mean_absolute_error(y3_test, gbm3_pred))
         print('RMSE:', np.sqrt(metrics.mean_squared_error(y3_test, gbm3_pred)))
         print('R2_Score: ', metrics.r2_score(y3_test, gbm3_pred))

         fig = plt.figure(figsize=(12, 6))
         plt.scatter(y_test,gbm1_pred,color='coral', linewidths=2, edgecolors='k')
         plt.xlabel('True GDP per Capita')
         plt.ylabel('Predictions')
         plt.title('Gradiant Boosting prediction Performance (No feature selection)')
         plt.grid()
         plt.show()
```

```
Gradiant Boosting Performance:

all features, No scaling:
MAE: 2280.4625959347395
RMSE: 3413.6352435789836
R2_Score:  0.8589714692004253

selected features, No scaling:
MAE: 2467.2081266874507
RMSE: 3789.2979753946875
R2_Score:  0.8262238105475073
```



GDP prediction web App using Gradient Boosting Regressor Machine Learning Algorithm

```
st.title('Country GDP Estimation Tool')
st.write('''
        This app will estimate the GDP per capita for a country, given some
        attributes for that specific country as input.

        Please fill in the attributes below, then hit the GDP Estimate button
        to get the estimate.
        ''')

st.header('Input Attributes')
att_popl = st.number_input('Population (Example: 7000000)', min_value=1e4, max_v
att_area = st.slider('Area (sq. Km)', min_value= 2.0, max_value= 17e6, value=6e5
att_dens = st.slider('Population Density (per sq. mile)', min_value= 0, max_valu
att_cost = st.slider('Coastline/Area Ratio', min_value= 0, max_value= 800, value
att_migr = st.slider('Annual Net Migration (migrant(s)/1,000 population)', min_v
att_mort = st.slider('Infant mortality (per 1000 births)', min_value= 0, max_val
att_litr = st.slider('Population literacy Percentage', min_value= 0, max_value=
att_phon = st.slider('Phones per 1000', min_value= 0, max_value= 1000, value=250
att_arab = st.slider('Arable Land (%)', min_value= 0, max_value= 100, value=25,
att_crop = st.slider('Crops Land (%)', min_value= 0, max_value= 100, value=5, st
att_othr = st.slider('Other Land (%)', min_value= 0, max_value= 100, value=70, s
st.text('(Arable, Crops, and Other land should add up to 100%)')
att_clim = st.selectbox('Climate', options=(1, 1.5, 2, 2.5, 3))
st.write('''
        * 1: Mostly hot (like: Egypt and Australia)
        * 1.5: Mostly hot and Tropical (like: China and Cameroon)
        * 2: Mostly tropical (like: The Bahamas and Thailand)
        * 2.5: Mostly cold and Tropical (like: India)
        * 3: Mostly cold (like: Argentina and Belgium)
        '''
```


Country GDP Estimation Tool
This app will estimate the GDP per capita for a country, given some attributes for that specific country as input.
Please fill in the attributes below, then hit the GDP Estimate button to get the estimate.
Input Attributes
Population (Example: 7000000)
31056997.00
-
+
Area (sq. Km)
6430002.00
2.00
17000000.00
Population Density (per sq. mile)
4880
0
12000
Coastline/Area Ratio
0
0
800
Annual Net Migration (migrant(s)/1,000 population)
22


Annual Birth Rate (births/1,000)
45
7
50
Annual Death Rate (deaths/1,000)
20
2
30
Agricultural Economy
0.35
0.00
1.00
Industrial Economy
0.25
0.00
1.00
Services Economy
0.40
0.00
1.00
(Agricultural, Industrial, and Services Economy should add up t
Region

```
user_input = np.array([att_popl, att_area, att_dens, att_cost, att_migr,
                       att_mort, att_litr, att_phon, att_arab, att_crop,
                       att_othr, att_clim, att_brth, att_deth, att_agrc,
                       att_inds, att_serv, att_regn_1, att_regn_2, att_regn_3,
                       att_regn_4, att_regn_5, att_regn_6, att_regn_7,
                       att_regn_8, att_regn_9, att_regn_10, att_regn_11]).reshape(1,-1)
```

```
#Data Split
y = data_final['gdp_per_capita']
X = data_final.drop(['gdp_per_capita','country'], axis=1)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=101)

#model training
gbm_opt = GradientBoostingRegressor(learning_rate=0.01, n_estimators=500,
                                    max_depth=5, min_samples_split=10,
                                    min_samples_leaf=1, subsample=0.7,
                                    max_features=7, random_state=101)
RandomForestRegressor(random_state=101, n_estimators=500)
gbm_opt.fit(X_train,y_train)
```

```
#making a prediction
gbm_predictions = gbm_opt.predict(user_input) #user_input is taken from input attrebutes
st.write('The estimated GDP per capita is: ', gbm_predictions)
```

RESULT

As a result of the above model we have successfully predicted the GDP of country by looking at the various data of the country such as its population, area, literacy rate, birth rate, death rate, agricultural land, migration etc. We have used machine learning algorithm in predicting the GDP value. The best prediction performance was achieved using Gradient Boosting regressor, using all features in the dataset, and resulted in the following metrics: Mean Absolute Error (MAE): 2280.46 Root mean squared error (RMSE): 3413.63 R-squared Score (R2_Score): 0.85