

GDP Prediction With Machine Learning

06.12.2021

- **By:**

SUBHAJIT PRAMANIK

Roll Number: 18398

BS-MS, 4th Year

Department of Economic Science

Indian Institute of Science Education and Research
(IISER), Bhopal




Introduction:

GDP is a measure of the economic activity or economic condition of a country. The full form of GDP is **Gross Domestic Product**. Here in this project, I have tried to apply machine learning algorithms to make a model which can predict the **GDP per capita** of a country if we give the correct attributes or inputs of that specific country. Now this time, predicting GDP can be a powerful tool for the industry as well as the government also. Both government and corporate policy-makers can take action according to the prediction and get better outputs. To make this model, I have taken the data set from Kaggle and trained it, optimized it and with the final model, I have made a web application, which can create a temporary link, and by that link, anyone can see the results, i.e. the predicted GDP by giving inputs. So, here in this report at first, we will discuss the theoretical overview of GDP and then step by step we will discuss the code. In the code, after importing all the required libraries and data set, I have done data cleaning and pre-processing and then applied some selected models which I thought to be better options for this dataset. In the end, we will see the web application interface and how it works. In summary, we can say that the goal of this project is to understand the dataset, visualize the statistical dependencies, analyze by extrapolating the data and visualizing it and then try to make a model which can predict GDP per capita for any country on its own from the given inputs. So, let's begin.....

GDP - A Brief Overview:

Gross domestic product (GDP) is the total monetary or market value of all the finished goods and services produced within a country's borders in a specific time period. As a broad measure of overall domestic production, it functions as a comprehensive scorecard of a given country's economic health. Though GDP is



typically calculated on an annual basis, it is sometimes calculated on a quarterly basis as well. In the U.S., for example, the government releases an annualized GDP estimate for each fiscal quarter and also for the calendar year. The individual data sets included in this report are given in real terms, so the data is adjusted for price changes and is, therefore, net of inflation. **GDP** is defined as the market value of *all final* goods and services produced *domestically* in a single year and is the most critical measure of macroeconomic performance. A related measure of the economy's total output product is **gross national product (GNP)**, which is the market value of all final goods and services produced by a *nation* in a single year.

Measuring GDP: There are two ways of measuring GDP, the expenditure approach and the income approach.

- The **expenditure approach** is to add up the market value of all domestic expenditures made on final goods and services in a single year. Final goods and services are goods and services that have been purchased for final use or goods and services that will not be resold or used in production within the year. Intermediate goods and services, which are used in the production of final goods and services, are not included in the expenditure approach to GDP because expenditures on intermediate goods and services are included in the market value of expenditures made on final goods and services. Including expenditures on both intermediate and final goods and services would lead to double counting and an exaggeration of the true market value of GDP.
- The **income approach** to measuring GDP is to add up all the income earned by households and firms in a single year. The rationale behind the income approach is that total expenditures on final goods and services are eventually received by households and firms in the form of wage, profit, rent, and interest income. Therefore, by adding together wage, profit, rent, and interest income, one should obtain the same value of GDP as is obtained using the expenditure approach.

GDP and ML:

So far we have seen the theoretical framework from the macroeconomic point of view and ways of measuring it. But now we will see how ML can be applied to predict the GDP. From the dataset, it can be easily seen how all the variables are statistically relevant to the GDP. We will see the plots regarding this later on. From those dependent variables, the correlation coefficient can be calculated easily and from that using various regression algorithms models can be trained by a certain percentage of the data and which can predict the GDP on the basis of those corresponding correlation coefficients between the variables.

Data Source:

The data has been collected from Kaggle. The link for the same is here: <https://www.kaggle.com/fernandol/countries-of-the-world>. According to the kaggle website, all these data sets are made up of data from the US government.

Data Description:

The dataset has information on 227 countries. There are 18 columns on various factors like region, area, population density etc. Here is a short description of every column in the figure on the next page. We can see that the data file has named the columns as *Area (sq. mi.)* or *GDP (\$ per capita)*. But using this type of names for columns is very difficult while coding. So I have changed the names to simpler versions of those names like: *GDP (\$ per capita)* to *gdp_per_capita*. The list of newly named columns are as follows:

```
country", "region", "population", "area", "density", "coastline_area_ratio", "net_migration", "infant_mortality", "gdp_per_capita", "literacy", "phones", "arable", "crops", "other", "climate", "birth_rate", "death_rate", "agriculture", "industry", "service".
```



```
df.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    object
1   Region                               227 non-null    object
2   Population                           227 non-null    int64
3   Area (sq. mi.)                       227 non-null    int64
4   Pop. Density (per sq. mi.)           227 non-null    object
5   Coastline (coast/area ratio)          227 non-null    object
6   Net migration                         224 non-null    object
7   Infant mortality (per 1000 births)    224 non-null    object
8   GDP ($ per capita)                    226 non-null    float64
9   Literacy (%)                          209 non-null    object
10  Phones (per 1000)                     223 non-null    object
11  Arable (%)                            225 non-null    object
12  Crops (%)                             225 non-null    object
13  Other (%)                             225 non-null    object
14  Climate                               205 non-null    object
15  Birthrate                             224 non-null    object
16  Deathrate                             223 non-null    object
17  Agriculture                           212 non-null    object
18  Industry                              211 non-null    object
19  Service                               212 non-null    object
dtypes: float64(1), int64(2), object(17)
memory usage: 35.6+ KB
```

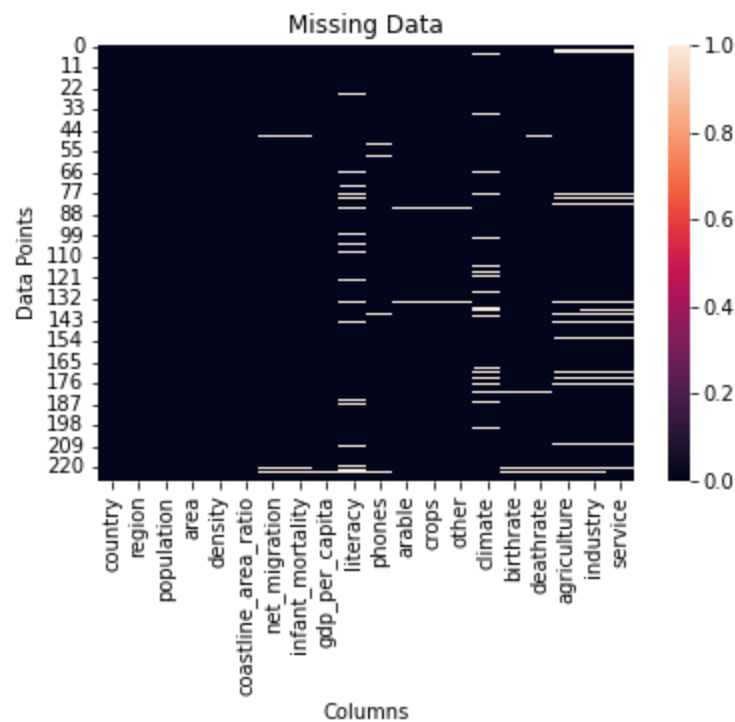
Here we see an issue; except for 'Country' and 'Region', all other columns are numerical, yet only 'Population', 'Area', and 'GDP' are float/int type; while the rest (15/20) are identified as the object type. We need to convert those into float types to continue our data analysis. Also, many columns in that dataset have *objects* as types. We will fix this by assigning float/string types to them. After this, in the code, we can see a short description of every column which says us the mean value, minimum and maximum values, standard deviation and density in various percentages as follows:

```
df.describe()
```

	population	area	density	coastline_area_ratio	net_migration	infant_mortality	gdp_per_capita	literacy	phones	arable	crops	other	climate	birthrate	deathr
count	2.270000e+02	2.270000e+02	227.000000	227.000000	224.000000	224.000000	226.000000	209.000000	223.000000	225.000000	225.000000	225.000000	205.000000	224.000000	223.000
mean	2.874028e+07	5.982270e+05	379.047137	21.165330	0.038125	35.506964	9689.823009	82.838278	236.061435	13.797111	4.564222	81.638311	2.139024	22.114732	9.241
std	1.178913e+08	1.790282e+06	1660.185825	72.286863	4.889269	35.389899	10049.138513	19.722173	227.991829	13.040402	8.361470	16.140835	0.699397	11.176716	4.990
min	7.026000e+03	2.000000e+00	0.000000	0.000000	-20.990000	2.290000	500.000000	17.600000	0.200000	0.000000	0.000000	33.330000	1.000000	7.290000	2.290
25%	4.376240e+05	4.647500e+03	29.150000	0.100000	-0.927500	8.150000	1900.000000	70.600000	37.800000	3.220000	0.190000	71.650000	2.000000	12.672500	5.910
50%	4.786994e+06	8.660000e+04	78.800000	0.730000	0.000000	21.000000	5550.000000	92.500000	176.200000	10.420000	1.030000	85.700000	2.000000	18.790000	7.840
75%	1.749777e+07	4.418110e+05	190.150000	10.345000	0.997500	55.705000	15700.000000	98.000000	389.650000	20.000000	4.440000	95.440000	3.000000	29.820000	10.605
max	1.313974e+09	1.707520e+07	16271.500000	870.660000	23.060000	191.190000	55100.000000	100.000000	1035.600000	62.110000	50.680000	100.000000	4.000000	50.730000	29.740

NaN values check:

Then in the code, search for NaN values, i.e. the blank data points have been searched out to avoid complications in future model training. Here I have prepared a heat map:



From this heat map, the NaN points can be seen easily. The white lines are those points where there is no data available. We can see from above that we have some missing data points, but it is not extensive. 14/20 of our columns have missing data points, the maximum percentage of missing data is in the 'Climate' column, and it is less than 10% (22/227). Later in the project, we will deal with these missing data points in the data cleaning section.

Undefined features Check:

In the data sets, some columns are not defined well. We need to understand what different values in the **climate**, **agriculture**, **industry**, and **service** columns refer to.

	country	region	climate	agriculture	industry	service
0	Afghanistan	ASIA (EX. NEAR EAST)	1.0	0.380	0.240	0.380
1	Albania	EASTERN EUROPE	3.0	0.232	0.188	0.579
2	Algeria	NORTHERN AFRICA	1.0	0.101	0.600	0.298
3	American Samoa	OCEANIA	2.0	NaN	NaN	NaN
4	Andorra	WESTERN EUROPE	3.0	NaN	NaN	NaN

If we see it carefully, then it can be clearly seen here that the values in (**agriculture**, **industry**, and **service**) columns are the percentages of different sectors in each country's economic activity. For example: agriculture is generating 38% of Afghanistan's GDP, the industry generates 24%, while service generates 38%; the total is 100%. Still, the **climate** column is not clear to us. So, now we will understand this column. For this I compared the climate data with the region's climate which is very dependent on the region of a country. So, for this, I ran the following code:

```
h1 = df.loc[:, ['country', 'region', 'climate']][df.climate == 1].head()
h2 = df.loc[:, ['country', 'region', 'climate']][df.climate == 2].head()
h3 = df.loc[:, ['country', 'region', 'climate']][df.climate == 3].head()
h4 = df.loc[:, ['country', 'region', 'climate']][df.climate == 4].head()
h5 = df.loc[:, ['country', 'region', 'climate']][df.climate == 1.5].head()
h6 = df.loc[:, ['country', 'region', 'climate']][df.climate == 2.5].head()
pd.concat([h1, h2, h3, h4, h5, h6])
```


The output from these few lines of code was fascinating. The table from the output is like this:

	country	region	climate
0	Afghanistan	ASIA (EX. NEAR EAST)	1.0
2	Algeria	NORTHERN AFRICA	1.0
11	Australia	OCEANIA	1.0
13	Azerbaijan	C.W. OF IND. STATES	1.0
15	Bahrain	NEAR EAST	1.0
3	American Samoa	OCEANIA	2.0
6	Anguilla	LATIN AMER. & CARIB	2.0
7	Antigua & Barbuda	LATIN AMER. & CARIB	2.0
10	Aruba	LATIN AMER. & CARIB	2.0
14	Bahamas, The	LATIN AMER. & CARIB	2.0
1	Albania	EASTERN EUROPE	3.0
4	Andorra	WESTERN EUROPE	3.0
8	Argentina	LATIN AMER. & CARIB	3.0
12	Austria	WESTERN EUROPE	3.0
19	Belgium	WESTERN EUROPE	3.0
9	Armenia	C.W. OF IND. STATES	4.0
18	Belarus	C.W. OF IND. STATES	4.0
25	Bosnia & Herzegovina	EASTERN EUROPE	4.0
69	France	WESTERN EUROPE	4.0
106	Kazakhstan	C.W. OF IND. STATES	4.0
24	Bolivia	LATIN AMER. & CARIB	1.5

From this output table, we can see that along with nan (representing missing data), the climate has six unique values, and they are: 1, 1.5, 2, 2.5, 3, and 4. our observations:

1. Countries with mostly desert/hot climates have 1
2. Countries with mostly tropical climates have 2
3. Countries with mostly cold/cool Climate have 3
4. Countries with Climate almost equally divided between hot and tropical have 1.5

5. Countries with Climate almost similarly divided between cold and tropical have 2.5
6. Countries under 'Climate' = 4 belong to the cold/cool climate group; It is not mentioned in the dataset source why this group is separate from group 3; yet we will combine both groups together in the data cleaning section of the project.
7. There are 22 countries with null values for the climate column; those will be replaced by 0 in a later step, where 0 will represent an 'unknown' value.

Data Cleaning:

So, the heat map for the missing data we already have seen in the earlier section. Now we need to fix this. The NaN values are distributed in every column as follows:

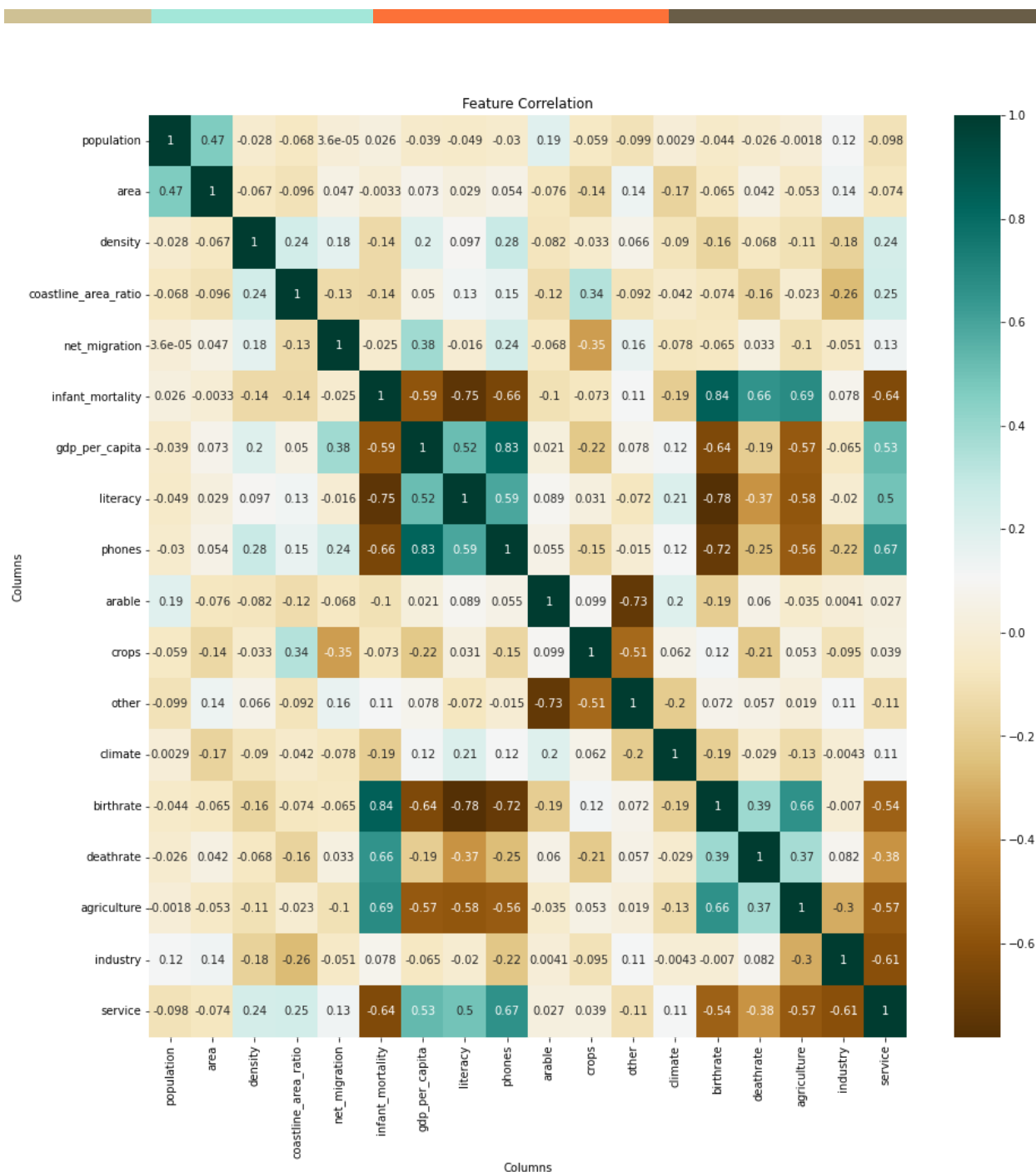
country	0
region	0
population	0
area	0
density	0
coastline_area_ratio	0
net_migration	3
infant_mortality	3
gdp_per_capita	1
literacy	18
phones	4
arable	2
crops	2
other	2
climate	22
birthrate	3
deathrate	4
agriculture	15
industry	16
service	15
dtype: int64	

To fix these points, I have done the following:

1. net_migration: 3 missing data points. All of them belong to tiny nations. We will put zero for those 3.
2. infant_mortality: 3 missing data points. All of them belong to very small nations. We will put zero for those 3.
3. gdp_per_capita: 1 missing value. West Sahara, from an internet search, their GDP per capita is \$2500, and we will put this value into our data set.
4. Literacy: 18 missing values, replaced by the mean literacy of each missing value's region.
5. Phones: 4 missing values, replaced by the mean phones of each missing value's region.
6. Arable, crops, and other: 2 missing values of very small islands, replaced with zero.
7. Climate: 22 missings, replaced with 0, where zero will represent an 'unknown' value.
8. Birth rate and death rate: 3 missings, replaced with their region's mean rates, since those rates are per 1000 and not population-related.
9. Agriculture, industry, and service: 15 missing values belong to very small island nations. After inspection for similar nations, I found that those kinds of nations usually have economies that rely heavily on services, with some agricultural and industrial activities. So I have replaced the missing values with the following: agriculture = 0.15, industry = 0.05. service = 0.8.

Extrapolatory Data Analysis:

For EDA, at first I prepared the correlation heatmap between all the columns to see how they are varying with each other and got the following plot.

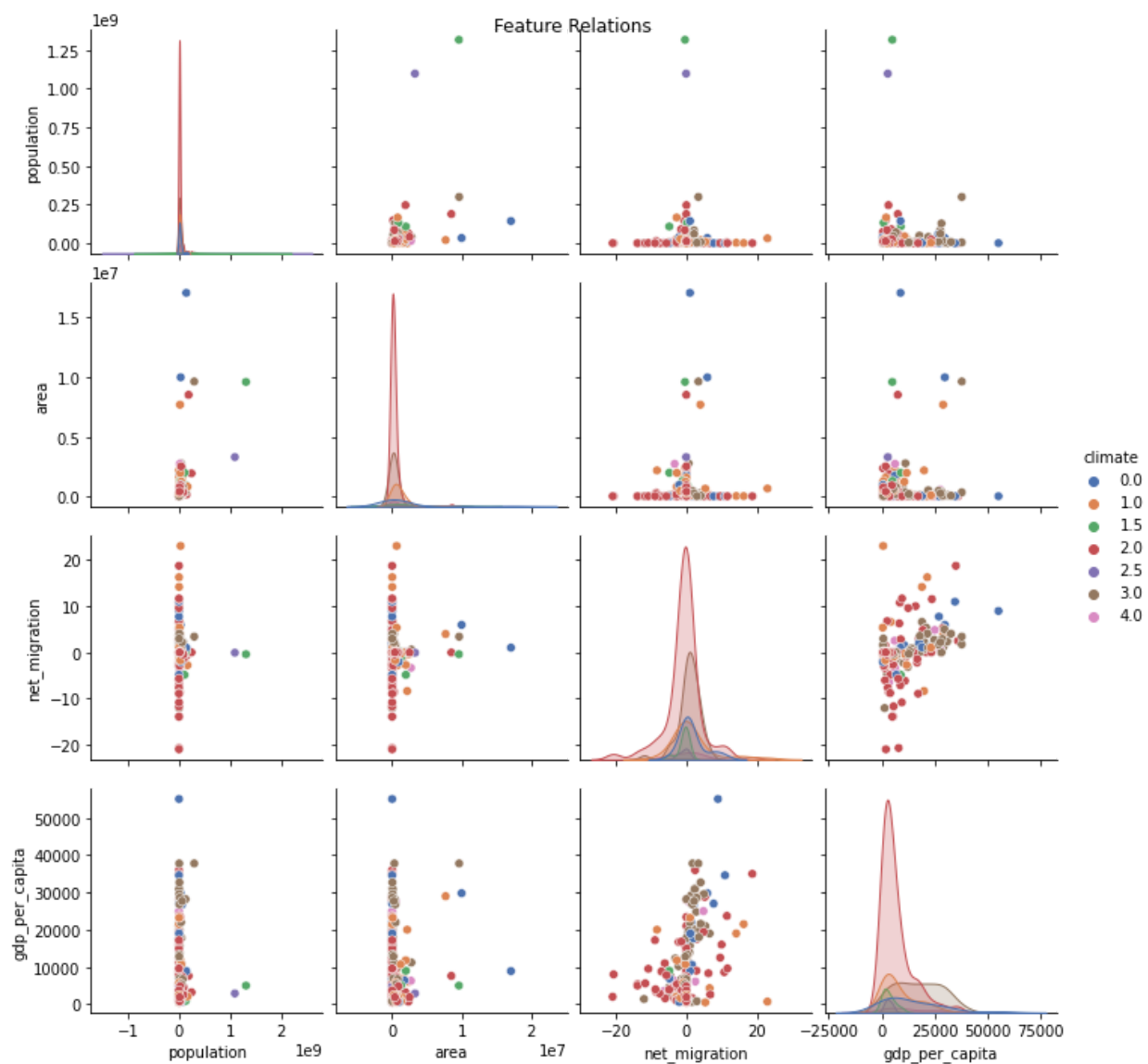


So, the insights we can see from the above correlation heatmap are as follows:

1. An expected strong correlation between *infant_mortality* and *birth rate*
2. An unexpected strong correlation between *infant_mortality* and *agriculture*
3. An expected strong correlation between *infant_mortality* and *literacy*

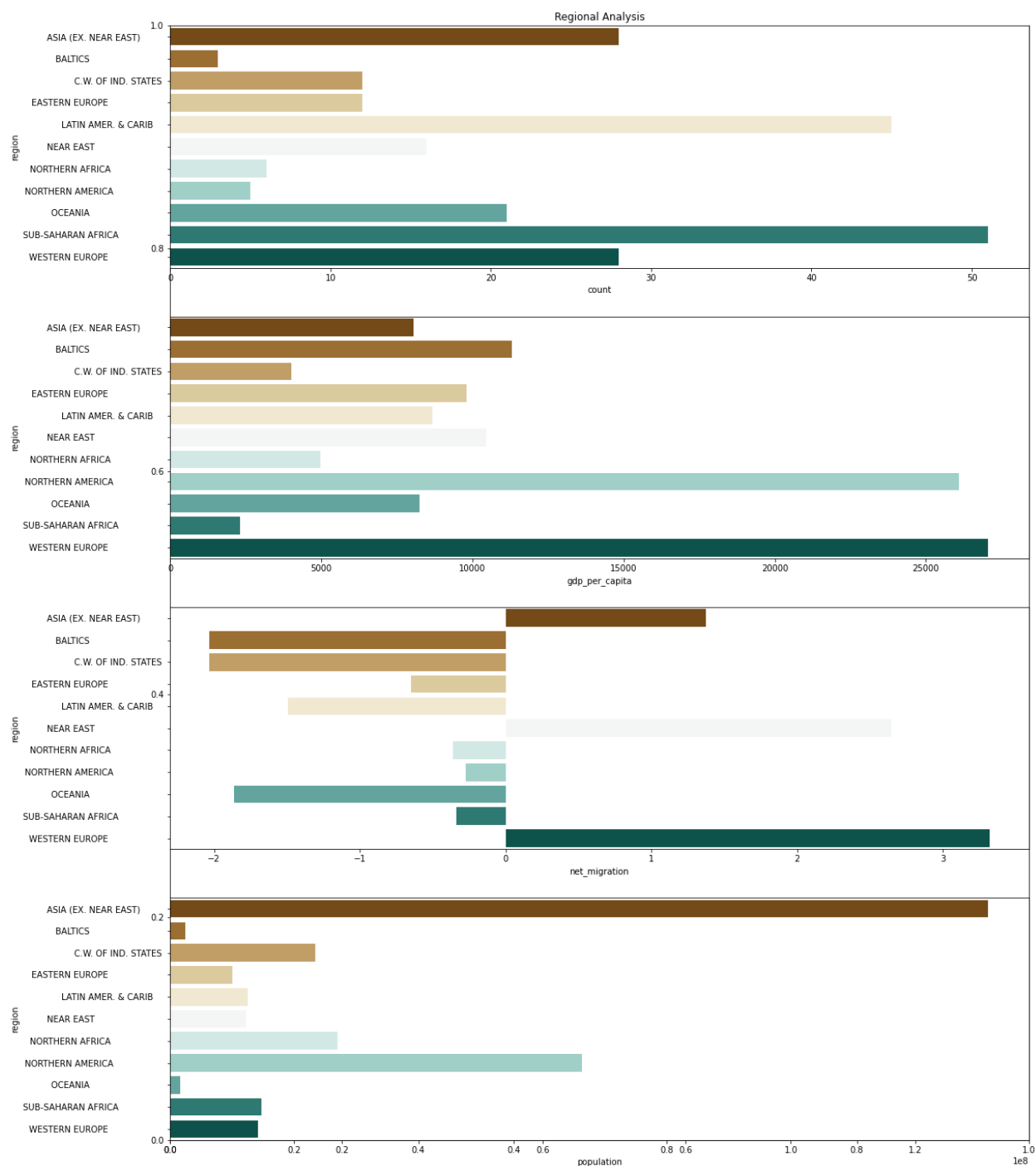
4. An expected strong correlation between *gdp_per_capita* and *phones*
5. An expected strong correlation between *arable* and *other* (other than crops)
6. An expected strong correlation between *birth rate* and *literacy* (the less literacy, the higher the birthrate)
7. An unexpected strong correlation between *birthrate* and *phones*.

Now let's see how correlations are among a selected few features, which I think may be more important compared to the other columns:



We can see a reasonable correlation between GDP and migration, which makes sense since migrants tend to move to countries with better opportunities and higher GDP per capita.

Regression Analysis:

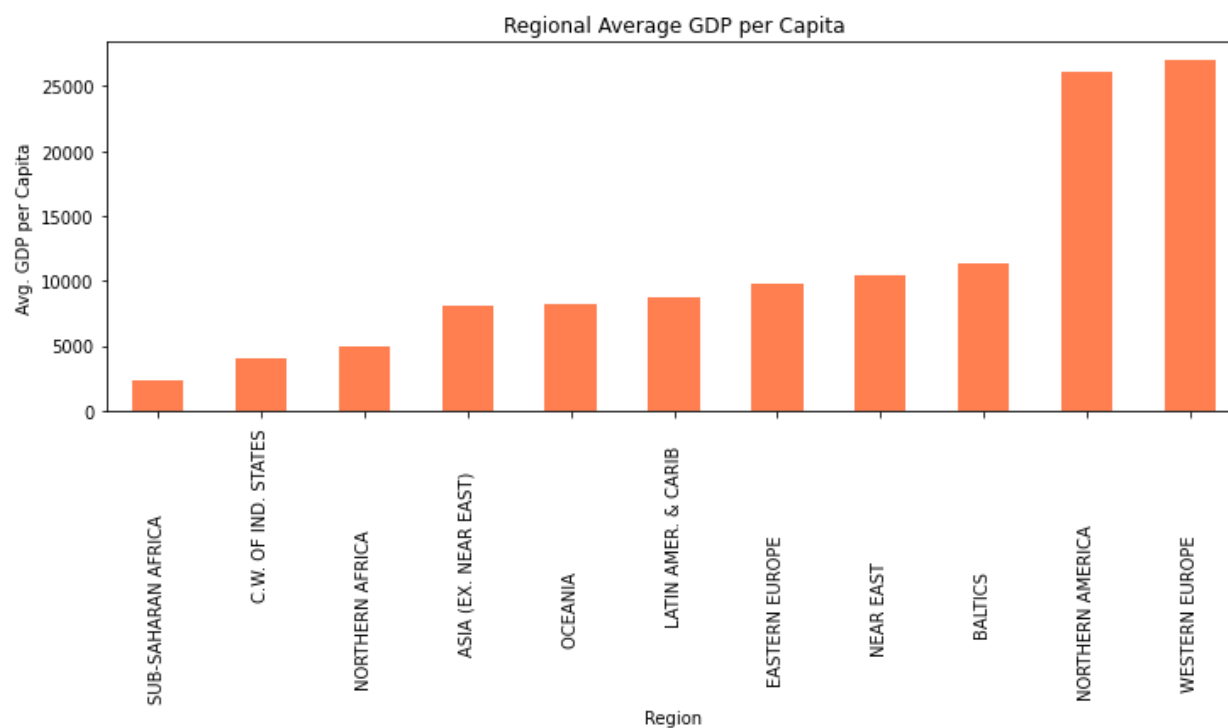


From the above figure of regression analysis, the following points can be concluded:

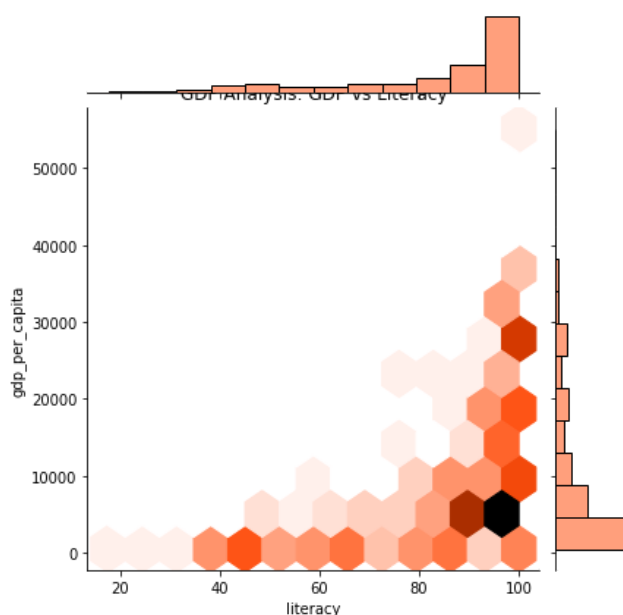
1. Sub-Saharan Africa and Latin America regions have the most countries within them.
2. Western Europe and North America have the highest GDP per capita, while Sub-Saharan Africa has the lowest.
3. Asia, North America, and North Europe are the central regions where migrants from the other areas go.
4. Asia has the largest population, Oceania has the smallest.

GDP Analysis:

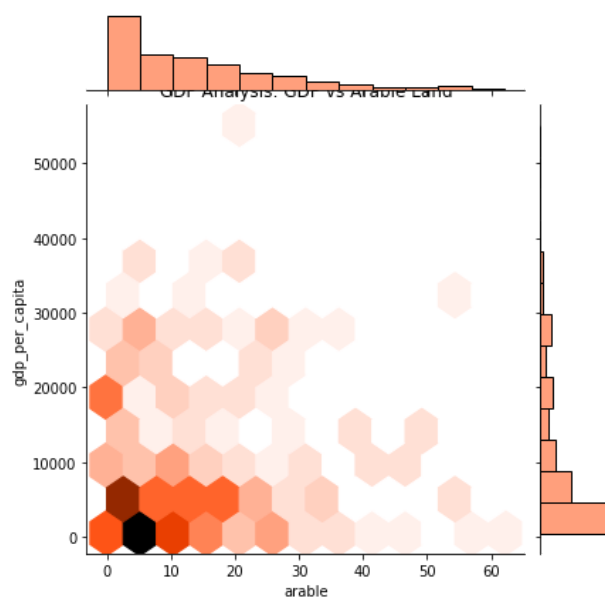
The figure below shows the regional ranking according to the average GDP per capita. As expected, North America and Western Europe have the highest GDP per capita. In contrast, Sub Saharan Africa has the lowest, which may describe the big migration trends in the world in the past decade.



Now we will see some more plots which help us to understand the variation of GDP with other features in different countries.

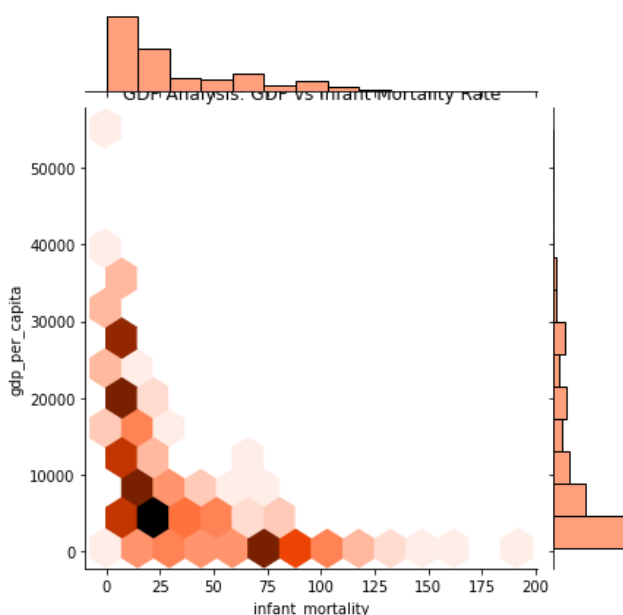


From the above figure, it is clear that the higher the country's GDP, the more literate the population is, and vice versa.



No clear relationship between GDP and percentage of arable land, an indication that agriculture is not the strongest factor economically, as it used to be for most of human history of all time.

From the next figure we can interpret that, it is very clear that poor countries suffer more from infant mortality.



Data Pre-Conditioning and Splitting:

Here in this section, we made the data ready for the training. For this, three steps has been done as follows:

1. Transform the 'region' column into numerical values.
2. Split data set into training and testing parts (80/20), while dropping the countries column (string, and not going to be used to train the models), and separating gdp_per_capita column, where it will be used as labels.
3. We will try different splits of our dataset (with/without feature selection, with/without feature scaling).

For the splitting I have tried four different methods as:

1. Select all of our final dataset with no scaling
2. Select **all of our final dataset but with scaling**
3. Select only a portion of our features, the ones with correlation score larger than ± 0.3 with *gdp_per_capita*.
4. Feature selected dataset as last point but this time with scaling.

After this training, the regression has been done with the help of various machine learning algorithms. Here I have used four different types of algorithms i.e. **linear regression**, **SVM**, **gradient boosting** and **random forest**.

Linear regression:

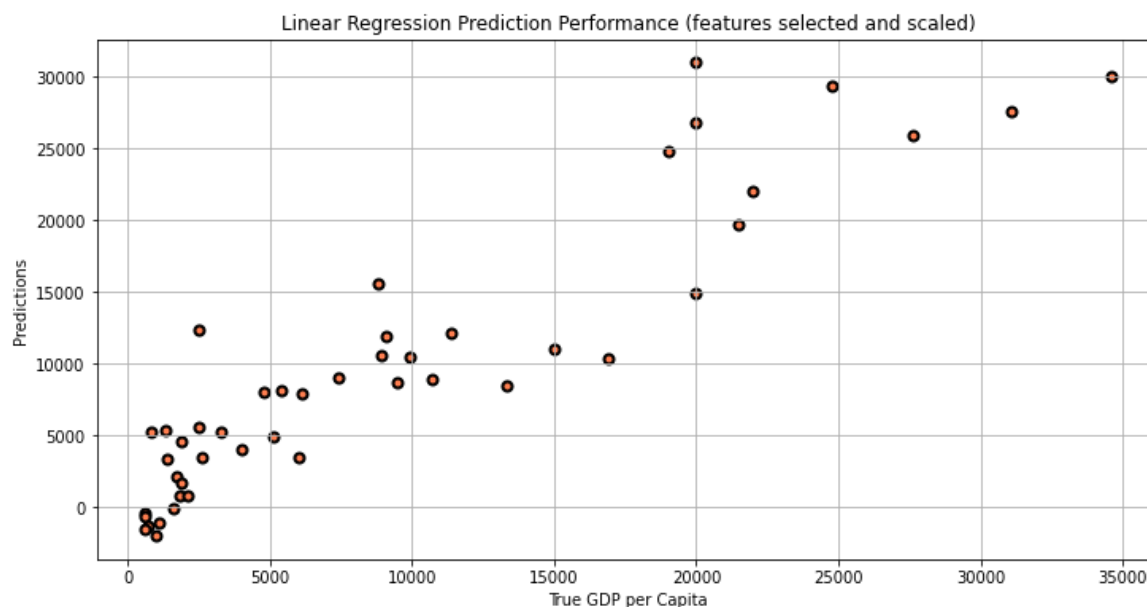
From our EDA, we can see that most features do not have a linear relationship with our labels (*gdp_per_capita*), yet I tried linear regression and used the result as a reference (as other methods should have better results). The results we got are:

```
all features, No scaling:  
MAE: 330350.8586601445  
RMSE: 1570337.5456390344  
R2_Score: -29843.120383351565
```

```
all features, with scaling:  
MAE: 569019.4687589288  
RMSE: 1283170.8219654495  
R2_Score: -19925.990118469563
```

```
selected features, No scaling:  
MAE: 2965.9357229398743  
RMSE: 4088.794580247939  
R2_Score: 0.7976685756859008
```

```
selected features, with scaling:  
MAE: 2879.5213243944404  
RMSE: 3756.4365885029656  
R2_Score: 0.8292247702712091
```



From the metrics above, it is clear that feature selection is essential for linear regression model training in order to get acceptable results on this dataset. On the other hand, feature scaling has a small positive effect on LR's prediction performance. Here we can see a decent prediction performance from LR with feature selection and scaling.

Support Vector Machine:

The results from the support vector machine are very poor. Feature scaling, and feature selection, made almost no difference in the prediction performance.

SVM Performance:

all features, No scaling:

MAE: 7049.984895264721

RMSE: 9811.73631340298

R2_Score: -0.16510345624387246

all features, with scaling:

MAE: 7042.737596769212

RMSE: 9800.406046613498

R2_Score: -0.16241416444556656

selected features, No scaling:

MAE: 7047.711927073501

RMSE: 9807.997922107874

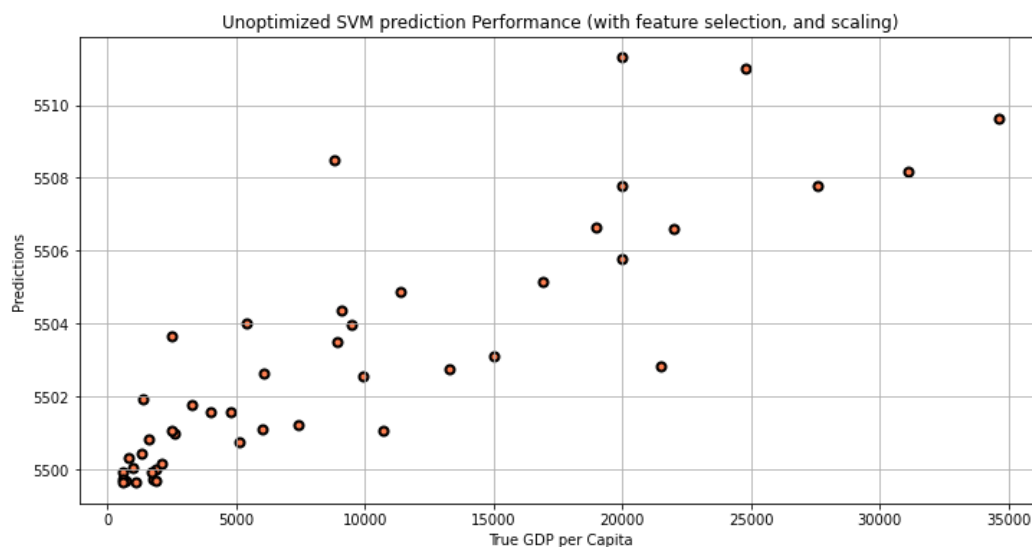
R2_Score: -0.16421578810668724

selected features, with scaling:

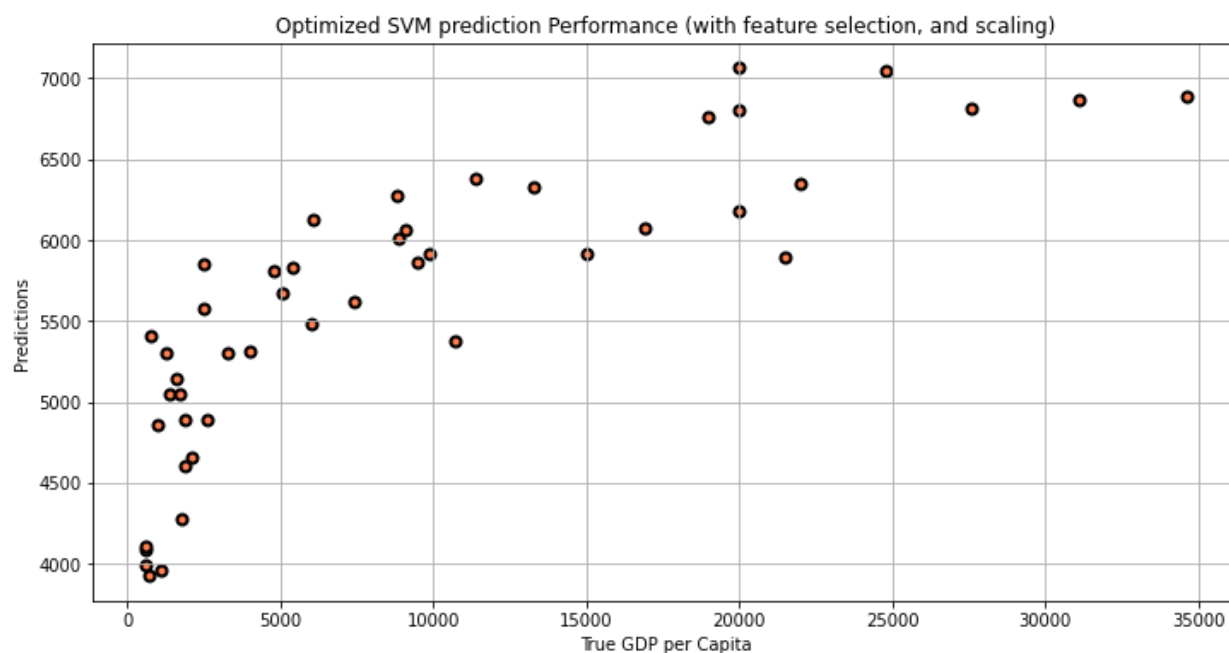
MAE: 7040.043820847137

RMSE: 9794.58886537642

R2_Score: -0.1610346364957338



The results of SVM are worse than that of Linear Regression, so I tried to improve SVM's performance by optimizing its parameters using grid search. After optimizing it we got: **MAE: 6386.413128432553, RMSE: 9133.499345710767, R2_Score: -0.009594923559210988.**



SVM has improved a little with grid search, but it still performs below linear regression.

Random Forest Regression:

After SVM, I tried **random forest** with the data splits (with and without feature selection). Scaling will not be tested for Random Forest since it should not affect this algorithm's performance. Later on, I tried to optimize this to get a better result. So, the results I got before optimization are as follows:

Random Forest Performance:

all features, No scaling:

MAE: 2142.1304347826085

RMSE: 3097.1944738255706

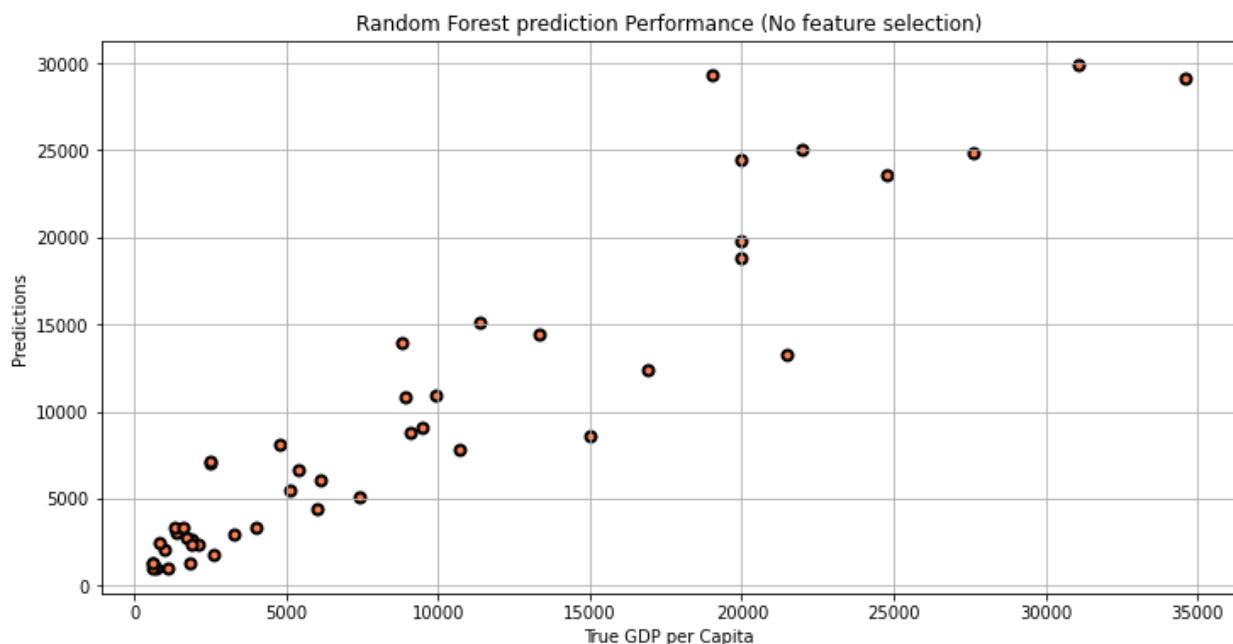
R2_Score: 0.8839060185534444

selected features, No scaling:

MAE: 2416.0652173913045

RMSE: 3533.590316058036

R2_Score: 0.8488858452472634



The results are better in a random forest regressor. After this, I tried to use grid search in order to obtain good parameters for the random forest regressor. The optimization here will be limited due to time and computing power constraints.

The parameters I used to optimize are as follows: (i) n_estimators, (ii) min_samples_leaf, (iii) max_features and (iv) bootstrap.

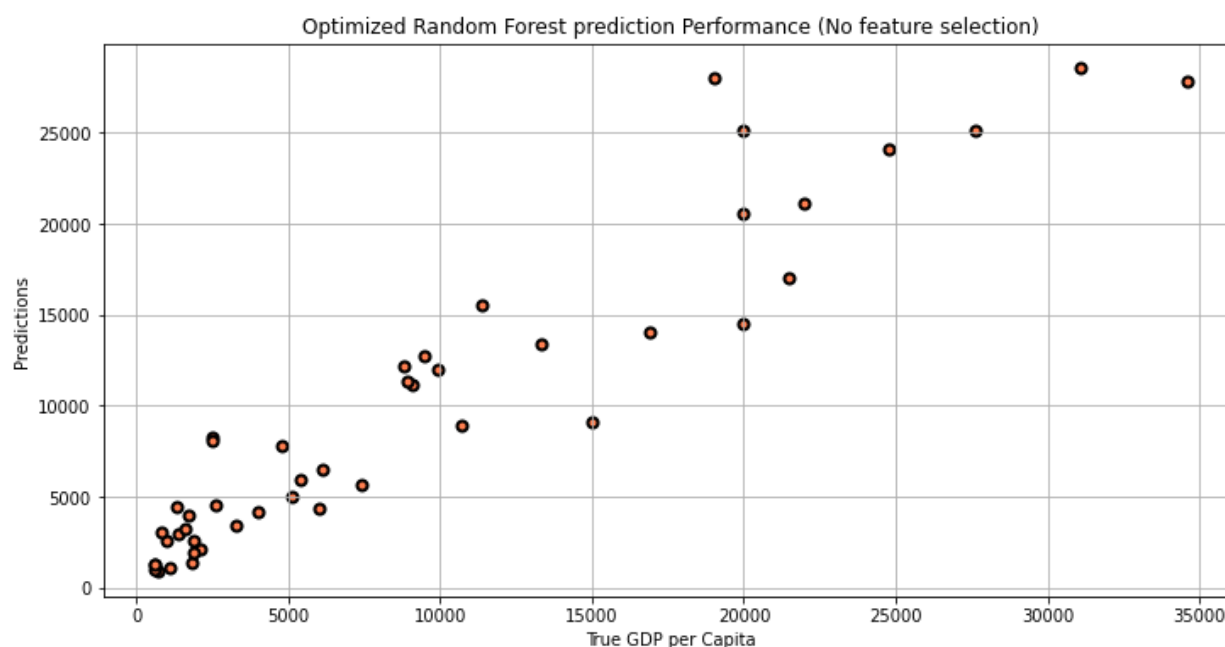
The results after the optimization are:

MAE: 2230.6304347826085

RMSE: 3061.91390487544

R2_Score: 0.8865358395020386

The plot for this optimized model is like this:



We can see that the optimization process on random forest regressor has not changed the performance in a noticeable manner, yet the slight change was actually to the worst, that is probably because the initial parameters were already very close to the optimum ones.

Gradient Boosting:

At first, I tried to train the GBM regressor with the default parameter values. The values I got indicate that the gradient boosting is giving better performance even before optimization. Its performance on our dataset is very close to that of Random Forest.

Gradient Boosting Performance:

all features, No scaling:

MAE: 2280.4625959347395

RMSE: 3413.6352435789836

R2_Score: 0.8589714692004253

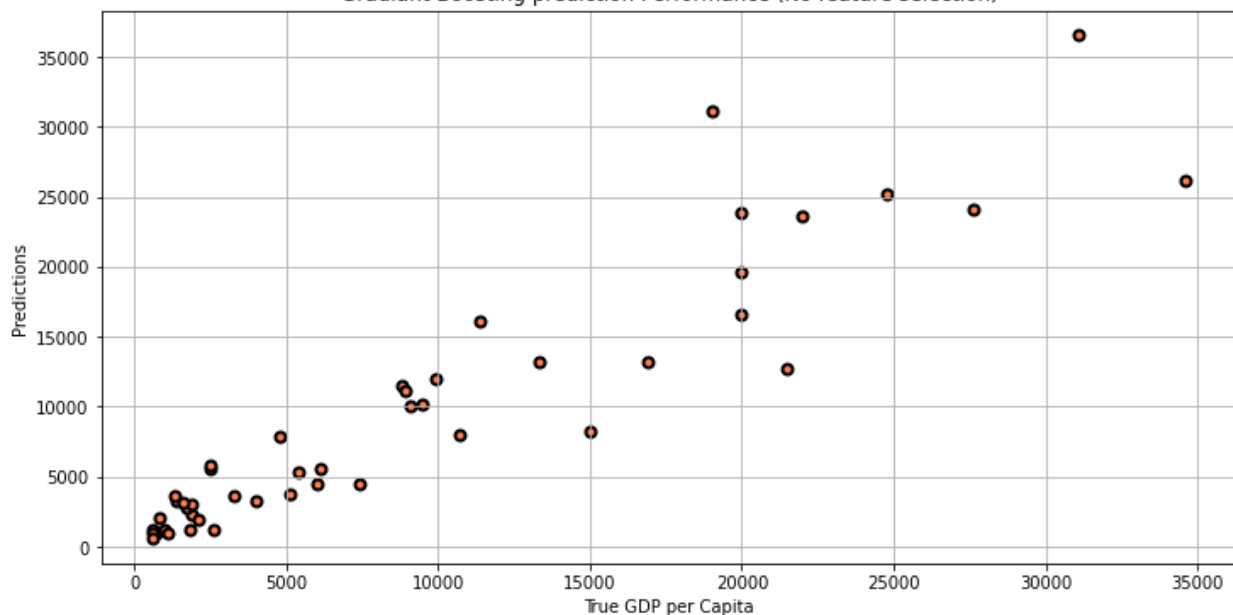
selected features, No scaling:

MAE: 2467.2081266874507

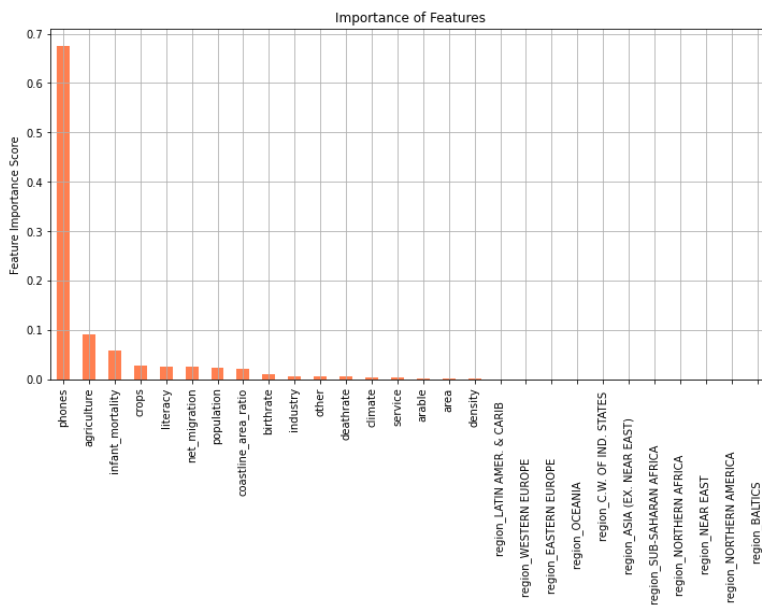
RMSE: 3789.2979753946875

R2_Score: 0.8262238105475073

Gradient Boosting prediction Performance (No feature selection)



Now from the below plot, we can see how this gradient boosting regressor sees the importance of different features in the dataset.

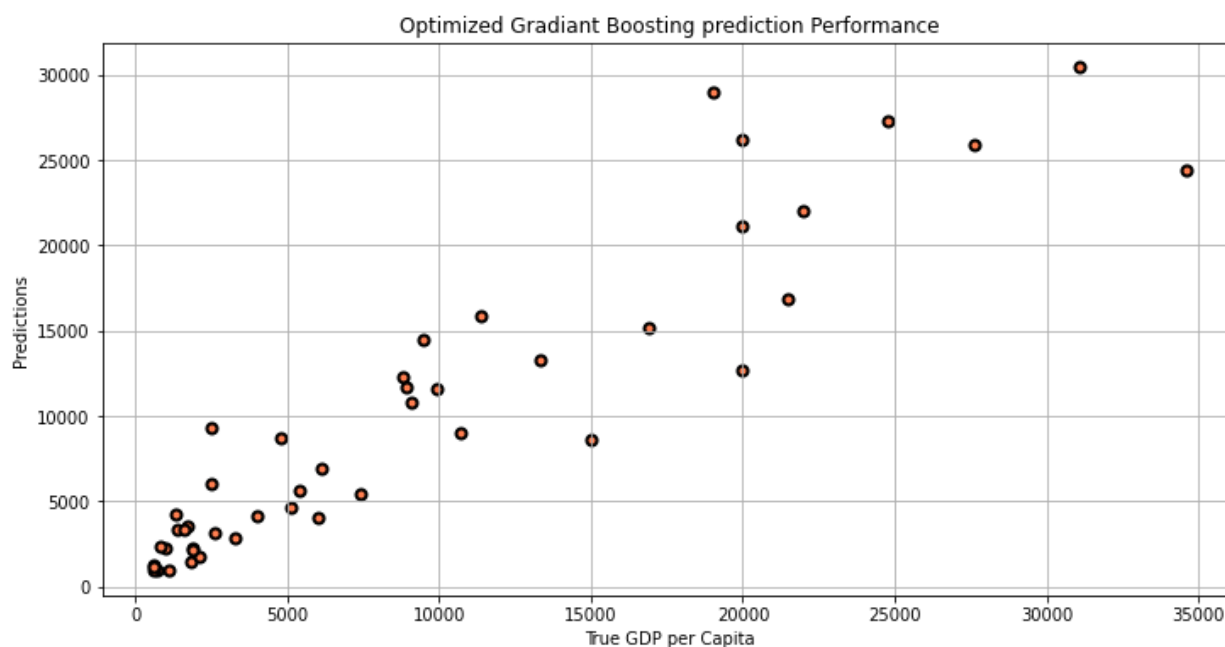


From the above plot, the number of phones seems to have the highest predictive power. With this first model, we obtain an **R2_Score** of **0.8589**, which is not very far behind that of the Random Forest regressor. So, after this, I tried to optimize GBM, and compare its performance with Random Forest and the one above. Also, I have plotted the feature importance chart and observed if GBM changed the features importance score after optimization.

GBM Optimization:

For optimization I have used grid search in order to obtain good parameters for our GBM regressor. The optimization here also is limited due to time and computing power constraints. The parameters I have optimized are:

- n-estimators: 100, 500, 1000
- learning_rate: 0.001, 0.01, 0.1, 1
- max_depth: 3, 5, 8
- subsample: 0.7, 1 (Values lower than one generally lead to a reduction of variance and an increase in bias)
- min_samples_leaf: 1, 20
- min_samples_split: 0.5-1% of our data --> we have 227 data points --> 10-20
- max_features: 4, 7 (sqrt of the number of features is a good guess)



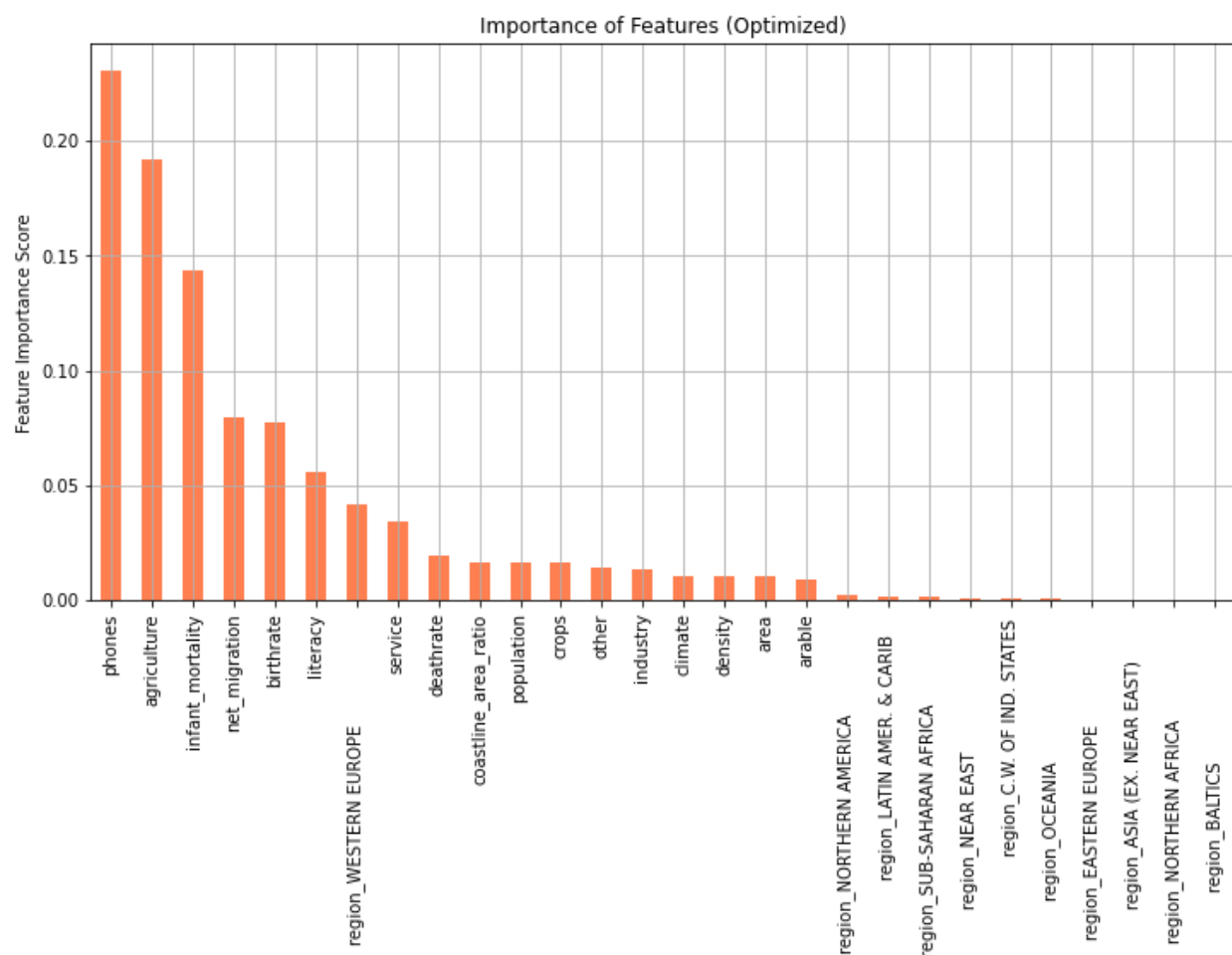
So, the results after optimizing are:

MAE: 2362.9354066125907

RMSE: 3469.360881371261

R2_Score: 0.8543294671599899

And the importance of the features after optimization are:



Here it can be seen that the grid search actually decreased the GBM performance a bit. The reason is that I could not extend the grid search limits due to the processing limits of my machine. Yet, it can be noticed that grid search resulted in different features' importance scores. In general, we can say that GBM has a similar performance to that of Random Forest on our dataset.

Further Possibilities:

When more time and computing power are available, we can try extending the ranges of Random Forest and Gradient Boosting grid searches and see if that gives us better performance than what has already been achieved.

Conclusion:

In this project, we used “countries_of_the_world.csv” dataset to build a **GDP** predictor. 4 different learning regressors (Linear Regression, SVM, Random Forest, and Gradient Boosting) were tested, and we have achieved the best prediction performance using Random Forest, followed by Gradient Boosting, and then Linear Regression, while SVM achieved the worst performance of the 4.

The best prediction performance was achieved using **Optimized Random Forest** regressor, using all features in the dataset, and resulted in the following metrics:

- Mean Absolute Error (**MAE**): 2230.6304347826085
- Root mean squared error (**RMSE**): 3061.91390487544
- R-squared Score (**R2_Score**): 0.8865358395020386

Deployment of the Model:

From the conclusion, we can see that the best model giving the highest possible prediction is the **Optimized Random Forest** algorithm. So I have created another code only using this model, which can produce a temporary link using a local tunnel that will direct to a web **application** where anyone can predict GDP per capita for any country by giving the attributes or inputs. In this way, this model can work as a predictor or calculator of **GDP per capita**. The layout I used here for the web page is

largely influenced by a GitHub repo whose link is in the reference. Below is the screenshot of the web application **GDP Predictor By Machine Learning**:

GDP Predictor by Machine Learning

This app will estimate the GDP per capita for a country on the basis of the given attributes for that specific country as input using Machine Learning algorithms.

Fill in the attributes below and then click on the GDP Estimate button to get the estimate.

By Subhajit Pramanik (4th Year BS - MS, Department of Economic Science, Indian Institute of Science Education and Research, Bhopal)

Input Attributes

Population (Example: 7000000)

20000000.00 - +

Area (sq. Km)

6000000.00 2.00 17000000.00

Population Density (per sq. mile)

400 0 12000

Coastline/Area Ratio

30 0 800

Annual Net Migration (migrant(s)/1,000 population)

0

References:

- <https://towardsdatascience.com/demystifying-maths-of-svm-13ccfe00091e>
- <https://www.investopedia.com/terms/g/gdp.asp>
- https://www.bis.org/ifc/publ/ifcb50_15.pdf
- <https://www.mdpi.com/2227-7390/8/2/241/htm>
- <https://github.com/zeglam/Countries-GDP-prediction>
- <https://github.com/streamlit/streamlit>
- <http://www.diva-portal.org/smash/get/diva2:1075467/FULLTEXT01.pdf>
- <https://www.mckinsey.com/featured-insights/artificial-intelligence/notes-from-the-ai-frontier-modeling-the-impact-of-ai-on-the-world-economy>
- <https://www.pwc.com/gx/en/issues/data-and-analytics/publications/artificial-intelligence-study.html>
- <https://data.oecd.org/gdp/real-gdp-forecast.htm>
- <https://towardsdatascience.com/predicting-real-gdp-growth-85f34fdca97a>