Let’s up GDP

Analysis of GDP

University of Mumbai

DEPARTMENT OF STATISTICS

CERTIFICATE

This is to certify that the following students of M.Sc. Part 2, have successfully completed the project entitled –

"Let's up GDP"

During the academic year 2021-2022

Students involved in this group are:

Yogesh Kerkar

Ganesh Walimbe

Jitendra Dalavi

Prashant Kalebag

Nikhil Pokharkar

This Research project, to the best of our knowledge and belief, is Original.

Project Mentor
Ms. Sheetal Chabukswar

Acknowledgement

WE ARE GRATEFUL TO THE HEAD OF OUR STATISTICS DEPARTMENT Dr. V. U. DIXIT MA'AM FOR GIVING US OPPORTUNIITY TO DO THIS PROJECT.

WE LIKE TO THANK OUR GUIDE Ms. SHEETAL CHABUKSWAR MA'AM AND Dr. S.R. KULKARNI SIR WHO GAVE US HIS VALUABLE SUGGESTIONS AND IDEAS THROUGOUT THE DURATION OF THE PROJECT.

WE WOULD LIKE TO THANK ALL OUR RESPONDERS FOR BEING PART OF OUR RESEARCH PROJECT AND HELPING US WITH THEIR VALUABLE FEEDBACK.

WE ARE DEEPLY GRATEFUL TO ALL THE TEACHING AND NON- TEACHING STAFF MEMBERS OF THE STATISTICS DEPARTMENT AT THE UNIVERSITY OF MUMBAI AND OUR CLASSMATES AT POST GRADUATE IN STATISTICS FOR SUPPORTING US IN VARIOUS ASPECTS.

FINALLY, WE ARE GRATEFUL TO ALL INVOLVED IN THIS PROJECT, AS WITHOUT THEIR INSPITRATION AND VALUABLE SUGGESTIONS IT WOULD NOT HAVE BEEN POSSIBLE TO DEVELOP THE PROJECT WITHIN THE PRESCRIBED TIME.

Index

Sr. no	Content	Page no.
1	Introduction	5
2	Methodology	6
3	Objectives	7
4	Introduction of the Data	8
5	Data Cleaning	9
6	Graphical Representation	11
7	Model Building	15
8	Multiple Linear Regression	17
9	Lasso Regression	24
10	Random Forest Technique	26
11	Conclusion	28
12	Python Coding	29

Introduction

Gross Domestic Product (GDP) is the money value of all final goods and services produced for sale within an economy over one year. It provides an economic snapshot of a country, used to estimate the size of an economy and growth rate. Though it has limitations, GDP is a key tool to guide policy-makers, investors, and businesses in strategic decision-making. The GDP of a country tends to increase when the total value of goods and services that domestic producers sell to foreign countries exceeds the total value of foreign goods and services that domestic consumers buy. Gross domestic product indicates three significant figures in a country's economy vision, which are national income, national output, and the national expenditure GDP effects some important issues in a country like; unemployment, standard of living, and investment.

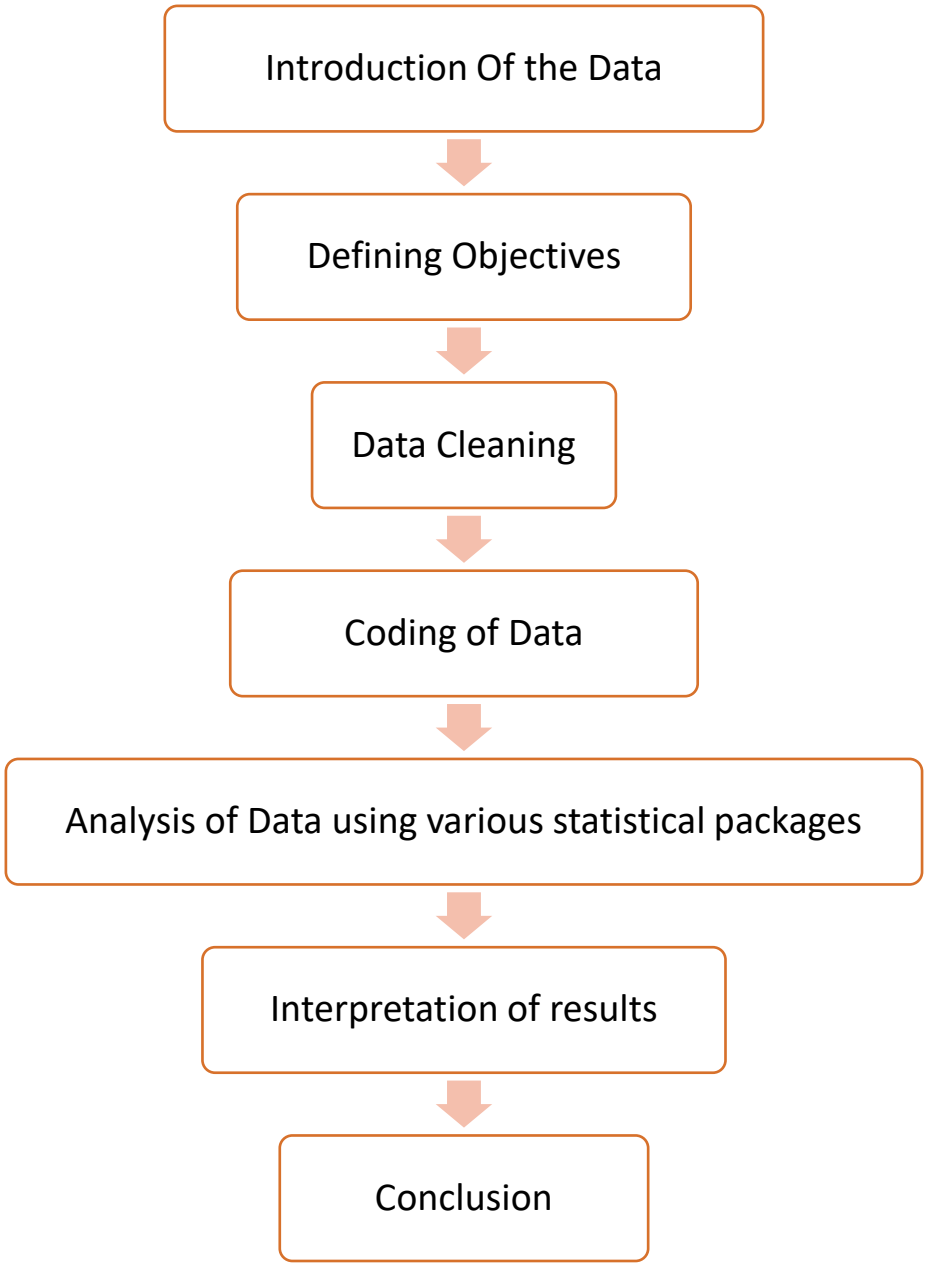
GDP per capita measures the economic output of a nation per person. It seeks to determine the prosperity of a nation by economic growth per person in that nation. It is calculated by dividing the GDP of a country by its population.

Per capita GDP is a global measure for gauging the prosperity of nations and is used by economists, along with GDP, to analyze the prosperity of a country based on its economic growth.

Small, rich countries and more developed industrial countries tend to have the highest per capita GDP.

So, throughout this project we will get to know some interesting facts and insights

Methodology



Objective

1) **Estimate the top factors that highly correlated with GDP**

2) **To compare the Economy structure for the ten countries with highest total GDP**

3) **To Build a predictive model based on important factors**

Introduction of Data

The Data is secondary data and directly downloaded from

https://data.worldbank.org/

- **The data consist of 178 countries**
- **Total no. of variables: 15**

- **Dependent variable:**
  - GDP per capita

- **Independent variables:**
  - Agriculture, forestry, and fishing
  - Land area (sq. km)
  - Death rate, crude (per 1,000 people)
  - Birth rate, crude (per 1,000 people)
  - Industry (including construction) value added per worker (constant 2015 US$)
  - Mobile cellular subscriptions (per 100 people)
  - Mortality rate (per 1,000 live births)
  - population total
  - Services value added (constant 2015 US$)
  - Exports of goods and services (constant 2015 US$)
  - Imports of goods and services (constant 2015 US$)
  - Population density (people per sq. km of land area)

Data Cleaning

Data preprocessing transforms the data into a format that is more easily and effectively processed in data mining, machine learning and other data science tasks.

Virtually any type of data analysis, data science or AI development requires some type of data preprocessing to provide reliable, precise and robust results for enterprise applications.

Real-world data is messy and is often created, processed and stored by a variety of humans, business processes and applications. As a result, a data set may be missing individual fields, contain manual input errors, or have duplicate data or different names to describe the same thing. Humans can often identify and rectify these problems in the data they use in the line of business, but data used to train machine learning or deep learning algorithms needs to be automatically preprocessed.

Dealing with missing values:

There are some ways to deal with missing values such that

1) Remove rows with missing data
2) Impute missing values with mean/median/mode

Type of variable	Impute with
Numerical	Mean/Median
Categorical	Mode

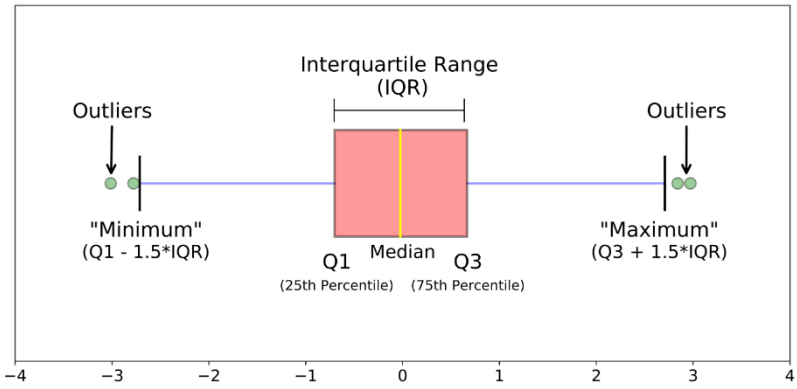
3) If data contains outliers which are true values, then fill the missing values with median

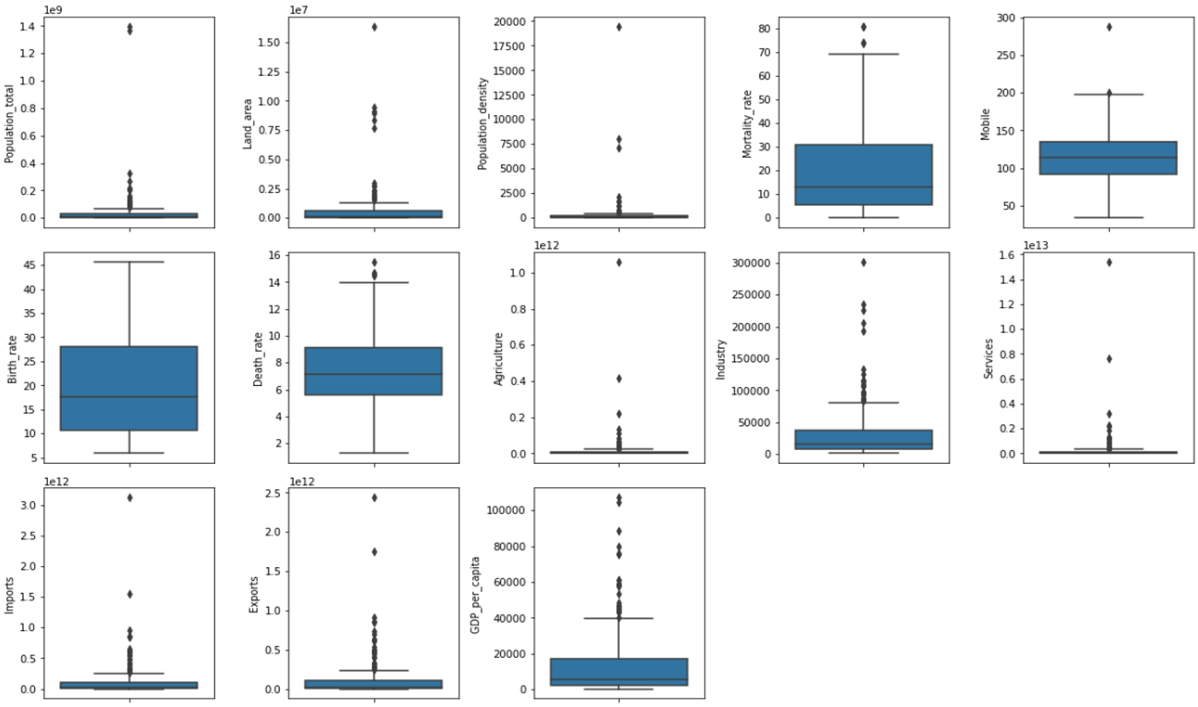
If values are missing completely at random, the data sample is likely still representative of the population. But if the values are missing systematically, analysis may be biased. Therefore, dealing with missing values are important.

There are total **110** missing values in the data

Here we first of all checked if there are any outliers or not by using **boxplot technique.**

Box plot is a type of chart often used in explanatory data analysis. Box plots visually shows the distribution of numerical data and skewness through displaying the data quartiles (or percentiles) and averages and it easily detects outliers.





Clearly from Box plot

Except Birth rate all remaining quantitative variables are having outliers.

Mortality rate, Mobile and death rate has low number of outliers.

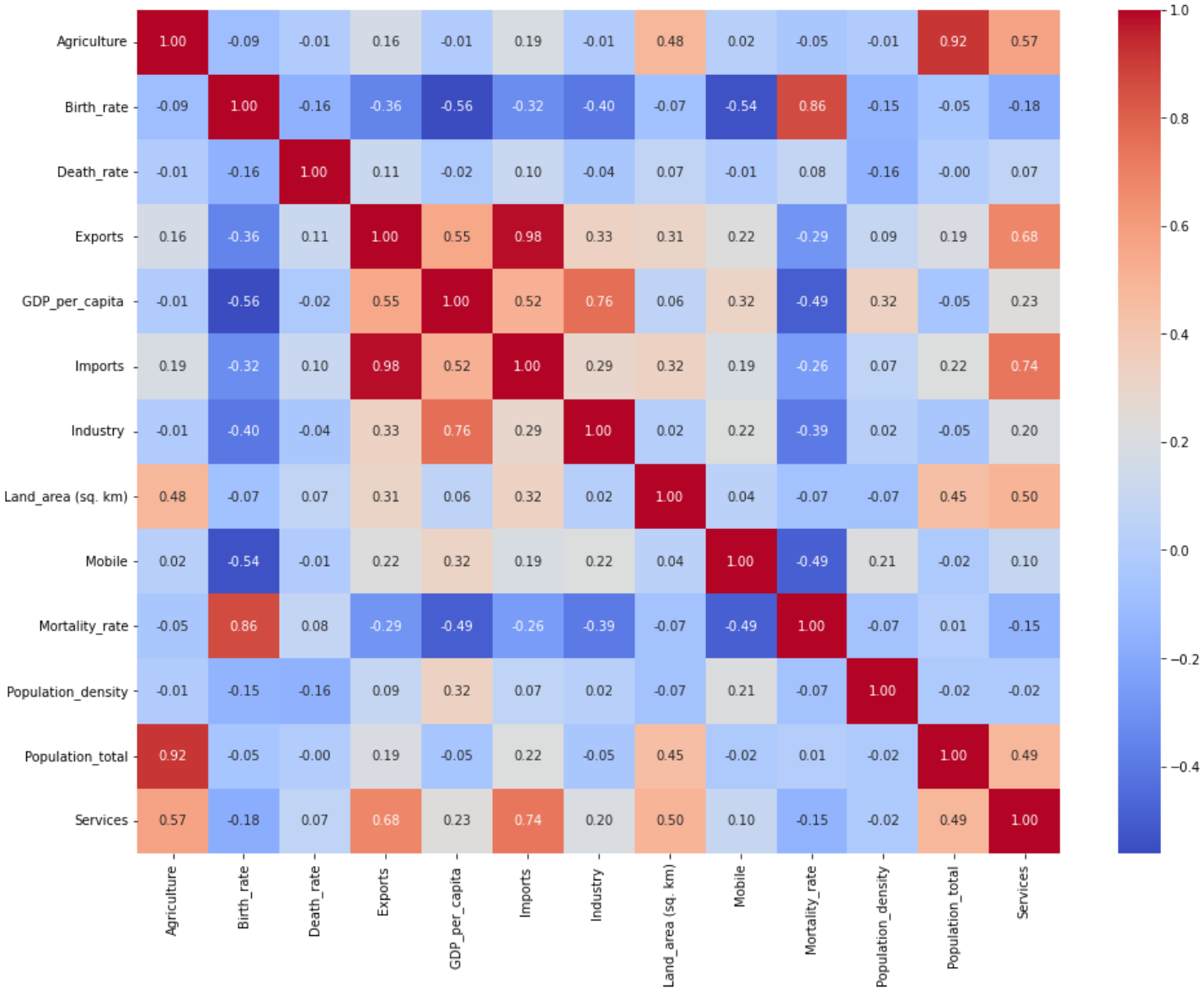
Population total, land area, population density, agriculture, industry, services, imports, exports, GDP per capita has significant number of outliers.

Therefore, the variables containing outliers are filled with **median** on the basis of the country region.

Graphical Representation

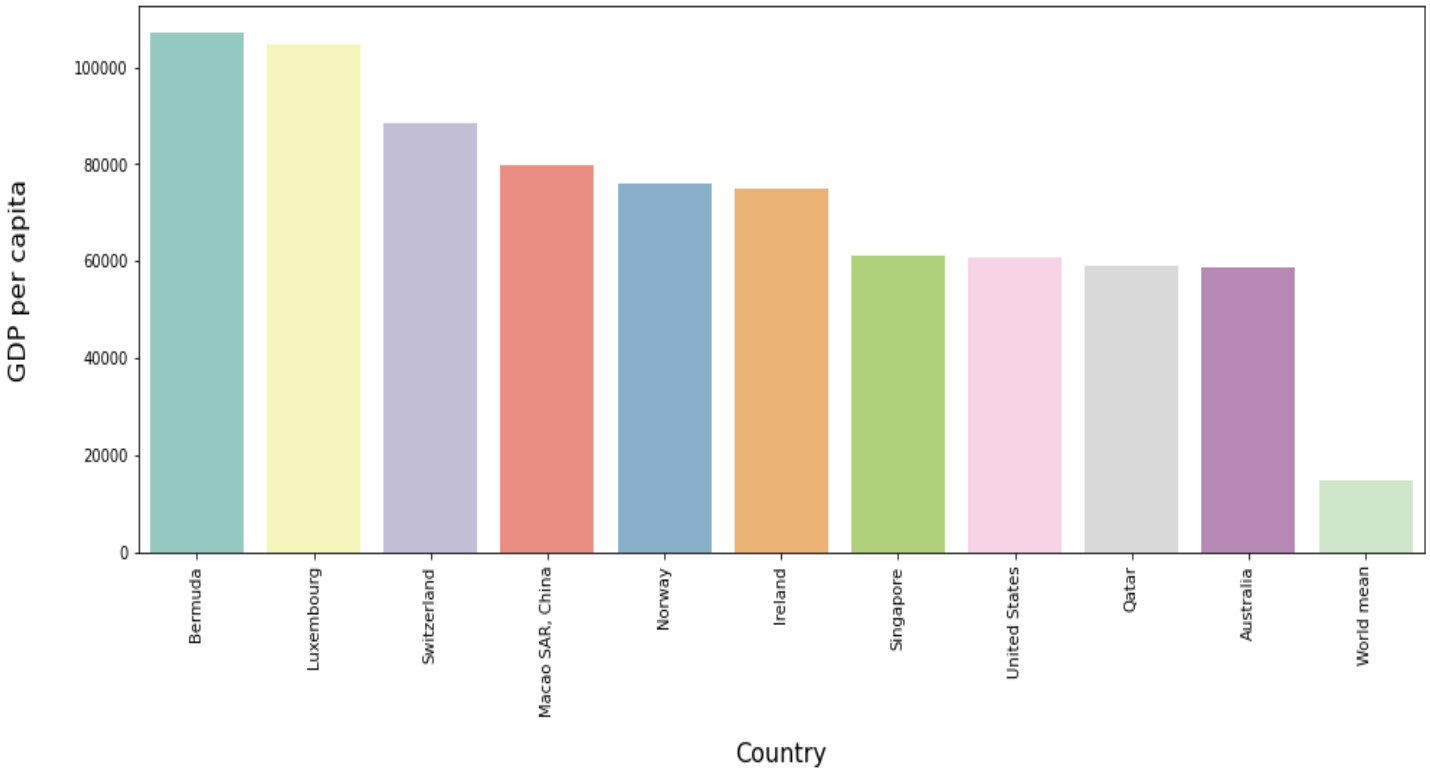
The main purpose of graphical representation is to readily give some idea about the entire data and draw instant conclusions.

❖ Correlation Matrix



Interpretation: This heatmap / correlation matrix shows the correlation between each other variables. And we can also see that various factors that are affecting (cause to increase/decrease) to GDP.

❖ Top 10 countries having highest GDP per capita

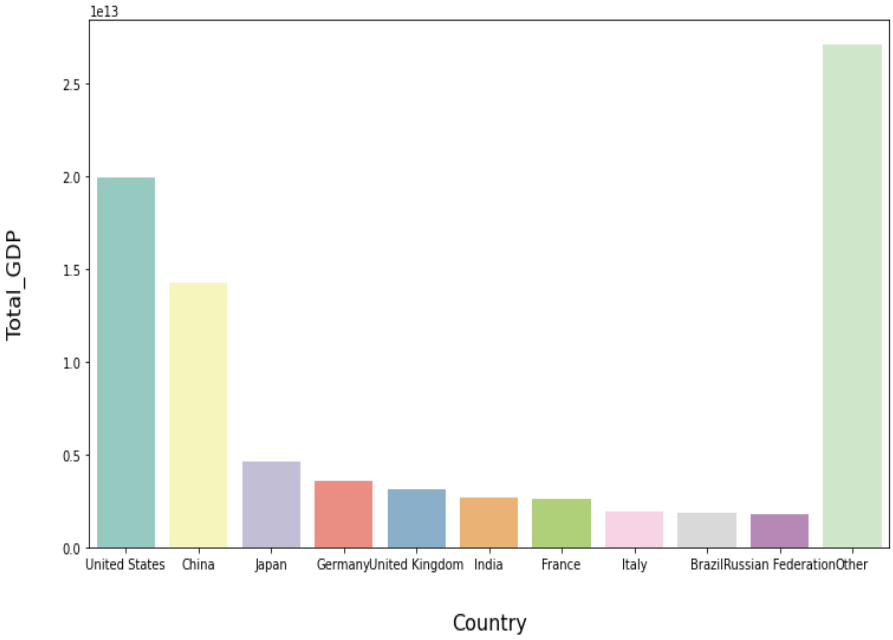


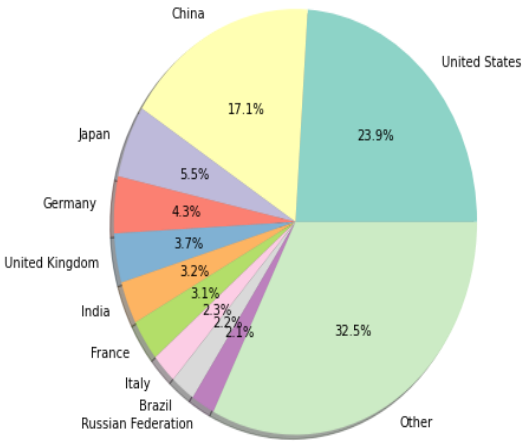
Interpretation:

From the above Bar plot, we get top 10 countries having highest GDP per Capita.

'**Bermuda**' have highest GDP per Capita of **107196.8**

❖ Top 10 countries with highest Total GDP



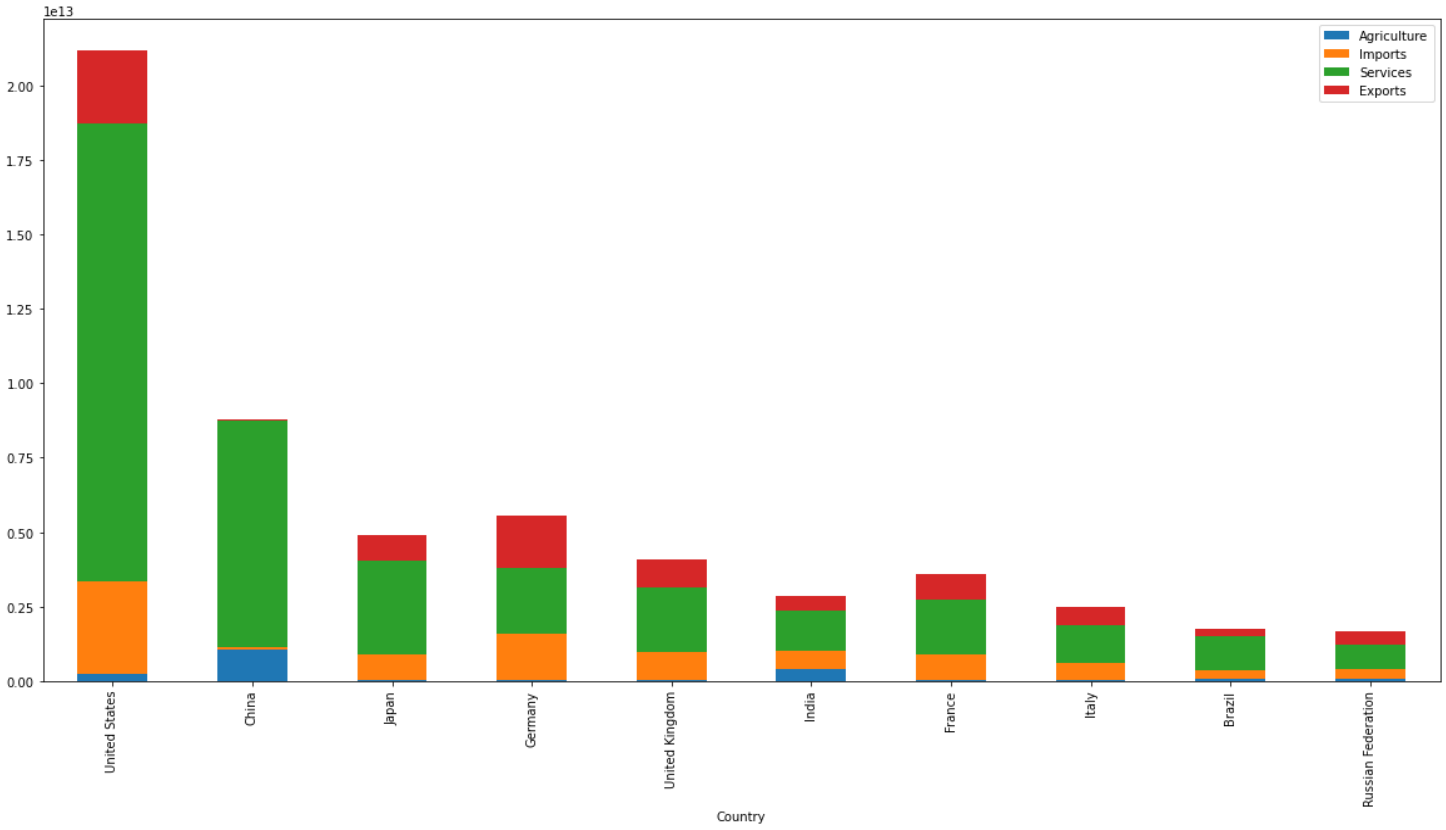


Interpretation:

Above Bar chart and pie chart shows the top 10 countries having highest total GDP

'**United States**' have highest Total GDP amongst all.

❖ Economy structure

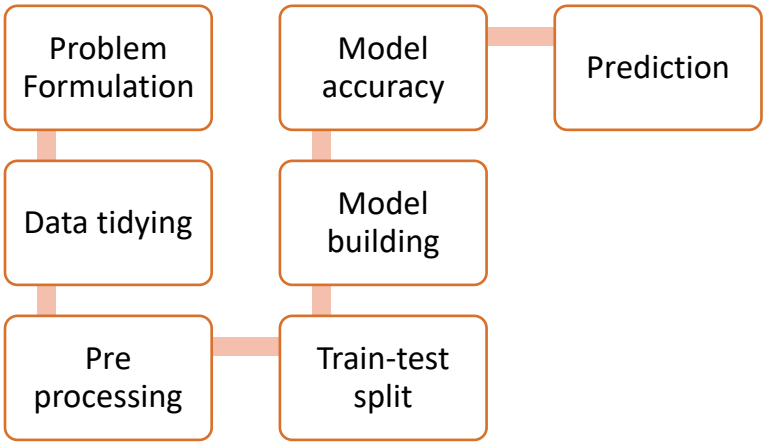


Interpretation:

Above Bar chart shows the Economy Structure of Top 10 countries having high Total GDP

Model Building

Steps involved in building machine learning model



Problem Formulation:

Convert your business problem into a statistical problem. Clearly define the dependent & independent variables. Identify whether you want to predict or infer.

Data Tidying:

Transform collected data into a usable data table format.

Preprocessing:

It includes,

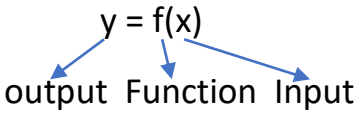
→ Filter Data
→ Missing Value Treatment
→ Variable Transformation
→ Aggregate Data
→ Outlier Treatment
→ Variable Reduction

Train – Test Split:

Training data is the information used to train an algorithm and it includes both input & corresponding output data. Based on this data (training data) the algorithm can learn the relationship between the input & output.

Testing Data includes only input data, not the expected output. It is used to asses the accuracy of model created or the predictor function created using the training data.

Model Building:



Model Accuracy & validation:

In sample error – Error resulted from applying your prediction algorithm to the dataset you built it with.

out of the sample error – Error resulted from applying your prediction algorithm to new dataset.

R^2 is a statistical metric that is used to measure how much of the variation in outcome can be explained by the variation in the independent variables. R^2 always increases as more predictors are added to the MLR model, even though the predictors may not be related to the outcome variable.

Prediction:

Setup a pipeline to use your model in real life scenario. Improve by monitoring your model over time. Try to automate.

Multiple Linear Regression

Multiple linear regression (MLR), also known simply as multiple regression, is a statistical technique that uses several explanatory variables to predict the outcome of a response variable. The goal of multiple linear regression is to model the linear relationship between the explanatory (independent) variables and response (dependent) variables. In essence, multiple regression is the extension of ordinary least-squares (OLS) regression because it involves more than one explanatory variable.

$$y_i=\beta_0+\beta_1 x_{i1}+\beta_2 x_{i2}+\ldots+\beta_p x_{ip}+\epsilon$$

where, for i = n observations:

yi = dependent variable

xi = explanatory variables

β0 = y-intercept (constant term)

βp = slope coefficients for each explanatory variable

ϵ = the model's error term (also known as the residuals)

The multiple regression model is based on the following **assumptions**:

- There is a linear relationship between the dependent variables and the independent variables
- The independent variables are not too highly correlated with each other
- yi observations are selected independently and randomly from the population
- Residuals should be normally distributed with a mean of 0 and variance σ

The coefficient of determination (R^2):

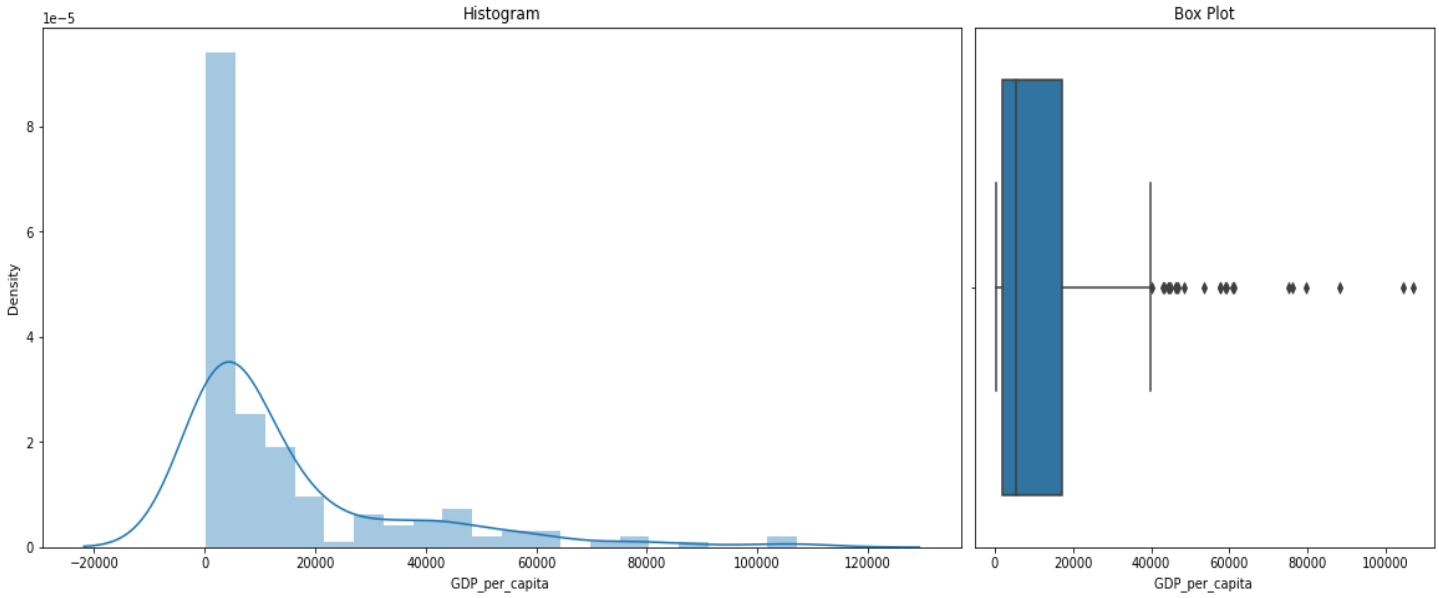
It is a statistical metric that is used to measure how much of the variation in outcome can be explained by the variation in the independent variables. R^2 always increases as more predictors are added to the MLR model, even though the predictors may not be related to the outcome variable.

R^2 by itself can't thus be used to identify which predictors should be included in a model and which should be excluded. R^2 can only be between 0 and 1, where 0 indicates that the outcome cannot be predicted by any of the independent variables and 1 indicates that the outcome can be predicted without error from the independent variables.

We first of all check the assumptions are satisfies or not.

1) **Normality**:

   Normality refers to a specific statistical distribution called a normal distribution, or sometimes the Gaussian distribution or bell-shaped curve. The normal distribution is a symmetrical continuous distribution defined by the mean and standard deviation of the data.
   Plot the histogram to see whether target variable i.e., “GDP per Capita” is following normal distribution.



- **Skewness and kurtosis of GDP per capita: -**

  Skewness: 2.139997

  Kurtosis: 4.729215

- **Observations:**

  - From histogram and boxplot, we can say that our target variable, GDP per Capita is not normally distributed.
  - Our target variable is right-skewed.
  - There are significant number of outliers in the variable
  - Therefore, either normalisation will be applied.
  - **Sometimes Normality does not bother**

2) Multicollinearity:

Multicollinearity is the occurrence of high intercorrelations among two or more independent variables in a multiple regression model. Multicollinearity can lead to skewed or misleading results when a researcher or analyst attempts to determine how well each independent variable can be used most effectively to predict or understand the dependent variable in a statistical model. In other words, multicollinearity can exist when two independent variables are highly correlated. It can also happen if an independent variable is computed from other variables in the data set or if two independent variables provide similar and repetitive results.

One of the most common ways of eliminating the problem of multicollinearity is to first identify collinear independent variables and then remove all but one.

It is also possible to eliminate multicollinearity by combining two or more collinear variables into a single variable. Statistical analysis can then be conducted to study the relationship between the specified dependent variable and only a single independent variable.

Detecting Multicollinearity

Naturally, we cannot simply assess whether or not two features have a significant relationship through intuition alone, especially when many datasets boast dozens of features.

Here are the two common evaluation metrics used for detecting multicollinearity.

1) **Correlation Coefficient**

The Pearson's correlation coefficient metric directly evaluates the strength of the relationship between two variables. Its values range between -1 and 1. The magnitude of the correlation coefficient signifies the strength of the relationship, with a higher value corresponding to a stronger relationship. By calculating the correlation coefficient between pairs of predictive features, you can identify features that may be contributing to multicollinearity.

2) **Variance Inflation Factor**

The second metric for gauging multicollinearity is the variance inflation factor (VIF). The VIF directly measures the ratio of the variance of the entire model to the variance of a model with only the feature in question.
A VIF of 1 indicates that the feature has no correlation with any of the other features.
Typically, a **VIF** value exceeding 5 or 10 is deemed to be too high. Any feature with such VIF values is likely to be contributing to multicollinearity.

From Heatmap we can observe strong relationships between,

Corr (Agriculture, Population Total) = 0.92

Corr (Birth Rate, Mortality rate) = 0.86

Corr (Exports, Imports) = 0.98

Corr (Services, Imports) = 0.74

Corr (Services, Exports) = 0.68

Eliminating this features wantonly runs the risk of removing too much important information

Computing the VIF:

So, we calculate the VIF values of all the previously identified features. The 'statsmodels' package contains a function that allows you to directly compute the VIF values of all features.

```
In [72]: vif.sort_values("VIF", ascending = False)
```

Out[72]:

	Variable	VIF
4	Imports	70.437734
3	Exports	59.097621
1	Birth rate (per 1,000 people)	14.299992
0	Agriculture	11.205196
7	Mobile (per 100 people)	9.242001
8	Mortality rate (per 1,000 live births)	9.129767
10	Population total	8.457110
2	Death rate(per 1,000 people)	7.381122
11	Services	6.283634
5	Industry	1.957504
6	Land area (sq. km)	1.693105
9	Population density (people per sq. km of land ...	1.193602

As shown in the table "Imports ", "Exports", "Birth rate (per 1,000 people)", "Agriculture"," Mobile (per 100 people)", "Mortality rate (per 1,000 live births)", "Population Total", "Death rate (per 1,000 people)", "services" all have VIF exceeding 5.

Feature selection is usually best performed by including or removing one feature at a time. This ensures that any information loss is minimized.

After removing "Imports" (i.e., the feature with the highest VIF), we calculate the VIF values again.

	Variable	VIF
1	Birth rate (per 1,000 people)	14.243791
0	Agriculture	9.868849
6	Mobile (per 100 people)	9.235874
7	Mortality rate (per 1,000 live births)	9.129521
9	Population total	7.941108
2	Death rate(per 1,000 people)	7.374979
10	Services	3.841479
3	Exports	3.376762
4	Industry	1.887009
5	Land area (sq. km)	1.678557
8	Population density (people per sq. km of land ...	1.192425

Next, remove "Birth rate (per 1,000 people)" and calculate VIF again,

	Variable	VIF
0	Agriculture	9.851983
8	Population total	7.940913
1	Death rate(per 1,000 people)	7.352868
5	Mobile (per 100 people)	6.653930
9	Services	3.832501
2	Exports	3.327945
6	Mortality rate (per 1,000 live births)	2.334186
3	Industry	1.878838
4	Land area (sq. km)	1.669761
7	Population density (people per sq. km of land ...	1.158651

Next, remove "Agriculture" and calculate VIF again,

	Variable	VIF
0	Death rate(per 1,000 people)	7.348662
4	Mobile (per 100 people)	6.631738
8	Services	2.791076
1	Exports	2.617449
5	Mortality rate (per 1,000 live births)	2.264250
2	Industry	1.874114
3	Land area (sq. km)	1.651649
7	Population total	1.598576
6	Population density (people per sq. km of land ...	1.151032

Next, remove "Death rate (per 1,000 people)" and calculate VIF again

:

	Variable	VIF
3	Mobile (per 100 people)	3.018306
7	Services	2.782918
0	Exports	2.568995
1	Industry	1.866558
4	Mortality rate (per 1,000 live births)	1.714622
2	Land area (sq. km)	1.644526
6	Population total	1.598569
5	Population density (people per sq. km of land ...	1.101616

Now that the variance inflation factors are all within the acceptable range, the derived model will be more likely to yield statistically significant results.

Before using multiple linear regression technique, we split data into Training and Testing dataset

70% for training, and 30% for testing.

After applying MLR model on training dataset we can find R^2 of test data and predicted test data is 0.676767832171

```
In [18]: r2_score(y_test, y_test_pred)
Out[18]: 0.676767832171012
```

R^2 near to 1 indicates that model is good fit

So, Multiple Linear Regression is giving around 61% accuracy.

Next, we go for Lasso Regression.

Lasso regression

Lasso regression is a regularization technique. It is used over regression methods for a more accurate prediction. This model uses shrinkage. Shrinkage is where data values are shrunk towards a central point as the mean. The lasso procedure encourages simple, sparse models (i.e., models with fewer parameters). This particular type of regression is well-suited for models showing high levels of multicollinearity or when you want to automate certain parts of model selection, like variable selection/parameter elimination. Lasso Regression uses L1 regularization technique (will be discussed later in this article). It is used when we have more features because it automatically performs feature selection.

The word "LASSO" stands for **L**east **A**bsolute **S**hrinkage and **S**election **O**perator. It is a statistical formula for the regularisation of data models and feature selection.

There are two main regularization techniques, namely Ridge Regression and Lasso Regression. They both differ in the way they assign a penalty to the coefficients.

Mathematical equation of Lasso Regression

Residual Sum of Squares + λ * (Sum of the absolute value of the magnitude of coefficients)

$$\sum_{i=1}^{n}(y_i - \sum_{j} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

Where,

λ denotes the amount of shrinkage.

λ = 0 implies all features are considered and it is equivalent to the linear regression where only the residual sum of squares is considered to build a predictive model

$\lambda = \infty$ implies no feature is considered i.e., as λ closes to infinity it eliminates more and more features

The bias increases with increase in λ

variance increases with decrease in λ

Lasso was introduced in order to improve the prediction accuracy and interpretability of regression models. It selects a reduced set of the known covariates for use in a model. Lasso was developed independently in geophysics literature in 1986, based on prior work that used the l^1 penalty for both fitting and penalization of the coefficients. Statistician Robert Tibshirani independently rediscovered and popularized it in 1996, based on Breiman's nonnegative garrote. Prior to lasso, the most widely used method for choosing covariates was stepwise selection. That approach only improves prediction accuracy in certain cases, such as when only a few covariates have a strong relationship with the outcome. However, in other cases, it can increase prediction error.

At the time, ridge regression was the most popular technique for improving prediction accuracy. Ridge regression improves prediction error by shrinking the sum of the squares of the regression coefficients to be less than a fixed value in order to reduce overfitting, but it does not perform covariate selection and therefore does not help to make the model more interpretable.

Lasso achieves both of these goals by forcing the sum of the absolute value of the regression coefficients to be less than a fixed value, which forces certain coefficients to zero, excluding them from impacting prediction. This idea is similar to ridge regression, which also shrinks the size of the coefficients, however Ridge Regression tends to set far fewer coefficients to zero.

Before using Lasso Regression Model, we split data into Training and Testing dataset

70% for training, and 30% for testing.

After applying Lasso Regression Model on training datasets, we can find R^2 of test data and predicted test data is **0.707204**

```
Lasso Regression performance
MAE: 5986.053485789598
RMSE: 9731.373687436539
R2_Score:  0.7072044590177244
```

So, Lasso Regression Model is giving around 70% accuracy

Now,

We try to fit Random Forest Model.

Random Forest Model

Random Forests are an extension of single Classification Trees in which multiple decision trees are built with random subsets of the data. All random subsets have the same number of data points, and are selected from the complete dataset. Used data is placed back in the full dataset and can be selected in subsequent trees. In Random Forests, the random subsets are selected in a procedure called 'bagging', in which each data point has an equal probability of being selected for each new random subset. Random forest algorithms have three main hyperparameters, which need to be set before training. These include node size, the number of trees, and the number of features sampled. From there, the random forest classifier can be used to solve for regression or classification problems.

The random forest algorithm is made up of a collection of decision trees, and each tree in the ensemble is comprised of a data sample drawn from a training set with replacement, called the bootstrap sample. Of that training sample, one-third of it is set aside as test data, known as the out-of-bag (oob) sample, which we'll come back to later. Another instance of randomness is then injected through feature bagging, adding more diversity to the dataset and reducing the correlation among decision trees. Depending on the type of problem, the determination of the prediction will vary. For a regression task, the individual decision trees will be averaged, and for a classification task, a majority vote—i.e. the most frequent categorical variable—will yield the predicted class. Finally, the oob sample is then used for cross-validation, finalizing that prediction.

Assumptions

No formal distributional assumptions, random forests are non-parametric and can thus handle skewed and multi-modal data as well as categorical data that are ordinal or non-ordinal.

Benefits and challenges of random forest

There are a number of key advantages and challenges that the random forest algorithm presents when used for classification or regression problems. Some of them include:

Key Benefits

- **Reduced risk of overfitting:** Decision trees run the risk of overfitting as they tend to tightly fit all the samples within training data. However, when there's a robust number of decision trees in a random forest, the classifier won't overfit the model since the averaging of uncorrelated trees lowers the overall variance and prediction error.
- **Provides flexibility:** Since random forest can handle both regression and classification tasks with a high degree of accuracy, it is a popular method among data scientists. Feature bagging also makes the random forest classifier an effective tool for estimating missing values as it maintains accuracy when a portion of the data is missing.
- **Easy to determine feature importance:** Random forest makes it easy to evaluate variable importance, or contribution, to the model. There are a few ways to evaluate feature importance. Gini importance and mean decrease in impurity (MDI) are usually used to measure how much the model's accuracy decreases when a given variable is excluded. However, permutation importance, also known as mean decrease accuracy (MDA), is another importance measure. MDA identifies the average decrease in accuracy by randomly permutating the feature values in oob samples.

Key Challenges

- **Time-consuming process**: Since random forest algorithms can handle large data sets, they can be provide more accurate predictions, but can be slow to process data as they are computing data for each individual decision tree.
- **Requires more resources**: Since random forests process larger data sets, they'll require more resources to store that data.
- **More complex**: The prediction of a single decision tree is easier to interpret when compared to a forest of them.

Advantages

- One of the most accurate learning algorithms available
- It can handle many predictors variables
- Provides estimates of the importance of different predictor variables
- Maintains accuracy even when a large proportion of the data is missing

Application

The random forest algorithm has been applied across a number of industries, allowing them to make better business decisions. Some use cases include:

- **Finance**: It is a preferred algorithm over others as it reduces time spent on data management and pre-processing tasks. It can be used to evaluate customers with high credit risk, to detect fraud, and option pricing problems.
- **Healthcare:** The random forest algorithm has applications within computational biology (link resides outside IBM) (PDF, 737 KB), allowing doctors to tackle problems such as gene expression classification, biomarker discovery, and sequence annotation. As a result, doctors can make estimates around drug responses to specific medications.
- **E-commerce**: It can be used for recommendation engines for cross-sell purposes.

Limitations

- Can overfit datasets that are particularly noisy
- For data including categorical predictor variables with different number of levels, random forests are biased in favour of those predictors with more levels. Therefore, the variable importance scores from random forest are not always reliable for this type of data

Before using technique Random

Forest Model, we split data into Training and Testing dataset

70% for training, and 30% for testing.

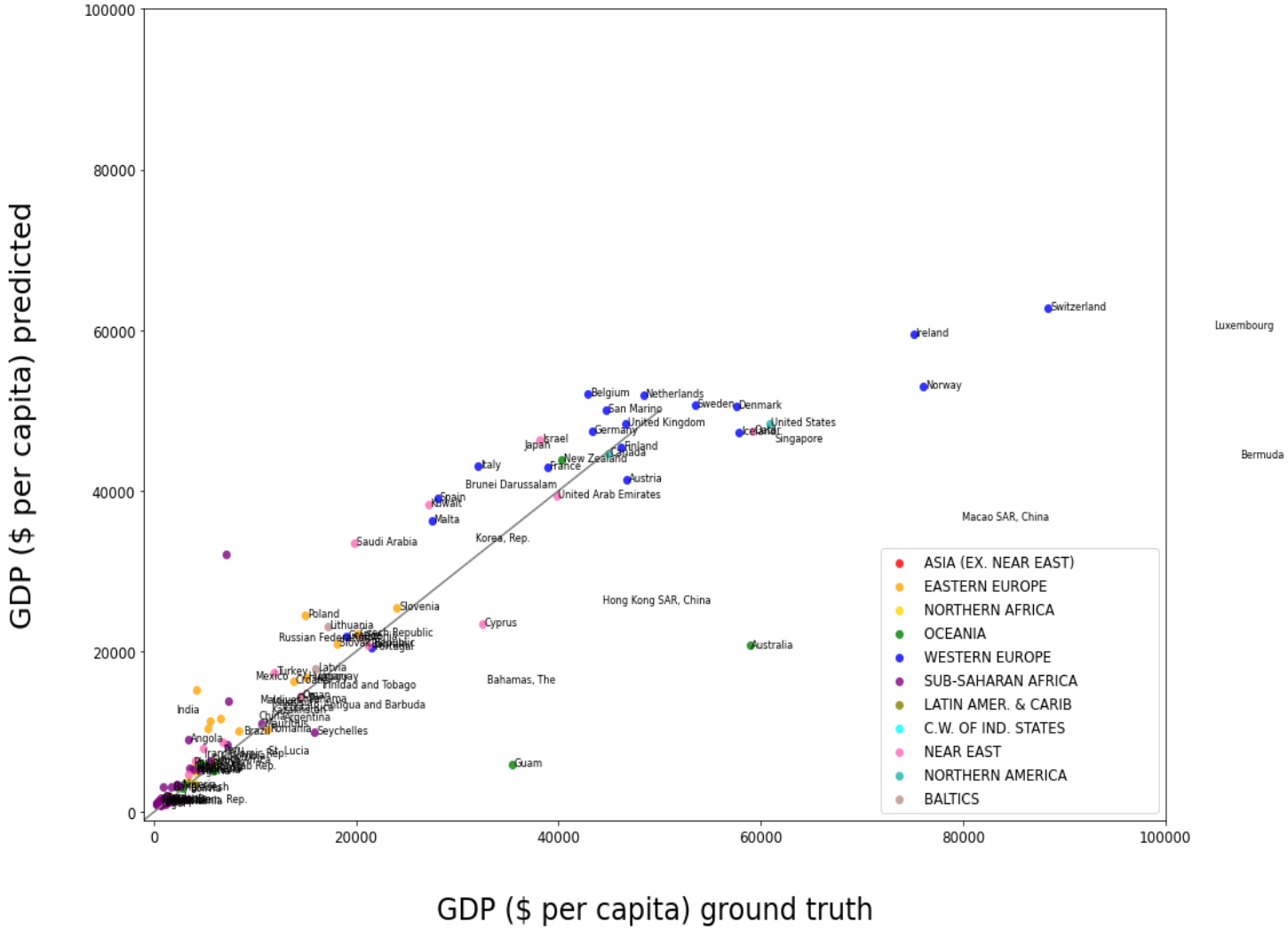
After applying Random Forest model on training dataset, we can find R^2 of test data and predicted test data is 0.8039488088

```
In [28]: r2_score(y_test,test_pred_Y)
Out[28]: 0.8039488088463822
```

R^2 near to 1 indicates that model is good fit

Here we can say that **Random Forest model is performing well**

Further, we will plot graphical representation of Random Forest model using python software



The above graphical representation shows the accuracy of actual and predicted GDP per Capita. The points near to the regression line are approximately correctly predicted. The points on the line shows the perfect prediction.

So, having the values or rough idea about the variables such as Services, Exports, Mortality rate etc., we can predict the GDP per Capita for a country. And also, focussing on these variables or working on these variables we can help to increase the GDP of our country.

Conclusions

- ✓ In objective 1, we used heat map to check correlation between Dependent and independent variable. We arrived at a conclusion that some factors affecting the most GDP per capita of a country. Those factors are imports, exports, Industry, Mortality Rate, Birth Rate and Population Density.
- ✓ In 2nd objective we used the bar graph to study how the population is affecting the ranks of the country when we calculate Total GDP and GDP per capita. And how the Imports, Services and agriculture contribute towards countries GDP. From graph we conclude that services have great impact on countries GDP followed by imports and exports.
- ✓ In our third objective we use MLR technique to check the relation between GDP per capita and other independent variable. MLR model gives us **60%** accuracy. Multicollinearity is present in our data to deal with multicollinearity we used lasso regression which gives us **70%** accuracy. Our data is skewed for that we used random forest model which gives **80%** accuracy.

Python codes

Python libraries:-

```
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 cleaning:-

```
data = pd.read_csv('world.csv',decimal=',')
print('number of missing data:')
print(data.isnull().sum())#To find out the missing values
data.describe(include='all')
fig = plt.figure(figsize=(16,30)) #To plot boxplot
features= ["population total","land area", “mortality rate", "Mobile ", “birth rate","death rate","agriculture","industry","service",”imports”,”exports”,”population density”]

for i in range(len(features)):
   fig.add_subplot(9, 5, i+1)
   sns.boxplot(y=data[features[i]])
plt.tight_layout()
plt.show()

data.groupby('Region')[['GDP ($ per capita)','Literacy (%)','Agriculture',]].median()
#Finding meadian on the basis of region.
```

Correlation matrix:-

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

Graphical representation of highest GDP per capita of top 10 countries:-

```
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()
```

Top 10 Countries with highest GDP:-

```
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 ($)'].sum() - top_gdp_countries['Total_GDP ($)'].sum()]})
gdps = pd.concat([top_gdp_countries[['Country','Total_GDP ($)']],other],ignore_index=True)
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(20,7),gridspec_kw = {'width_ratios':[2,1]})
sns.barplot(x='Country',y='Total_GDP ($)',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 ($)'], hh
axes[1].axis('equal'); plt.show()
```

Multicollinearity ;

```
X = df[["Agriculture","Birth rate (per 1,000 people)","Death rate(per 1,000 people)","Exports ","Imports ","Industry ","Land area (sq. km)","Mobile (per 100 people)","Mortality rate (per 1,000 live births)","Population density (people per sq. km of land area)","Services"]]
y = df["GDP per capita "]
from statsmodels.stats.outliers_influence import variance_inflation_factor
from statsmodels.tools.tools import add_constant
considered_features = df[["Agriculture","Birth rate (per 1,000 people)","Death rate(per 1,000 people)","Exports ","Imports ","Industry ","Land area (sq. km)","Mobile (per 100 people)","Mortality rate (per 1,000 live births)","Population density (people per sq. km of land area)","Services"]]
vif = pd.DataFrame()
```

```
vif["Variable"] = considered_features.columns
vif["VIF"] = [variance_inflation_factor(considered_features.values, i) for i in
range(considered_features.shape[1])]
vif.sort_values("VIF", ascending = False)
del considered_features["Imports "]
vif = pd.DataFrame()
vif["Variable"] = considered_features.columns
vif["VIF"] = [variance_inflation_factor(considered_features.values, i) for i in
range(considered_features.shape[1])]

vif.sort_values("VIF", ascending = False)
del considered_features["Birth rate (per 1,000 people)"]
vif = pd.DataFrame()
vif["Variable"] = considered_features.columns
vif["VIF"] = [variance_inflation_factor(considered_features.values, i) for i in
range(considered_features.shape[1])]

vif.sort_values("VIF", ascending = False)
del considered_features["Agriculture"]
vif = pd.DataFrame()
vif["Variable"] = considered_features.columns
vif["VIF"] = [variance_inflation_factor(considered_features.values, i) for i in
range(considered_features.shape[1])]

vif.sort_values("VIF", ascending = False)
del considered_features["Death rate(per 1,000 people)"]
vif = pd.DataFrame()
vif["Variable"] = considered_features.columns
vif["VIF"] = [variance_inflation_factor(considered_features.values, i) for i in
range(considered_features.shape[1])]

vif.sort_values("VIF", ascending = False)
X1 = considered_features
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X1, y, test_size = 0.3, random_state =0)
lm1 = LinearRegression()
lm1.fit(X_train, y_train)
from sklearn.metrics import r2_score
r2_score(y_test, y_test_pred)
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_squared_log_error
from sklearn.metrics import mean_absolute_error
model = RandomForestRegressor(n_estimators = 50,
```

```
                    max_depth = 6,
                    min_weight_fraction_leaf = 0.05,
                    max_features = 0.8,
                    random_state = 42)
model.fit(X_train, y_train)
train_pred_Y = model.predict(X_train)
test_pred_Y = model.predict(X_test)
train_pred_Y = pd.Series(train_pred_Y.clip(0, train_pred_Y.max()), index=y_train.index)
test_pred_Y = pd.Series(test_pred_Y.clip(0, test_pred_Y.max()), index=y_test.index)

rmse_train = np.sqrt(mean_squared_error(train_pred_Y, y_train))
msle_train = mean_squared_log_error(train_pred_Y, y_train)
rmse_test = np.sqrt(mean_squared_error(test_pred_Y, y_test))
msle_test = mean_squared_log_error(test_pred_Y, y_test)

print('rmse_train:',rmse_train,'msle_train:',msle_train)
print('rmse_test:',rmse_test,'msle_test:',msle_test)
r2_score(y_test,test_pred_Y)
d1 = df.drop(["Agriculture", "Birth rate (per 1,000 people)", "Death rate(per 1,000
people)", "Imports "], axis = 1)
plt.figure(figsize=(16,12))

train_test_Y = y_train.append(y_test)
train_test_pred_Y = train_pred_Y.append(test_pred_Y)
d1_shuffled = d1.loc[train_test_Y.index]
label = d1_shuffled['Country Name']
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[d1_shuffled['Region']==region]
   Y = train_test_pred_Y.loc[d1_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(-1000,100000)
plt.ylim(-1000,100000)

for i in range(0,train_test_Y.shape[0]):
   if((d1_shuffled["Land area (sq. km)"].iloc[i]>8e5) |
      (d1_shuffled["Population  total"].iloc[i]>1e8) |
      (d1_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')
Lasso  Regression
print("Lasso Regression performance")
model_lasso = Lasso(alpha=0.01,tol=0.01)
model_lasso.fit(X_train1,y_train1)
pred_test_lasso= model_lasso.predict(X_test1)
print('MAE:', mean_absolute_error(y_test1,pred_test_lasso))
print('RMSE:', np.sqrt(mean_squared_error(y_test1,pred_test_lasso)))
print('R2_Score: ', r2_score(y_test1,pred_test_lasso))
```

Thank You