GDP ANALYSIS

A PROJECT REPORT

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

BACHELOR OF TECHNOLOGY

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

Under the guidance of

MAHENDRA DATTA

BY

AYUSH ROHAN

JIS COLLEGE OF ENGINEERING, KALYANI

In association with

(ISO9001:2015)

SDF Building, Module #132, Ground Floor, Salt Lake City, GP Block, Sector V, Kolkata, West Bengal 700091

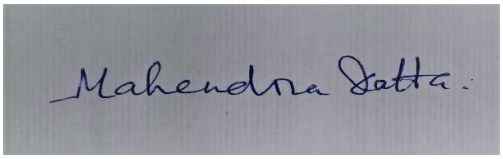
(Note: All entries of the proforma of approval should be filled up with appropriate and complete information. Incomplete proforma of approval in any respect will be summarily rejected.)

1. Title of the Project: ***GDP ANALYSIS***
2. Project Members: **AYUSH ROHAN**

3. Name of the guide: **Mr. MAHENDRA DUTTA**
4. Address: Ardent Computech Pvt. Ltd
(An ISO 9001:2015 Certified)
SDF Building, Module #132, Ground Floor, Salt Lake City, GP Block, Sector V, Kolkata, West Bengal, 700091

Project Version Control History

Version	Primary Author	Description of Version	Date Completed
Final	AYUSH ROHAN	Project Report	24th July,2022



Signature of Team Member

Date: 24/7/2022

Signature of Approver

Date:1/8/2022

For Office Use Only

MR.MAHENDA DUTTA

Approved **Not Approved** Project Proposal Evaluator

DECLARATION

We hereby declare that the project work being presented in the project proposal entitled **"GDP ANALYSIS"** in partial fulfilment of the requirements for the award of the degree of **BACHELOR OF TECHNOLOGY** at **ARDENT COMPUTECH PVT. LTD, SALTLAKE, KOLKATA, WEST BENGAL,** is an authentic work carried out under the guidance of **MR. MAHENDRA DUTTA**. The matter embodied in this project work has not been submitted elsewhere for the award of any degree of our knowledge and belief.

Date:24/7/2022

Name of the Student: AYUSH ROHAN

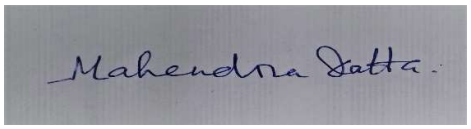
Signature of the students: 

Ardent Computech Pvt. Ltd (An ISO 9001:2015 Certified)SDF Building, Module #132, Ground Floor, Salt Lake City, GP Block, Sector V, Kolkata, West Bengal 700091

CERTIFICATE

This is to certify that this proposal of minor project entitled **"GDP ANALYSIS"** is a record of bonafide work, carried out by ***AYUSH ROHAN*** under my guidance at **ARDENT COMPUTECH PVT LTD**. In my opinion, the report in its present form is in partial fulfilment of the requirements for the award of the degree of **BACHELOR OF TECHNOLOGY** and as per regulations of the **ARDENT®.** To the best of my knowledge, the results embodied in this report, are original in nature and worthy of incorporation in the present version of the report.

Guide / Supervisor



MR. MAHENDRA DUTTA

Project Engineer

Ardent Computech Pvt. Ltd (An ISO 9001:2015 Certified)

SDF Building, Module #132, Ground Floor, Salt Lake City, GP Block, Sector V, Kolkata, West Bengal 700091

ACKNOWLEDGEMENT

Success of any project depends largely on the encouragement and guidelines of many others. I take this sincere opportunity to express my gratitude to the people who have been instrumental in the successful completion of this project work.

I would like to show our greatest appreciation to ***Mr. MAHENDRA DUTTA***, Project Engineer at Ardent, Kolkata. I always feel motivated and encouraged every time by his valuable advice and constant inspiration; without his encouragement and guidance this project would not have materialized.

Words are inadequate in offering our thanks to the other trainees, project assistants and other members at Ardent Computech Pvt. Ltd. for their encouragement and cooperation in carrying out this project work. The guidance and support received from all the members and who are contributing to this project, was vital for the success of this project.

CONTENTS

OVERVIEW

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and has fewer syntactical constructions than other languages.

Python is interpreted: Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to Perl and PHP.

Python is Interactive: You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python is Object-Oriented: Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

Python is a Beginner's Language: Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

HISTORY OF PYTHON

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands. Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, Small Talk, UNIX shell, and other scripting languages. Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL). Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

FEATURES OF PYTHON

Easy-to-learn: Python has few Keywords, simple structure and clearly defined syntax. This allows a student to pick up the language quickly.

Easy-to-Read: Python code is more clearly defined and visible to the eyes.

Easy -to-Maintain: Python's source code is fairly easy-to-maintain.

A broad standard library: Python's bulk of the library is very portable and cross platform compatible on UNIX, Windows, and Macintosh.

Interactive Mode: Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.

Portable: Python can run on the wide variety of hardware platforms and has the same interface on all platforms.

Extendable: You can add low level modules to the python interpreter. These modules enables programmers to add to or customize their tools to be more efficient.

Databases: Python provides interfaces to all major commercial databases.

GUI Programming: Python supports GUI applications that can be created and ported to many system calls, libraries, and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

Scalable: Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below:

- It support functional and structured programming methods as well as OOP.

- It can be used as a scripting language or can be compiled to byte code for building large applications.
- It provides very high level dynamic datatypes and supports dynamic type checking.
- It supports automatic garbage collections.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA and JAVA.

ENVIRONMENT SETUP

Open a terminal window and type "python" to find out if it is already installed and which version is installed.

- UNIX (Solaris, Linux, FreeBSD, AIX, HP/UX, SunOS, IRIX, etc.)
- Win 9x/NT/2000
- Macintosh (Intel, PPC, 68K)
- OS/2
- DOS (multiple versions)
- PalmOS
- Nokia mobile phones
- Windows CE
- Acorn/RISC OS

BASIC SYNTAX OF PYTHON PROGRAM

Type the following text at the Python prompt and press the Enter –

>>> print "Hello, Python!"

If you are running new version of Python, then you would need to use print statement with parenthesis as in **print ("Hello, Python!");**.
However in Python version 2.4.3, this produces the following result –

Hello, Python!

Python Identifiers

A Python identifier is a name used to identify a variable, function, class, module or other object. An identifier starts with a letter A to Z or a to z or an underscore (_) followed by zero or more letters, underscores and digits (0 to 9).
Python does not allow punctuation characters such as @, $, and % within identifiers. Python is a case sensitive programming language.

Python Keywords

The following list shows the Python keywords. These are reserved words and you cannot use them as constant or variable or any other identifier names. All the Python keywords contain lowercase letters only.

And, exec, not
Assert, finally, or
Break, for, pass
Class, from, print
continue, global, raise
def, if, return
del, import, try
elif, in, while
else, is, with
except, lambda, yield

Lines & Indentation

Python provides no braces to indicate blocks of code for class and function definitions or flow control. Blocks of code are denoted by line indentation, which is rigidly enforced. The number of spaces in the indentation is variable, but all statements within the block must be indented the same amount. For example –

```
if True:
print "True"
else:
print "False"
```

Command Line Arguments

Many programs can be run to provide you with some basic information about how they should be run. Python enables you to do this with -h −

```
$ python-h
usage: python [option]...[-c cmd|-m mod | file |-][arg]...
```

Options and arguments (and corresponding environment variables):

-c cmd: program passed in as string(terminates option list)

-d : debug output from parser (also PYTHONDEBUG=x)

-E : ignore environment variables (such as PYTHONPATH)

-h : print this help message and exit [etc.]

VARIABLE TYPES

Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.

Assigning Values to Variables

Python variables do not need explicit declaration to reserve memory space. The declaration happens automatically when you assign a value to a variable. The equal sign (=) is used to assign values to variables.

```
counter=10          # An integer assignment
weight=10.60        # A floating point
name="Ardent"       # A string
```

Multiple Assignment

Python allows you to assign a single value to several variables simultaneously. For example −

```
a = b = c = 1
a,b,c = 1,2,"hello"
```

Standard Data Types

The data stored in memory can be of many types. For example, a person's age is stored as a numeric value and his or her address is stored as alphanumeric characters. Python has five standard data types −

- String
- List
- Tuple
- Dictionary
- Number

Data Type Conversion

Sometimes, you may need to perform conversions between the built-in types. To convert between types, you simply use the type name as a function.

There are several built-in functions to perform conversion from one data type to another.

Sr.No.	Function & Description
1	**int(x [,base])** Converts x to an integer. base specifies the base if x is a string
2	**long(x [,base])** Converts x to a long integer. base specifies the base if x is a string.
3	**float(x)** Converts x to a floating-point number.
4	**complex(real [,imag])** Creates a complex number.
5	**str(x)** Converts object x to a string representation.
6	**repr(x)** Converts object x to an expression string.
7	**eval(str)** Evaluates a string and returns an object.
8	**tuple(s)** Converts s to a tuple.
9	**list(s)** Converts s to a list.

FUNCTIONS

Defining a Function

- def function name(parameters):
  "function_docstring"
  function suite
  return [expression]

Pass by reference vs Pass by value

All parameters (arguments) in the Python language are passed by reference. It means if you change what a parameter refers to within a function, the change also reflects back in the calling function. For example –

 # Function definition is here

```
def change me(mylist):
    "This changes a passed list into this function"
    mylist.append([1,2,3,4]);
    print"Values inside the function: ",mylist
    return

# Now you can call changeme function

mylist=[10,20,30];
change me(mylist);
print" Values outside the function: ",mylist
```

Here, we are maintaining reference of the passed object and appending values in the same object. So, this would produce the following result −

```
Values inside the function: [10, 20, 30, [1, 2, 3, 4]]
Values outside the function: [10, 20, 30, [1, 2, 3, 4]]
```

Global vs. Local variables

Variables that are defined inside a function body have a local scope, and those defined outside have a global scope . For Example-

```
total=0;                # This is global variable.
```

```
# Function definition is here

def sum( arg1, arg2 ):

# Add both the parameters and return them."

total= arg1 + arg2;       # Here total is local variable.
print"Inside the function local total: ", total
return total;

# Now you can call sum function

sum(10,20);
Print”Outside the function global total: ", total
```

When the above code is executed, it produces the following result −

```
Inside the function local total: 30
Outside the function global total: 0
```

MODULES

A module allows you to logically organize your Python code. Grouping related code into a module makes the code easier to understand and use. A module is a Python object with arbitrarily named attributes that you can bind and reference.

The Python code for a module named *aname* normally resides in a file named *aname.py*. Here's an example of a simple module, support.py

```
def print_func( par ):
        print"Hello : ", par
        return
```

The *import* Statement

You can use any Python source file as a module by executing an import statement in some other Python source file. The *import* has the following syntax –

```
Import module1 [, module2 [… moduleN]
```

PACKAGES

A package is a hierarchical file directory structure that defines a single Python application environment that consists of modules and sub packages and sub-subpackages, and so on.

Consider a file *Pots.py* available in *Phone* directory. This file has following line of source code −

```
def Pots ():
        print "I'm Pots Phone"
```

Similar way, we have another two files having different functions with the same name as above –

- *Phone/Isdn.py* file having function Isdn ()
- *Phone/G3.py* file having function G3 ()

Now, create one more file __init__.py in *Phone* directory −

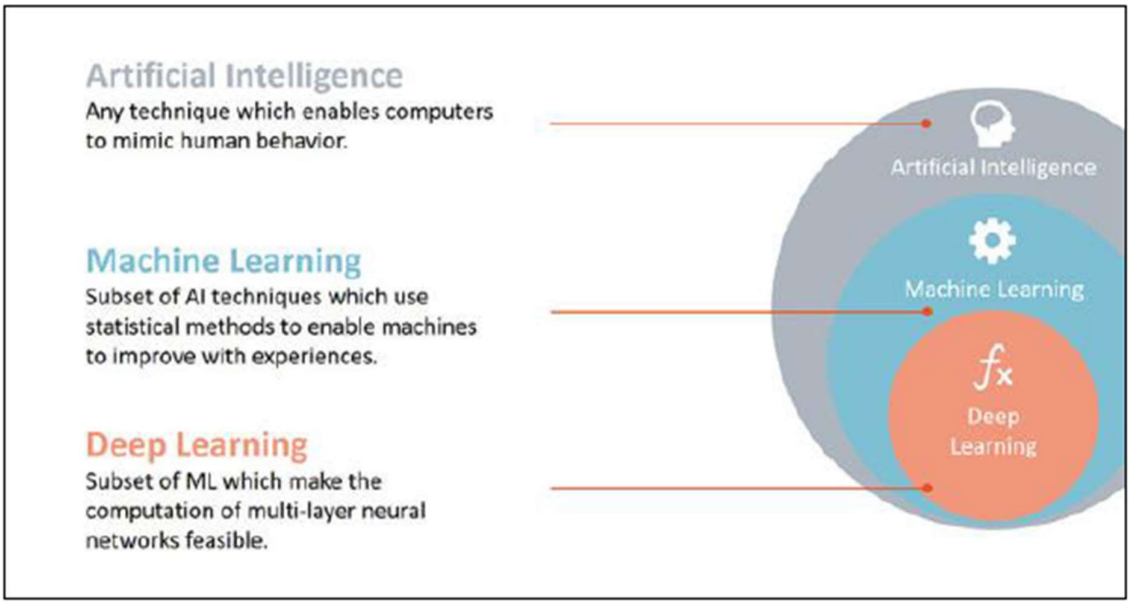
- Phone/__init__.py

To make all of your functions available when you've imported Phone, you need to put explicit import statements in __init__.py as follows −

```
from Pots import Pots
from Isdn import Isdn
from G3 import
```

ARTIFICIAL INTELLIGENCE

Introduction



According to the father of Artificial Intelligence, John McCarthy, it is *"The science and engineering of making intelligent machines, especially intelligent computer programs".*

Artificial Intelligence is a way of **making a computer, a computer-controlled robot, or a software think intelligently**, in the similar manner the intelligent humans think.

AI is accomplished by studying how human brain thinks, and how humans learn, decide, and work while trying to solve a problem, and then using the outcomes of this study as a basis of developing intelligent software and systems.

The development of AI started with the intention of creating similar intelligence in machines that we find and regard high in humans.

Goals of AI

To Create Expert Systems − The systems which exhibit intelligent behaviour, learn, demonstrate, explain, and advice its users.
To Implement Human Intelligence in Machines − Creating systems that understand, think, learn, and behave like humans.

Applications of AI

AI has been dominant in various fields such as:-

Gaming − AI plays crucial role in strategic games such as chess, poker, tic-tac-toe, etc., where machine can think of large number of possible positions based on heuristic knowledge.

Natural Language Processing − It is possible to interact with the computer that understands natural language spoken by humans.

Expert Systems − There are some applications which integrate machine, software, and special information to impart reasoning and advising. They provide explanation and advice to the users.

Vision Systems − These systems understand, interpret, and comprehend visual input on the computer.

For example: A spying aeroplane takes photographs, which are used to figure out spatial information

Or map of the areas.

Doctors use clinical expert system to diagnose the patient.

Police use computer software that can recognize the face of criminal with the stored

portrait made by forensic artist.

Speech Recognition − Some intelligent systems are capable of hearing and comprehending the language in terms of sentences and their meanings while a human talks to it. It can handle different accents, slang words, noise in the background, change in human's noise due to cold, etc.

Handwriting Recognition − The handwriting recognition software reads the text written on paper by a pen or on screen by a stylus. It can recognize the shapes of the letters and convert it into editable text.

Intelligent Robots − Robots are able to perform the tasks given by a human. They have sensors to detect physical data from the real world such as light, heat, temperature, movement, sound, bump, and pressure. They have efficient processors, multiple sensors and huge memory, to exhibit intelligence. In addition, they are capable of learning from their mistakes and they can adapt to the new environment.

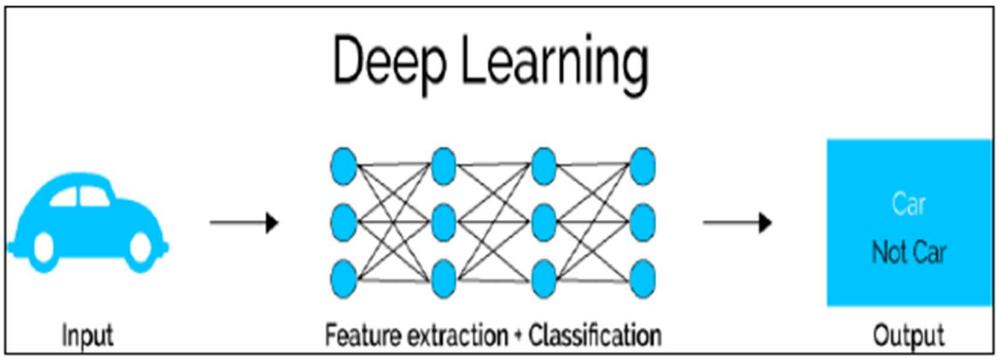
Application of AI

Deep Learning

Deep learning is a subset of machine learning. Usually, when people use the term deep learning, they are referring to deep artificial neural networks, and somewhat less frequently to deep reinforcement learning.

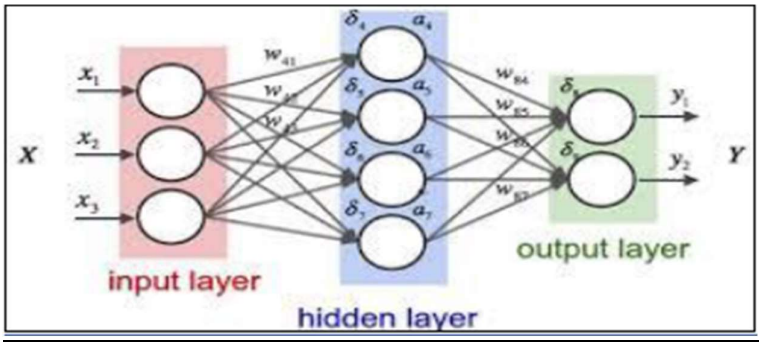
Deep learning is a class of machine learning algorithms that:

- Use a cascade of multiple layers of nonlinear processing units for feature extraction and transformation. Each successive layer uses the output from the previous layer as input.
- Learn in supervised (e.g., classification) and/or unsupervised (e.g., pattern analysis) manners.
- Learn multiple levels of representations that correspond to different levels of abstraction; the levels form a hierarchy of concepts.
- Use some form of gradient descent for training via backpropagation.



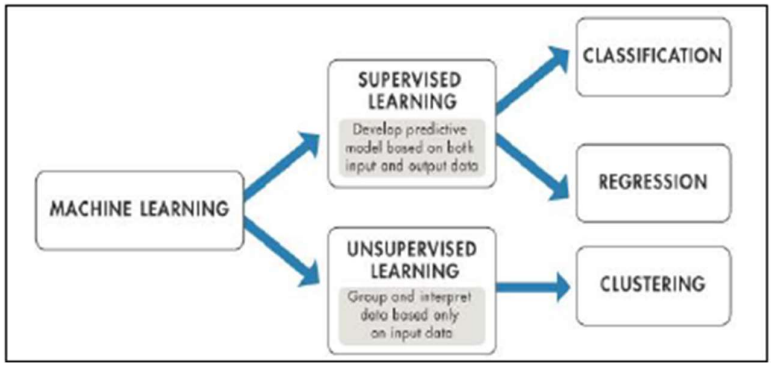
NEURAL NETWORKING

Artificial neural networks (**ANNs**) or **connectionist systems** are computing systems inspired by the biological neural networks that constitute animal brains. Such systems learn (progressively improve performance on) tasks by considering examples, generally without task-specific programming



An ANN is based on a collection of connected units or nodes called artificial neurons (analogous to biological neurons in an animal brain). Each connection between artificial neurons can transmit a signal from one to another.

MACHINE LEARNING



Machine learning is a field of computer science that gives computers the ability to learn without being explicitly programmed.
Evolved from the study of pattern recognition and computational learning theory in artificial intelligence, machine learning explores the study and construction of algorithms that can learn from and make predictions on data.

INTRODUCTION TO MACHINE LEARNING

Machine learning is a field of computer science that gives computers the ability to learn without being explicitly programmed.

Arthur Samuel, an American pioneer in the field of computer gaming and artificial intelligence, coined the term "Machine Learning" in 1959 while at IBM. Evolved from the study of pattern recognition and computational learning theory in artificial intelligence, machine learning explores the study and construction of algorithms that can learn from and make predictions on data

Machine learning tasks are typically classified into two broad categories, depending on whether there is a learning "signal" or "feedback" available to a learning system:-

SUPERVISED LEARNING

Supervised learning is the machine learning task of inferring a function from *labelled training data*.[1] The training data consist of a set of *training examples*. In supervised learning, each example is a *pair* consisting of an input object (typically a vector) and a desired output value.

A supervised learning algorithm analyses the training data and produces an inferred function, which can be used for mapping new examples. An optimal scenario will allow for the algorithm to correctly determine the class labels for unseen instances. This requires the learning algorithm to generalize from the training data to unseen situations in a "reasonable" way.

UNSUPERVISED LEARNING

Unsupervised learning is the machine learning task of inferring a function to describe hidden structure from "unlabelled" data (a classification or categorization is not included in the observations). Since the examples given to the learner are unlabelled, there is no evaluation of the accuracy of the structure that is output by the relevant algorithm—which is one way of distinguishing unsupervised learning from supervised learning and reinforcement learning.

A central case of unsupervised learning is the problem of density estimation in statistics, though unsupervised learning encompasses many other problems (and solutions) involving summarizing and explaining key features of the data.

NUMPY

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. The ancestor of NumPy, Numeric, was originally created by Jim Hugunin.

NumPy targets the CPython reference implementation of Python, which is a non-optimizing bytecode interpreter. Mathematical algorithms written for this version of Python often run much slower than compiled equivalents.

Using NumPy in Python gives functionality comparable to MATLAB since they are both interpreted, and they both allow the user to write fast programs as long as most operations work on arrays or matrices instead of scalars.

NUMPY ARRAY

NumPy's main object is the homogeneous multidimensional array. It is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers. In NumPy dimensions are called *axes*. The number of axes is *rank*.

For example, the coordinates of a point in 3D space [1, 2, 1] is an array of rank 1, because it has one axis. That axis has a length of 3. In the example pictured below, the array has rank 2 (it is 2-dimensional). The first dimension (axis) has a length of 2, the second dimension has a length of 3.

[[1., 0., 0.],
 [0., 1., 2.]]

NumPy's array class is called *ndarray*. It is also known by the alias.

SLICING NUMPY ARRAY

Import numpy as np

a = np.array ([[1, 2, 3],[3,4,5],[4,5,6]])

print 'Our array is:'
Print a
print '\n'

print 'The items in the second column are:'
print a[...,1]
print '\n'

print 'The items in the second row are:'
print a[1...]
print '\n'

print 'The items columns 1 onwards are:'
print a [...,1:]
OUTPUT

Our array is:
[[1 2 3]
[3 4 5]
[4 5 6]]

The items in the second column are:
[2 4 5]

The items in the second row are:
[3 4 5]

The items column 1 onwards are:
[[2 3]

[4 5]
[5 6]]

SCIPY

modules for optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing, ODE solvers and other tasks common in science and engineering.

SciPy builds on the NumPy array object and is part of the NumPy stack which includes tools like Matplotlib, pandas and SymPy, and an expanding set of scientific computing libraries. This NumPy stack has similar users to other applications such as MATLAB, GNU Octave, and Scilab. The NumPy stack is also sometimes referred to as the SciPy stack.

The SciPy Library/Package

The SciPy package of key algorithms and functions core to Python's scientific computing capabilities. Available sub-packages include:

- **constants:** physical constants and conversion factors (since version 0.7.0)
- **cluster:** hierarchical clustering, vector quantization, K-means
- **fftpack:** Discrete Fourier Transform algorithms
- **integrate:** numerical integration routines
- **interpolate:** interpolation tools
- **io:** data input and output
- **lib:** Python wrappers to external libraries
- **linalg:** linear algebra routines
- **misc:** miscellaneous utilities (e.g. image reading/writing)
- **ndimage:** various functions for multi-dimensional image processing
- **optimize:** optimization algorithms including linear programming
- **signal:** signal processing tools
- **sparse:** sparse matrix and related algorithms
- **spatial:** KD-trees, nearest neighbours, distance functions
- **special:** special functions
- **stats:** statistical functions
- **weave:** tool for writing C/C++ code as Python multiline strings

Data Structures

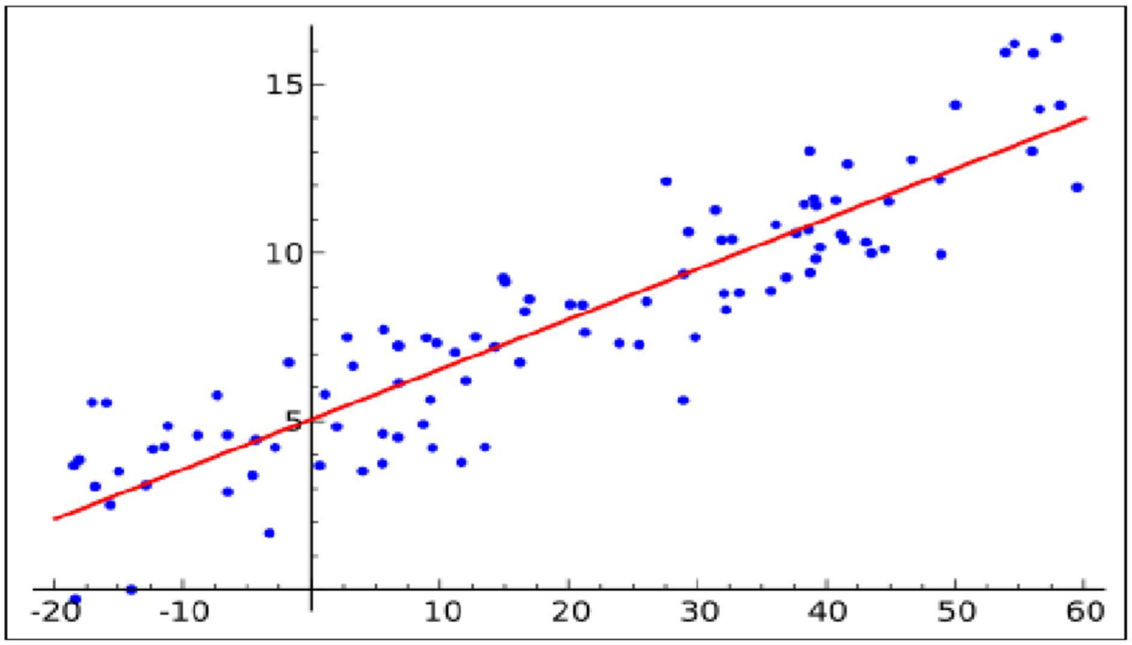
The basic data structure used by SciPy is a multidimensional array provided by the NumPy module. NumPy provides some functions for linear algebra, Fourier transforms and random number generation, but not with the generality of the equivalent functions in SciPy. NumPy can also be used as an efficient multi-dimensional container of data with arbitrary data-types. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases. Older versions of SciPy used Numeric as an array type, which is now deprecated in favour of the newer NumPy array code.

SCIKIT-LEARN

Scikit-learn is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, *k*-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

The scikit-learn project started as scikits.learn, a Google Summer of Code project by David Cournapeau. Its name stems from the notion that it is a "SciKit" (SciPy Toolkit), a separately-developed and distributed third-party extension to SciPy.[4] The original codebase was later rewritten by other developers. In 2010 Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort and Vincent Michel, all from INRIA took leadership of the project and made the first public release on February the 1st 2010[5]. Of the various scikits, scikit-learn as well as scikit-image were described as "well-maintained and popular" in November 2012.

REGRESSION ANALYSIS



In statistical modelling, **regression analysis** is a set of statistical processes for estimating the relationships among variables. It includes many techniques for modelling and analysing several variables, when the focus is on the relationship between a dependent variable and one or more independent variables (or 'predictors'). More specifically, regression analysis helps one understand how the typical value of the dependent variable (or 'criterion variable') changes when any one of the independent variables is varied, while the other independent variables are held fixed.

Regression analysis is widely used for prediction and forecasting, where its use has substantial overlap with the field of machine learning. Regression analysis is also used to understand which among the independent variables are related to the dependent variable, and to explore the forms of these relationships. In restricted circumstances, regression analysis can be used to infer casual relationships between the independent and dependent variables. However this can lead to illusions or false relationships, so caution is advisable

LINEAR REGRESSION

Linear regression is a linear approach for modelling the relationship between a scalar dependent variable y and one or more explanatory variables (or independent variables) denoted X. The case of one explanatory variable is called *simple linear regression*. For more than one explanatory variable, the process is called *multiple linear regression*.

In linear regression, the relationships are modelled using linear predictor functions whose unknown model parameters are estimated from the data. Such models are called *linear models*.

LOGISTIC REGRESSION

Logistic regression, or logit regression, or logit model [1] is a regression model where the dependent variable (DV) is categorical. This article covers the case of a binary dependent variable—that is, where the output can take only two values, "0" and "1", which represent outcomes such as pass/fail, win/lose, alive/dead or healthy/sick. Cases where the dependent variable has more than two outcome categories may be analysed in multinomial logistic regression, or, if the multiple categories are ordered, in ordinal logistic regression. In the terminology of economics, logistic regression is an example of a qualitative response/discrete choice model.

POLYNOMIAL REGRESSION

Polynomial regression is a form of regression analysis in which the relationship between the independent variable x and the dependent variable y is modelled as an n^{th} degree polynomial in x.

Polynomial regression fits a nonlinear relationship between the value of x and the corresponding conditional mean of y , denoted E(y | x), and has been used to describe nonlinear phenomena such as the growth rate of tissues, the distribution of carbon isotopes in lake sediments, and the progression of disease epidemics.

Although *polynomial regression* fits a nonlinear model to the data, as a statistical estimation problem it is linear, in the sense that the regression function E(y | x) is linear in the unknown parameters that are estimated from the data.

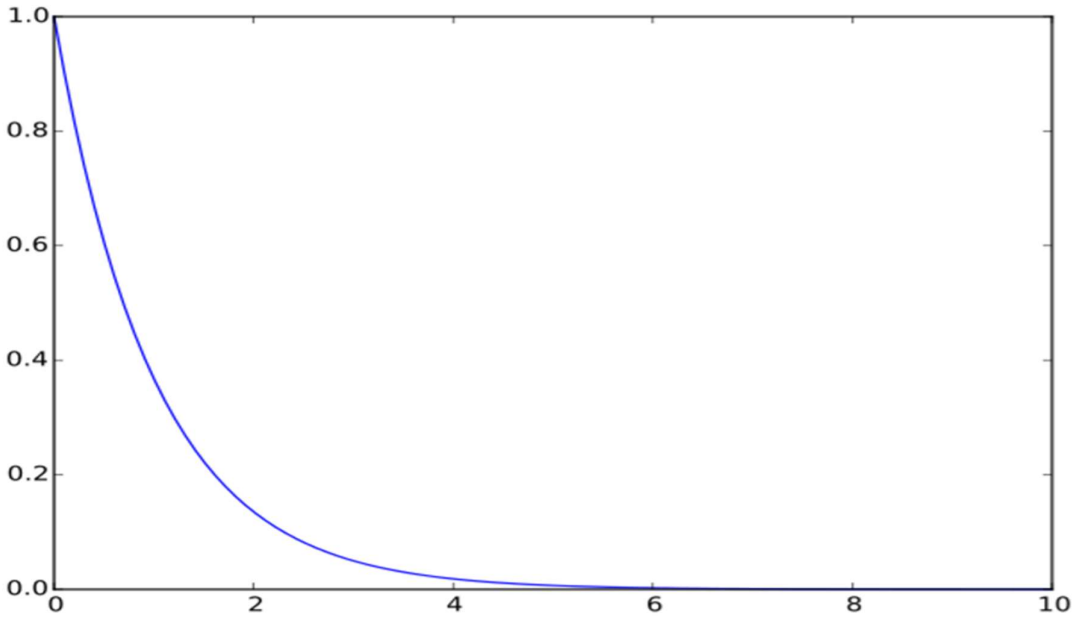
MATPLOTLIB

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter,wxPython, Qt, or GTK+. There is also a procedural "pylab" interface based on a state machine (like OpenGL), designed to closely resemble that of MATLAB, though its use is discouraged .SciPy makes use of matplotlib.

EXAMPLE

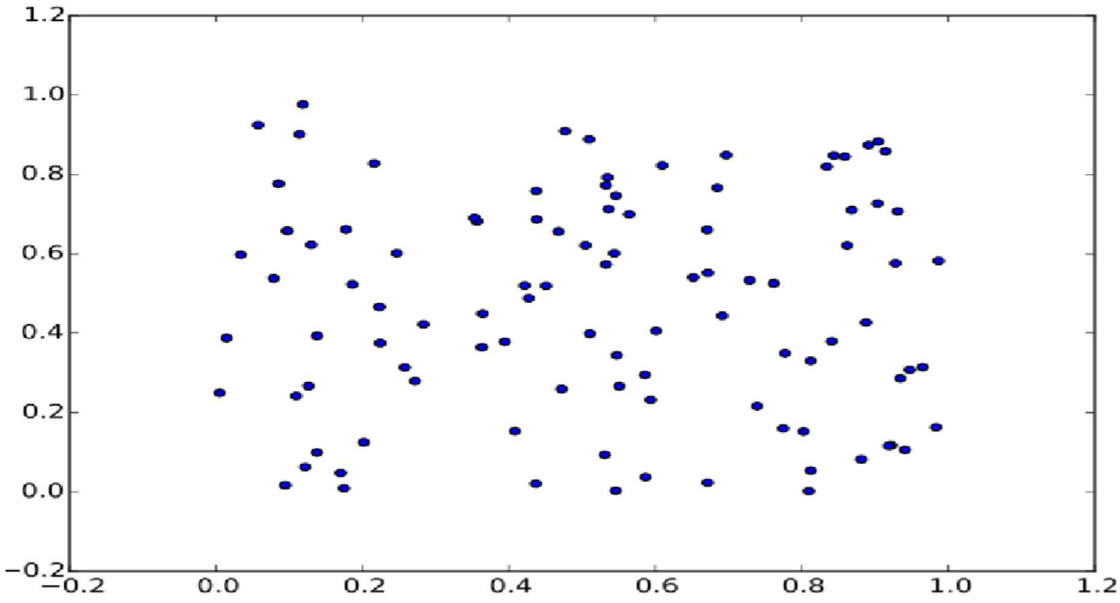
- **LINE PLOT**

```
>>>importmatplotlib.pyplotasplt
>>>importnumpyasnp
>>> a =np.linspace(0,10,100)
>>> b =np.exp(-a)
>>>plt.plot (a,b)
>>>plt.show ()
```



- **SCATTER PLOT**

```
>>>importmatplotlib.pyplotasplt
>>>fromnumpy.randomimport rand
>>> a =rand(100)
>>> b =rand(100)
>>>plt.scatter(a, b)
>>>plt.show ()
```



PANDAS

In computer programming, **pandas** is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license. "Panel data", an econometrics term for multidimensional, structured data sets.

LIBRARY FEATURES

- Data Frame object for data manipulation with integrated indexing.
- Tools for reading and writing data between in-memory data structures and different file formats.
- Data alignment and integrated handling of missing data.
- Reshaping and pivoting of data sets.
- Label-based slicing, fancy indexing, and sub setting of large data sets.
- Data structure column insertion and deletion.
- Group by engine allowing split-apply-combine operations on data sets.
- Data set merging and joining.
- Hierarchical axis indexing to work with high-dimensional data in a lower-dimensional data structure.
- Time series-functionality: Date range generation.

CLUSTERING

Cluster analysis or **clustering** is the task of grouping a set of objects in such a way that objects in the same group (called a **cluster**) are more similar (in some sense or another) to each other than to those in other groups (clusters). It is a main task of exploratory data mining, and a common technique for statistical data analysis, used in many fields, including machine learning, pattern recognition, image analysis, information retrieval, bioinformatics, data compression, and computer graphics.

Cluster analysis itself is not one specific algorithm, but the general task to be solved. It can be achieved by various algorithms that differ significantly in their notion of what constitutes a cluster and how to efficiently find them. Popular notions of clusters include groups with small distances among the cluster members, dense areas of the data space, intervals or particular statistical distributions. Clustering can therefore be formulated as a multi-objective optimization problem.

The appropriate clustering algorithm and parameter settings (including values such as the distance function to use, a density threshold or the number of expected clusters) depend on the individual data set and intended use of the results. Cluster analysis as such is not an automatic task, but an iterative process of knowledge discovery or interactive multi-objective optimization that involves trial and failure. It is often necessary to modify data pre-processing and model parameters until the result achieves the desired properties.

ALGORITHM

- Data Collection
- Data Formatting
- Model Selection
- Training
- Testing

Data Collection: We have collected data sets of weather from online website. We have downloaded the .csv files in which information was present.

Data Formatting: The collected data is formatted into suitable data sets. We check the collinearity with mean temperature. The data sets which have collinearity nearer to 1.0 has been selected.

Model Selection: We have selected different models to minimize the error of the predicted value. The different models used are Linear Regression Linear Model, Ridge Linear model, Lasso Linear Model and Bayesian Ridge Linear Model.

Training: The data set was divided such that x_train is used to train the model with corresponding x_test values and some y_train kept reserved for testing.

Testing: The model was tested with y_train and stored in y_predict. Both y_train and y_predict was compared.

Actual Codes For GDP Analysis

Introduction

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

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

Problem Statement

In this Data Science Project, I am investigating the dataset “Countries of the World”. I will be focusing on the factors affecting a country’s GDP per capita and try to make a model using the data of 227 countries from the dataset. I will also briefly discuss the total GDP.

Advantages & Disadvantages

Gross Domestic Product (GDP) is an economic measure of a nation's total income and output for a given time period (usually a year). Economists use GDP to measure the relative wealth and prosperity of different nations, as well as to measure the overall growth or decline of a nation's economy.

The most common way to measure GDP is the expenditure approach. With the expenditure approach, GDP is the sum of the following elements:

- **Total domestic consumption:** This is the total amount spent on domestically produced final goods and services. *Final goods* are items that will not be resold or used in production within the next year — milk, cars, bow ties, and so on.
- **Total domestic investment expenditures:** This measurement includes not only investments in stocks and bonds, but also investments in equipment — such as bulldozers, computer

servers, and commercial buildings — that will be useful over a long period of time. It also includes **inventory goods** — final goods waiting to be sold that a company still has on hand.
- **Government expenditures:** This includes everything from paying military salaries to building roads and maintaining monuments, but does not include welfare and social security payments.
- **Net exports:** Net exports is the total of goods and services produced domestically and sold to foreigners *minus* goods and services produced by foreigners but sold domestically (imports).

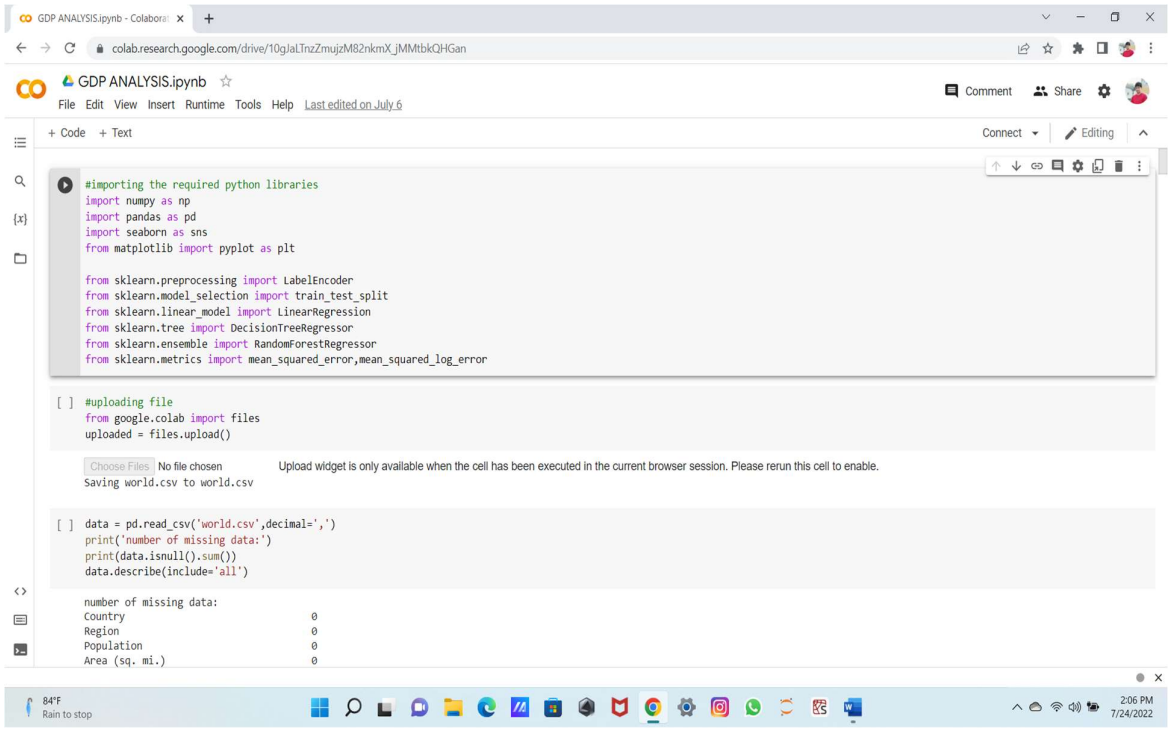
But there are a number of shortcomings to using GDP. Here are just a few:

- **GDP doesn't count unpaid volunteer work:** GDP doesn't take into account work that people do for free, from an afternoon spent picking up litter on the roadside to the millions of man-hours spent on free and open source software (such as Linux). In fact, volunteer work can actually *lower* GDP when volunteers do work that might otherwise have gone to a paid employee or contractor.
- **Disasters can raise GDP:** Wars require soldiers, oil spills require cleanup, and natural disasters require health workers, builders, and all manner of helping hands. Rebuilding after a disaster or war can greatly increase economic activity and boost GDP.
- **GDP doesn't account for quality of goods:** Consumers may buy cheap, low-quality, short-lived products repeatedly instead of buying more expensive, longer-lasting goods. Over time, consumers could spend more replacing cheap goods than they would have if they had bought higher-quality goods in the first place, and GDP would grow as a result of waste and inefficiency.

Future scope

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

Explanation :



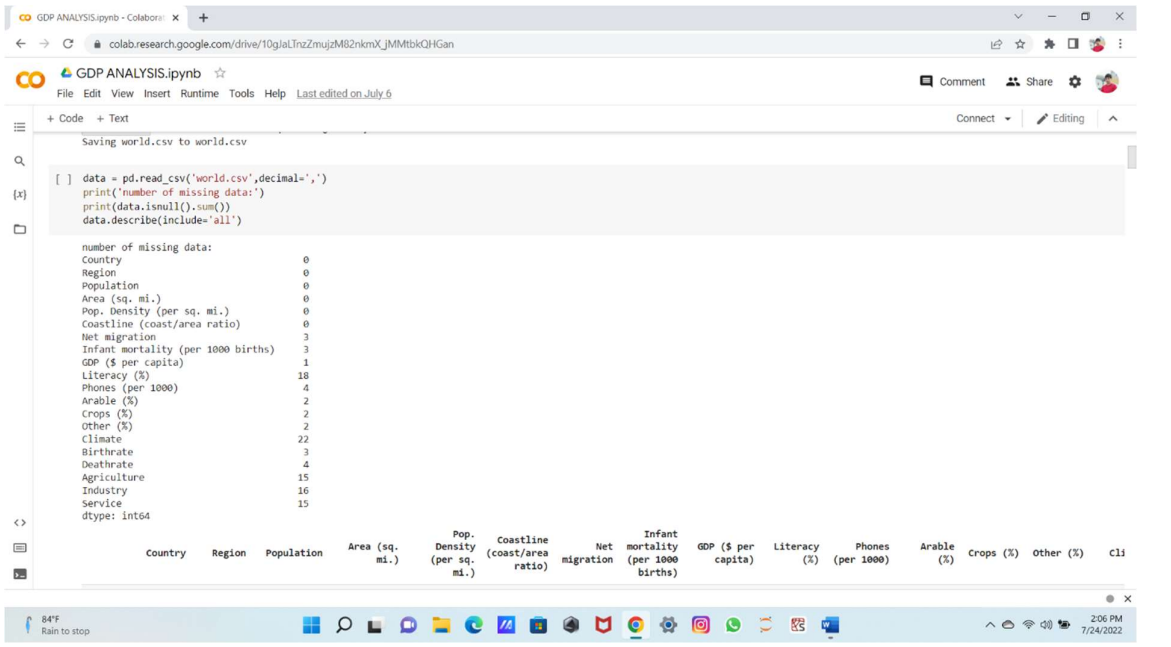
```
#importing the required python libraries
import numpy as np
import pandas as pd
import seaborn as sns
from matplotlib import pyplot as plt

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
```

```
#uploading file
from google.colab import files
uploaded = files.upload()
```

```
Choose Files  No file chosen        Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving world.csv to world.csv
```

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

```
number of missing data:
Country                             0
Region                              0
Population                          0
Area (sq. mi.)                      0
```

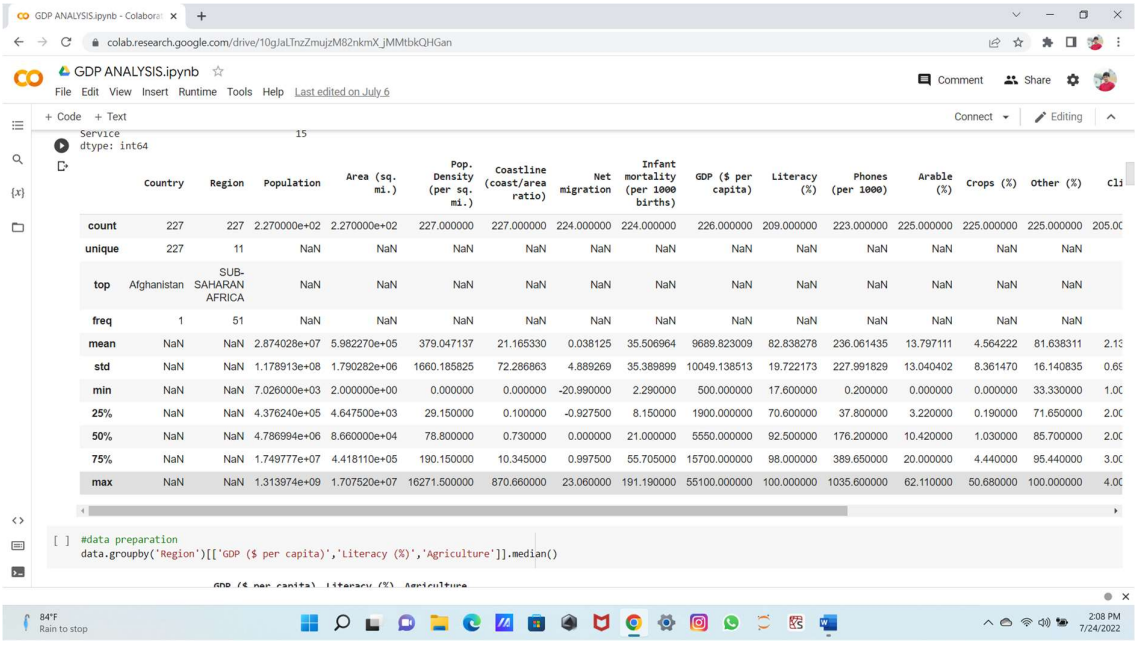


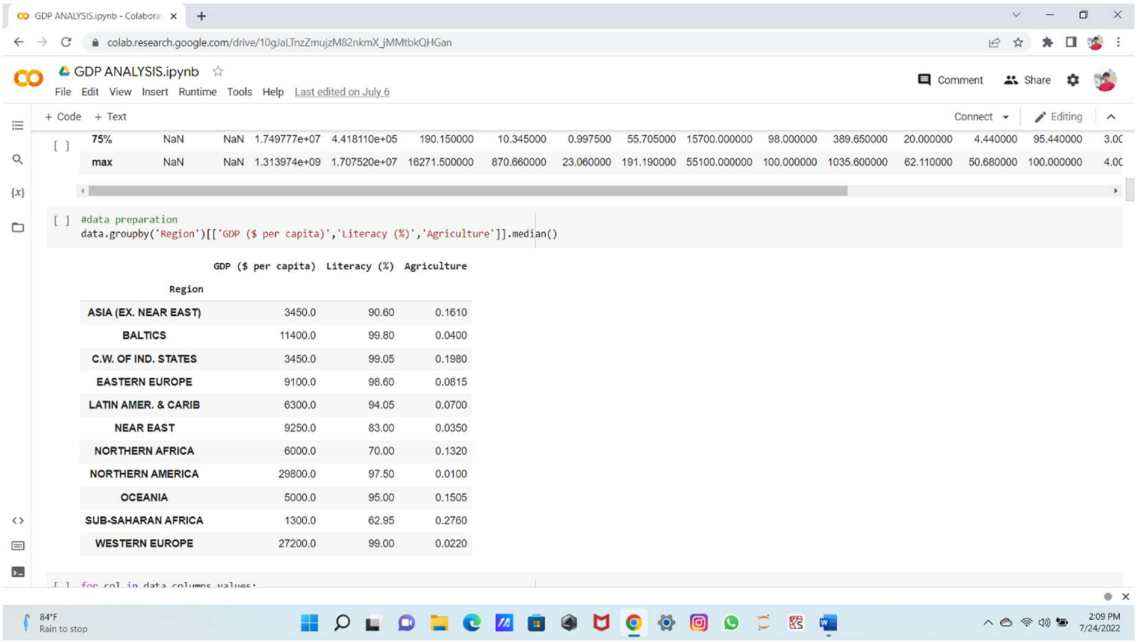
```
Saving world.csv to world.csv
```

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

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

	Country	Region	Population	Area (sq. mi.)	Pop. Density (per sq. mi.)	Coastline (coast/area ratio)	Net migration	Infant mortality (per 1000 births)	GDP ($ per capita)	Literacy (%)	Phones (per 1000)	Arable (%)	Crops (%)	Other (%)	Cli


GDP ANALYSIS.ipynb - Colaborat
colab.research.google.com/drive/10gJaLTnzZmujzM82nkmX_jMMtbkQHGan
GDP ANALYSIS.ipynb
File Edit View Insert Runtime Tools Help Last edited on July 6
Comment
Share
+ Code + Text
Connect
Editing
Service 15
dtype: int64
Country Region Population Area (sq. mi.) Pop. Density (per sq. mi.) Coastline (coast/area ratio) Net migration Infant mortality (per 1000 births) GDP ($ per capita) Literacy (%) Phones (per 1000) Arable (%) Crops (%) Other (%) Cli
count 227 227 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 205.0
unique 227 11 NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN
top Afghanistan SUB-SAHARAN AFRICA NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN
freq 1 51 NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN
mean NaN NaN 2.874028e+07 5.982270e+05 379.047137 21.165330 0.038125 35.506964 9689.823009 82.838278 236.061435 13.797111 4.564222 81.638311 2.13
std NaN NaN 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 0.69
min NaN NaN 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 1.00
25% NaN NaN 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 2.00
50% NaN NaN 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 2.00
75% NaN NaN 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 3.00
max NaN NaN 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 4.00
[ ] #data preparation
data.groupby('Region')[['GDP ($ per capita)','Literacy (%)','Agriculture']].median()
84°F
Rain to stop
2:08 PM
7/24/2022


GDP ANALYSIS.ipynb - Colaborat
colab.research.google.com/drive/10gJaLTnzZmujzM82nkmX_jMMtbkQHGan
GDP ANALYSIS.ipynb
File Edit View Insert Runtime Tools Help Last edited on July 6
Comment
Share
+ Code + Text
Connect
Editing
75% NaN NaN 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 3.00
max NaN NaN 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 4.00
[ ] #data preparation
data.groupby('Region')[['GDP ($ per capita)','Literacy (%)','Agriculture']].median()
GDP ($ per capita) Literacy (%) Agriculture
Region
ASIA (EX. NEAR EAST) 3450.0 90.60 0.1610
BALTICS 11400.0 99.80 0.0400
C.W. OF IND. STATES 3450.0 99.05 0.1980
EASTERN EUROPE 9100.0 98.60 0.0815
LATIN AMER. & CARIB 6300.0 94.05 0.0700
NEAR EAST 9250.0 83.00 0.0350
NORTHERN AFRICA 6000.0 70.00 0.1320
NORTHERN AMERICA 29800.0 97.50 0.0100
OCEANIA 5000.0 95.00 0.1505
SUB-SAHARAN AFRICA 1300.0 62.95 0.2760
WESTERN EUROPE 27200.0 99.00 0.0220
[ ] for col in data.columns.values:
84°F
Rain to stop
2:09 PM
7/24/2022

GDP ANALYSIS.ipynb - Colaborat

colab.research.google.com/drive/10gJaLTnzZmujzM82nkmX_jMMtbkQHGan

GDP ANALYSIS.ipynb

File Edit View Insert Runtime Tools Help Last edited on July 6

Comment Share

+ Code + Text

Connect Editing

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

```
/usr/local/lib/python3.7/dist-packages/pandas/core/indexing.py:1732: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self._setitem_single_block(indexer, value, name)
```

```
[ ]
    #data exploration
    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()
```

84°F Rain to stop

2:10 PM 7/24/2022

GDP ANALYSIS.ipynb - Colaborat

colab.research.google.com/drive/10gJaLTnzZmujzM82nkmX_jMMtbkQHGan

GDP ANALYSIS.ipynb

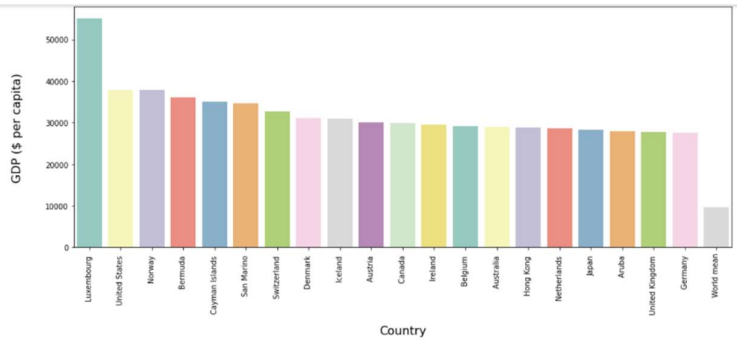
File Edit View Insert Runtime Tools Help Last edited on July 6

Comment Share

+ Code + Text

Connect Editing

```
[ ]
```

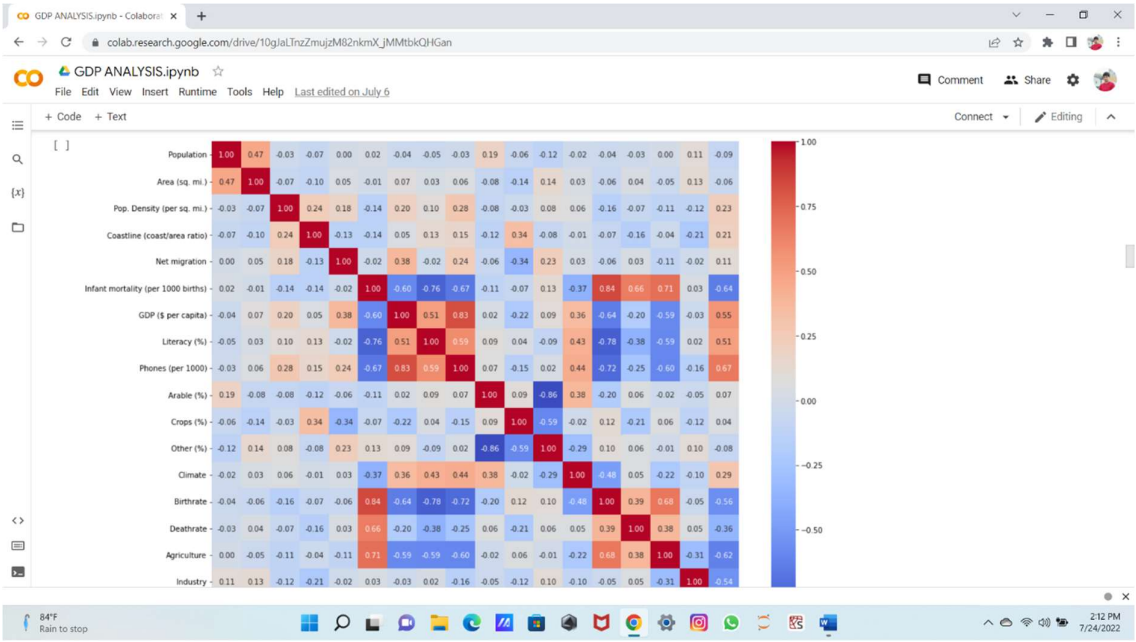


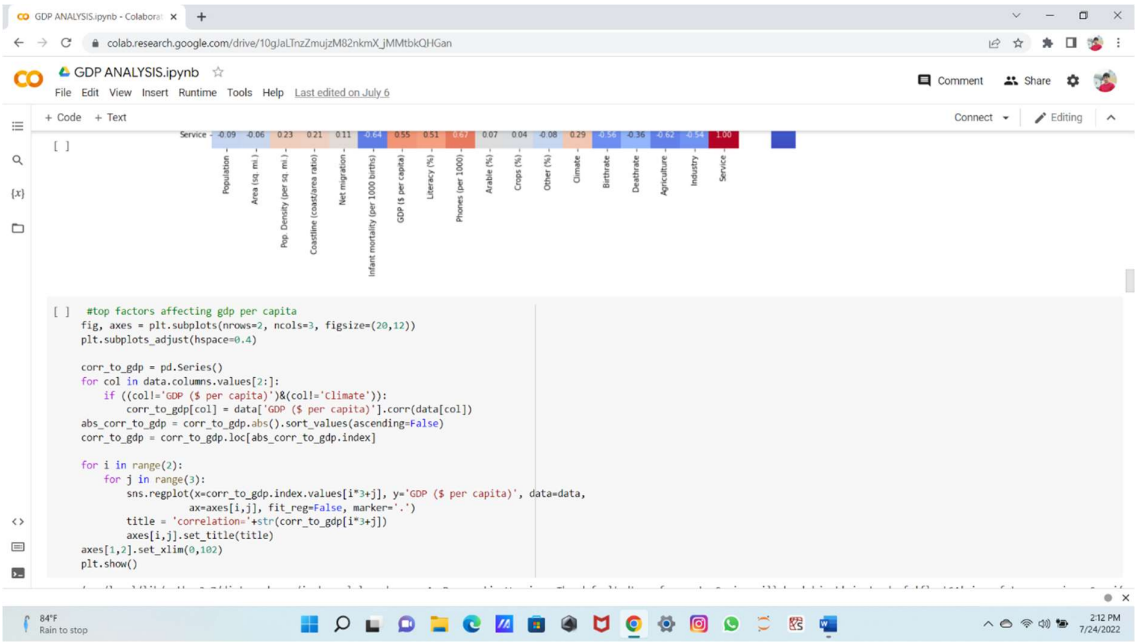

```
[ ]
    #Correlation between variables
    plt.figure(figsize=(16,12))
    sns.heatmap(data=data.iloc[:,2:].corr(),annot=True,fmt='.2f',cmap='coolwarm')
    plt.show()
```

1.00

84°F Rain to stop

2:11 PM 7/24/2022


GDP ANALYSIS.ipynb - Colaborat
colab.research.google.com/drive/10gJaLTnzZmujzM82nkmX_jMMtbkQHGan
GDP ANALYSIS.ipynb
File Edit View Insert Runtime Tools Help Last edited on July 6
Comment
Share
+ Code + Text
Connect
Editing
Population
Area (sq. mi.)
Pop. Density (per sq. mi.)
Coastline (coast/area ratio)
Net migration
Infant mortality (per 1000 births)
GDP ($ per capita)
Literacy (%)
Phones (per 1000)
Arable (%)
Crops (%)
Other (%)
Climate
Birthrate
Deathrate
Agriculture
Industry
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
84°F
Rain to stop
2:12 PM
7/24/2022


GDP ANALYSIS.ipynb - Colaborat
colab.research.google.com/drive/10gJaLTnzZmujzM82nkmX_jMMtbkQHGan
GDP ANALYSIS.ipynb
File Edit View Insert Runtime Tools Help Last edited on July 6
Comment
Share
+ Code + Text
Connect
Editing
Service
Population
Area (sq. mi.)
Pop. Density (per sq. mi.)
Coastline (coast/area ratio)
Net migration
Infant mortality (per 1000 births)
GDP ($ per capita)
Literacy (%)
Phones (per 1000)
Arable (%)
Crops (%)
Other (%)
Climate
Birthrate
Deathrate
Agriculture
Industry
Service
#top factors affecting gdp per capita
fig, axes = plt.subplots(nrows=2, ncols=3, figsize=(20,12))
plt.subplots_adjust(hspace=0.4)
corr_to_gdp = pd.Series()
for col in data.columns.values[2:]:
if ((col!='GDP ($ per capita)')&(col!='Climate')):
corr_to_gdp[col] = data['GDP ($ per capita)'].corr(data[col])
abs_corr_to_gdp = corr_to_gdp.abs().sort_values(ascending=False)
corr_to_gdp = corr_to_gdp.loc[abs_corr_to_gdp.index]
for i in range(2):
for j in range(3):
sns.regplot(x=corr_to_gdp.index.values[i*3+j], y='GDP ($ per capita)', data=data,
ax=axes[i,j], fit_reg=False, marker='.')
title = 'correlation='+str(corr_to_gdp[i*3+j])
axes[i,j].set_title(title)
axes[1,2].set_xlim(0,102)
plt.show()
84°F
Rain to stop
2:12 PM
7/24/2022

GDP ANALYSIS.ipynb - Colaborat

colab.research.google.com/drive/10gJaLTnzZmujzM82nkmX_jMMtbkQHGan

GDP ANALYSIS.ipynb

File Edit View Insert Runtime Tools Help Last edited on July 6

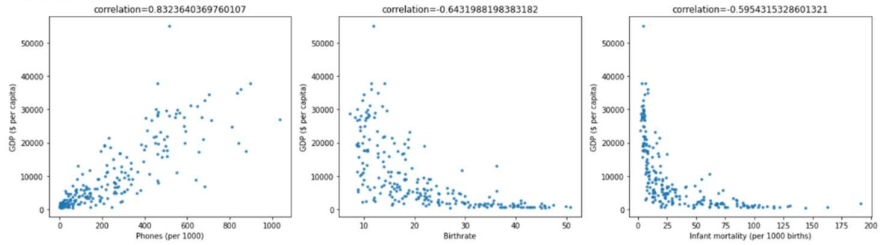
Comment Share

+ Code + Text

Connect Editing

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

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:4: DeprecationWarning: The default dtype for empty Series will be 'object' instead of 'float64' in a future version. Specif
  after removing the cwd from sys.path.
```



84°F Rain to stop

2:13 PM 7/24/2022

GDP ANALYSIS.ipynb - Colaborat

colab.research.google.com/drive/10gJaLTnzZmujzM82nkmX_jMMtbkQHGan

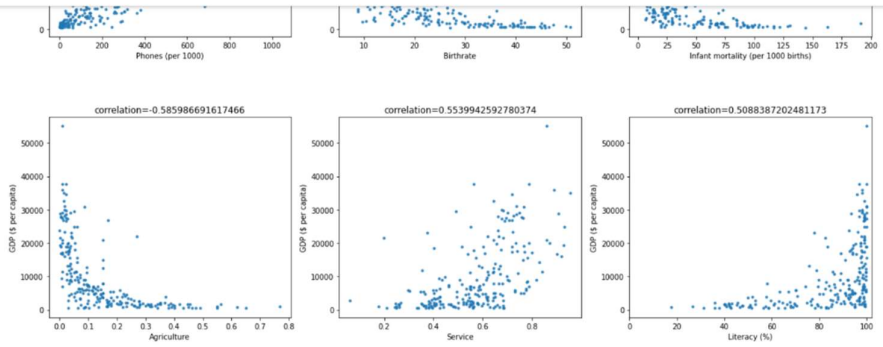
GDP ANALYSIS.ipynb

File Edit View Insert Runtime Tools Help Last edited on July 6

Comment Share

+ Code + Text

Connect Editing

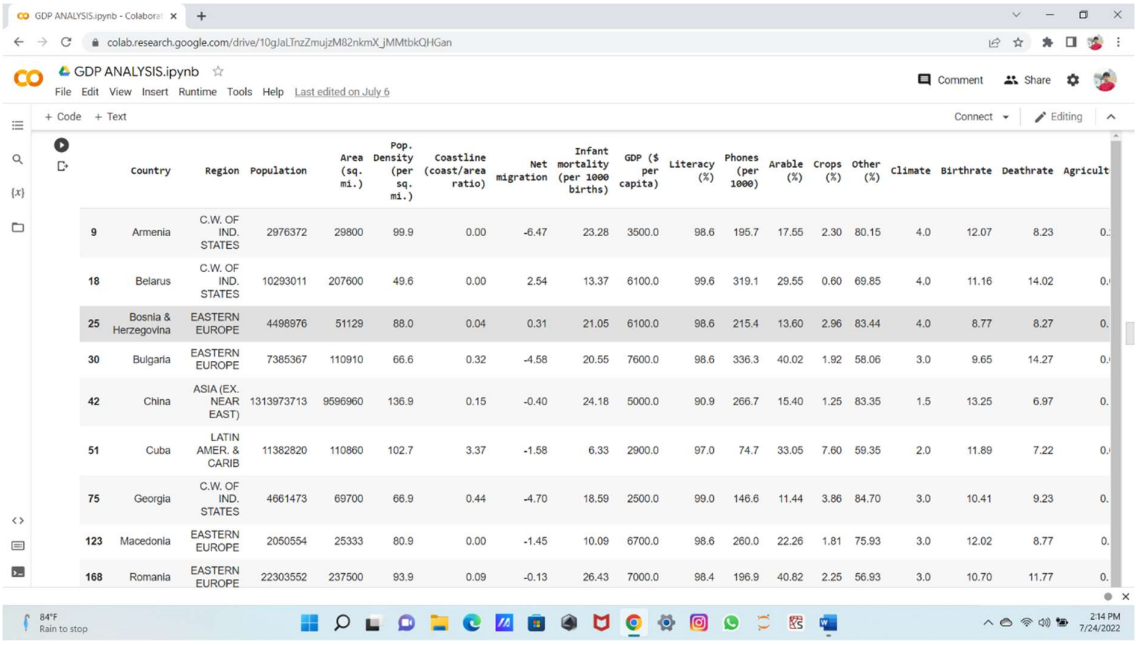



```
[ ] #companies with low birthrate and low gdp per capita
    data.loc[(data['Birthrate']<14)&(data['GDP ($ per capita)']<10000)]
```

Country	Region	Population	Area (sq.	Pop. Density (per	Coastline (coast/area	Net	Infant mortality	GDP ($ per	Literacy	Phones (per	Arable	Crops	Other	Climate	Birthrate	Deathrate	Agricult

84°F Rain to stop

2:13 PM 7/24/2022



GDP ANALYSIS.ipynb - Colaborat

colab.research.google.com/drive/10gJaLTnzZmujzM82nkmX_jMMtbkQHGan

GDP ANALYSIS.ipynb

File Edit View Insert Runtime Tools Help Last edited on July 6

Comment Share

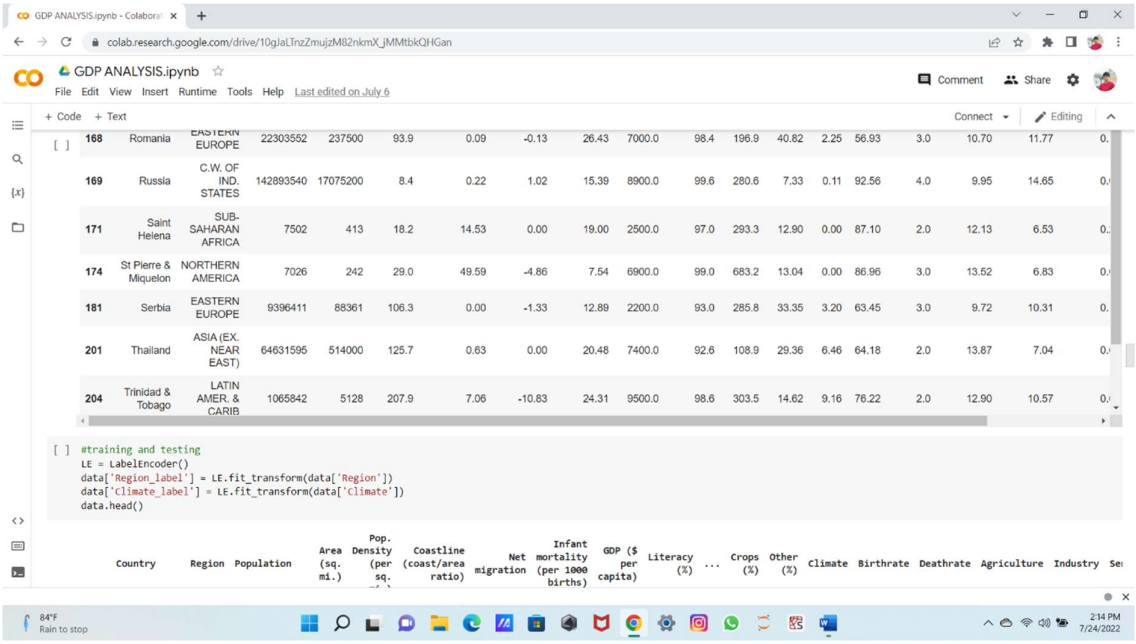
+ Code + Text

Connect Editing

	Country	Region	Population	Area (sq. mi.)	Pop. Density (per sq. mi.)	Coastline (coast/area ratio)	Net migration	Infant mortality (per 1000 births)	GDP ($ per capita)	Literacy (%)	Phones (per 1000)	Arable (%)	Crops (%)	Other (%)	Climate	Birthrate	Deathrate	Agricult
9	Armenia	C.W. OF IND. STATES	2976372	29800	99.9	0.00	-6.47	23.28	3500.0	98.6	195.7	17.55	2.30	80.15	4.0	12.07	8.23	0.
18	Belarus	C.W. OF IND. STATES	10293011	207600	49.6	0.00	2.54	13.37	6100.0	99.6	319.1	29.55	0.60	69.85	4.0	11.16	14.02	0.
25	Bosnia & Herzegovina	EASTERN EUROPE	4498976	51129	88.0	0.04	0.31	21.05	6100.0	98.6	215.4	13.60	2.96	83.44	4.0	8.77	8.27	0.
30	Bulgaria	EASTERN EUROPE	7385367	110910	66.6	0.32	-4.58	20.55	7600.0	98.6	336.3	40.02	1.92	58.06	3.0	9.65	14.27	0.
42	China	ASIA (EX. NEAR EAST)	1313973713	9596960	136.9	0.15	-0.40	24.18	5000.0	90.9	266.7	15.40	1.25	83.35	1.5	13.25	6.97	0.
51	Cuba	LATIN AMER. & CARIB	11382820	110860	102.7	3.37	-1.58	6.33	2900.0	97.0	74.7	33.05	7.60	59.35	2.0	11.89	7.22	0.
75	Georgia	C.W. OF IND. STATES	4661473	69700	66.9	0.44	-4.70	18.59	2500.0	99.0	146.6	11.44	3.86	84.70	3.0	10.41	9.23	0.
123	Macedonia	EASTERN EUROPE	2050554	25333	80.9	0.00	-1.45	10.09	6700.0	98.6	260.0	22.26	1.81	75.93	3.0	12.02	8.77	0.
168	Romania	EASTERN EUROPE	22303552	237500	93.9	0.09	-0.13	26.43	7000.0	98.4	196.9	40.82	2.25	56.93	3.0	10.70	11.77	0.

84°F Rain to stop

2:14 PM 7/24/2022



GDP ANALYSIS.ipynb - Colaborat

colab.research.google.com/drive/10gJaLTnzZmujzM82nkmX_jMMtbkQHGan

GDP ANALYSIS.ipynb

File Edit View Insert Runtime Tools Help Last edited on July 6

Comment Share

+ Code + Text

Connect Editing

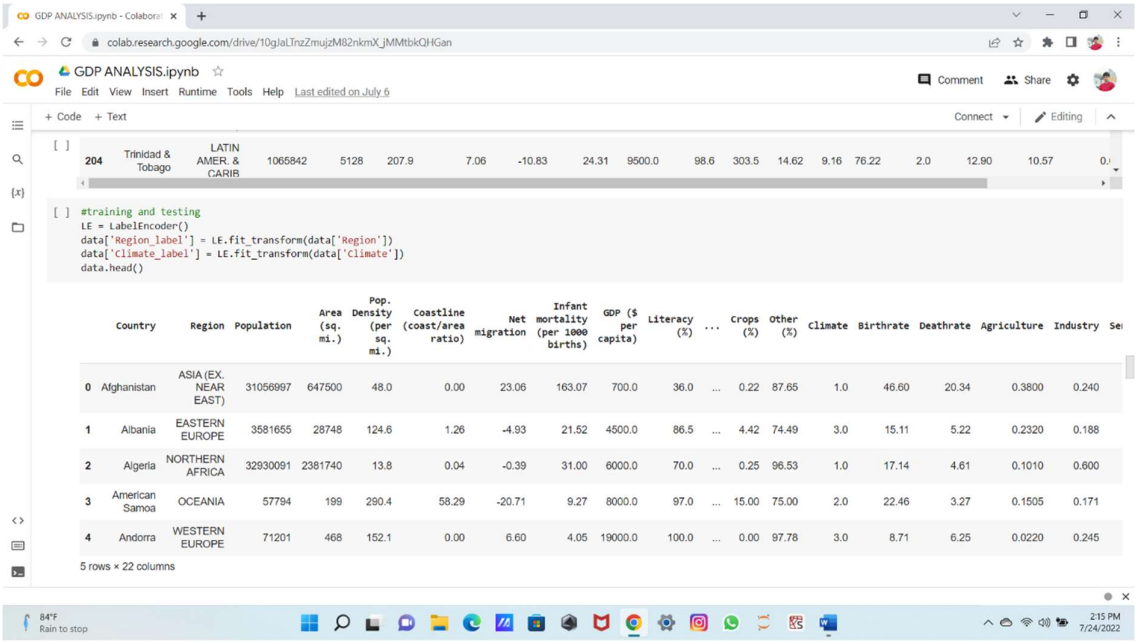
[ ]	168	Romania	EASTERN EUROPE	22303552	237500	93.9	0.09	-0.13	26.43	7000.0	98.4	196.9	40.82	2.25	56.93	3.0	10.70	11.77	0.
	169	Russia	C.W. OF IND. STATES	142893540	17075200	8.4	0.22	1.02	15.39	8900.0	99.6	280.6	7.33	0.11	92.56	4.0	9.95	14.65	0.
	171	Saint Helena	SUB-SAHARAN AFRICA	7502	413	18.2	14.53	0.00	19.00	2500.0	97.0	293.3	12.90	0.00	87.10	2.0	12.13	6.53	0.
	174	St Pierre & Miquelon	NORTHERN AMERICA	7026	242	29.0	49.59	-4.86	7.54	6900.0	99.0	683.2	13.04	0.00	86.96	3.0	13.52	6.83	0.
	181	Serbia	EASTERN EUROPE	9396411	88361	106.3	0.00	-1.33	12.89	2200.0	93.0	285.8	33.35	3.20	63.45	3.0	9.72	10.31	0.
	201	Thailand	ASIA (EX. NEAR EAST)	64631595	514000	125.7	0.63	0.00	20.48	7400.0	92.6	108.9	29.36	6.46	64.18	2.0	13.87	7.04	0.
	204	Trinidad & Tobago	LATIN AMER. & CARIB	1065842	5128	207.9	7.06	-10.83	24.31	9500.0	98.6	303.5	14.62	9.16	76.22	2.0	12.90	10.57	0.

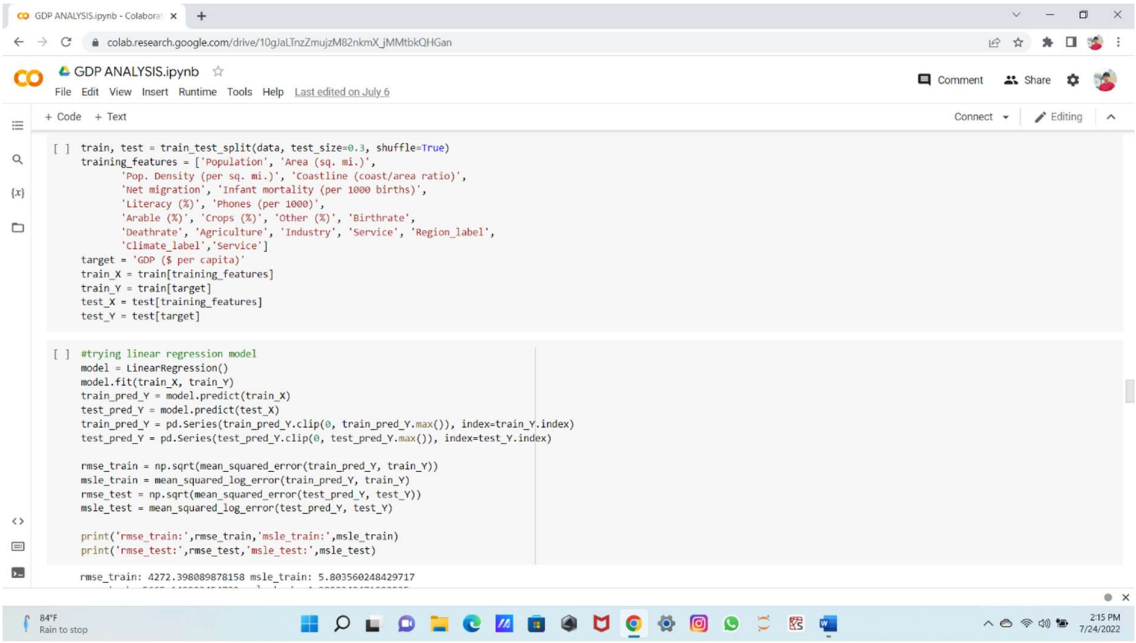
```
[ ] #training and testing
    LE = LabelEncoder()
    data['Region_label'] = LE.fit_transform(data['Region'])
    data['Climate_label'] = LE.fit_transform(data['Climate'])
    data.head()
```

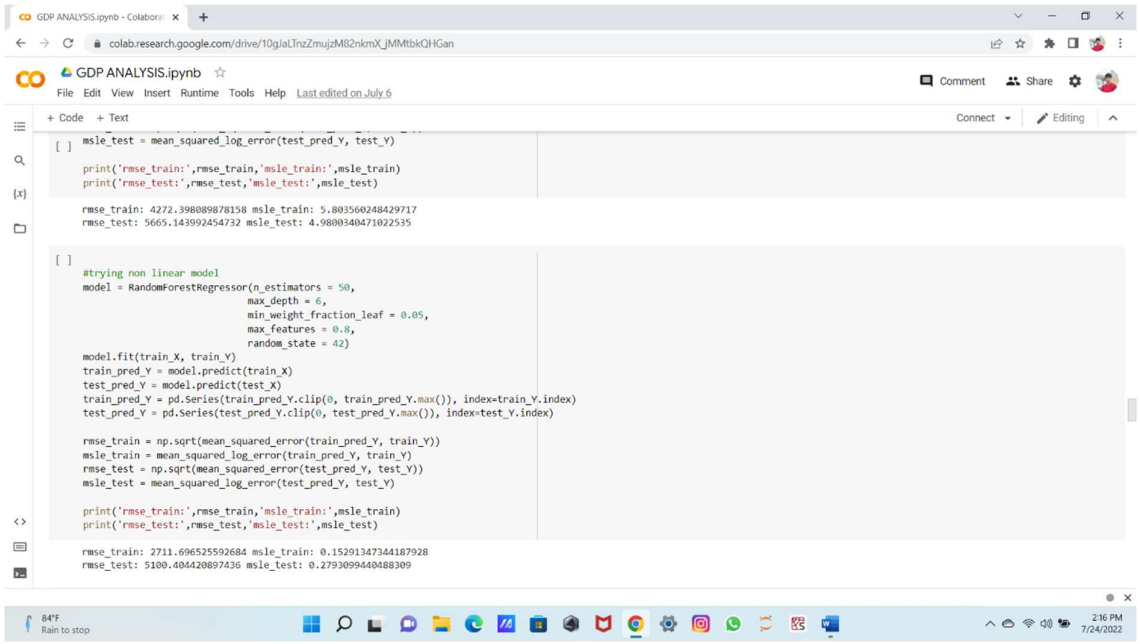
	Country	Region	Population	Area (sq. mi.)	Pop. Density (per sq.	Coastline (coast/area ratio)	Net migration	Infant mortality (per 1000 births)	GDP ($ per capita)	Literacy (%)	...	Crops (%)	Other (%)	Climate	Birthrate	Deathrate	Agriculture	Industry	Se

84°F Rain to stop

2:14 PM 7/24/2022


GDP ANALYSIS.ipynb - Colaborat
colab.research.google.com/drive/10gJaLTnzZmujzM82nkmX_jMMtbkQHGan
GDP ANALYSIS.ipynb
File Edit View Insert Runtime Tools Help Last edited on July 6
Comment
Share
+ Code + Text
Connect
Editing
204 Trinidad & Tobago LATIN AMER. & CARIB 1065842 5128 207.9 7.06 -10.83 24.31 9500.0 98.6 303.5 14.62 9.16 76.22 2.0 12.90 10.57
#training and testing
LE = LabelEncoder()
data['Region_label'] = LE.fit_transform(data['Region'])
data['Climate_label'] = LE.fit_transform(data['Climate'])
data.head()
Country Region Population Area (sq. mi.) Pop. Density (per sq. mi.) Coastline (coast/area ratio) Net migration Infant mortality (per 1000 births) GDP ($ per capita) Literacy (%) ... Crops (%) Other (%) Climate Birthrate Deathrate Agriculture Industry Ser
0 Afghanistan ASIA (EX. NEAR EAST) 31056997 647500 48.0 0.00 23.06 163.07 700.0 36.0 ... 0.22 87.65 1.0 46.60 20.34 0.3800 0.240
1 Albania EASTERN EUROPE 3581655 28748 124.6 1.26 -4.93 21.52 4500.0 86.5 ... 4.42 74.49 3.0 15.11 5.22 0.2320 0.188
2 Algeria NORTHERN AFRICA 32930091 2381740 13.8 0.04 -0.39 31.00 6000.0 70.0 ... 0.25 96.53 1.0 17.14 4.61 0.1010 0.600
3 American Samoa OCEANIA 57794 199 290.4 58.29 -20.71 9.27 8000.0 97.0 ... 15.00 75.00 2.0 22.46 3.27 0.1505 0.171
4 Andorra WESTERN EUROPE 71201 468 152.1 0.00 6.60 4.05 19000.0 100.0 ... 0.00 97.78 3.0 8.71 6.25 0.0220 0.245
5 rows × 22 columns
84°F Rain to stop
2:15 PM 7/24/2022


GDP ANALYSIS.ipynb - Colaborat
colab.research.google.com/drive/10gJaLTnzZmujzM82nkmX_jMMtbkQHGan
GDP ANALYSIS.ipynb
File Edit View Insert Runtime Tools Help Last edited on July 6
Comment
Share
+ Code + Text
Connect
Editing
train, test = train_test_split(data, test_size=0.3, shuffle=True)
training_features = ['Population', 'Area (sq. mi.)',
'Pop. Density (per sq. mi.)', 'Coastline (coast/area ratio)',
'Net migration', 'Infant mortality (per 1000 births)',
'Literacy (%)', 'Phones (per 1000)',
'Arable (%)', 'Crops (%)', 'Other (%)', 'Birthrate',
'Deathrate', 'Agriculture', 'Industry', 'Service', 'Region_label',
'Climate_label','Service']
target = 'GDP ($ per capita)'
train_X = train[training_features]
train_Y = train[target]
test_X = test[training_features]
test_Y = test[target]
#trying linear regression model
model = LinearRegression()
model.fit(train_X, train_Y)
train_pred_Y = model.predict(train_X)
test_pred_Y = model.predict(test_X)
train_pred_Y = pd.Series(train_pred_Y.clip(0, train_pred_Y.max()), index=train_Y.index)
test_pred_Y = pd.Series(test_pred_Y.clip(0, test_pred_Y.max()), index=test_Y.index)
rmse_train = np.sqrt(mean_squared_error(train_pred_Y, train_Y))
msle_train = mean_squared_log_error(train_pred_Y, train_Y)
rmse_test = np.sqrt(mean_squared_error(test_pred_Y, test_Y))
msle_test = mean_squared_log_error(test_pred_Y, test_Y)
print('rmse_train:',rmse_train,'msle_train:',msle_train)
print('rmse_test:',rmse_test,'msle_test:',msle_test)
rmse_train: 4272.398089878158 msle_train: 5.803560248429717
84°F Rain to stop
2:15 PM 7/24/2022

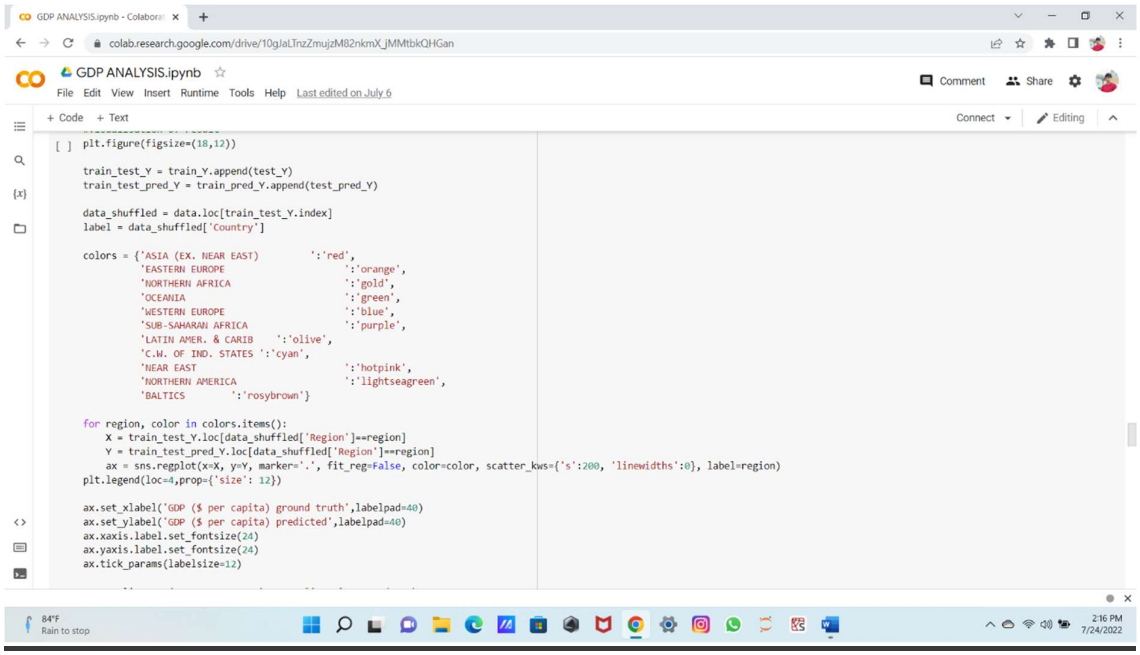


```
msle_test = mean_squared_log_error(test_pred_Y, test_Y)

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

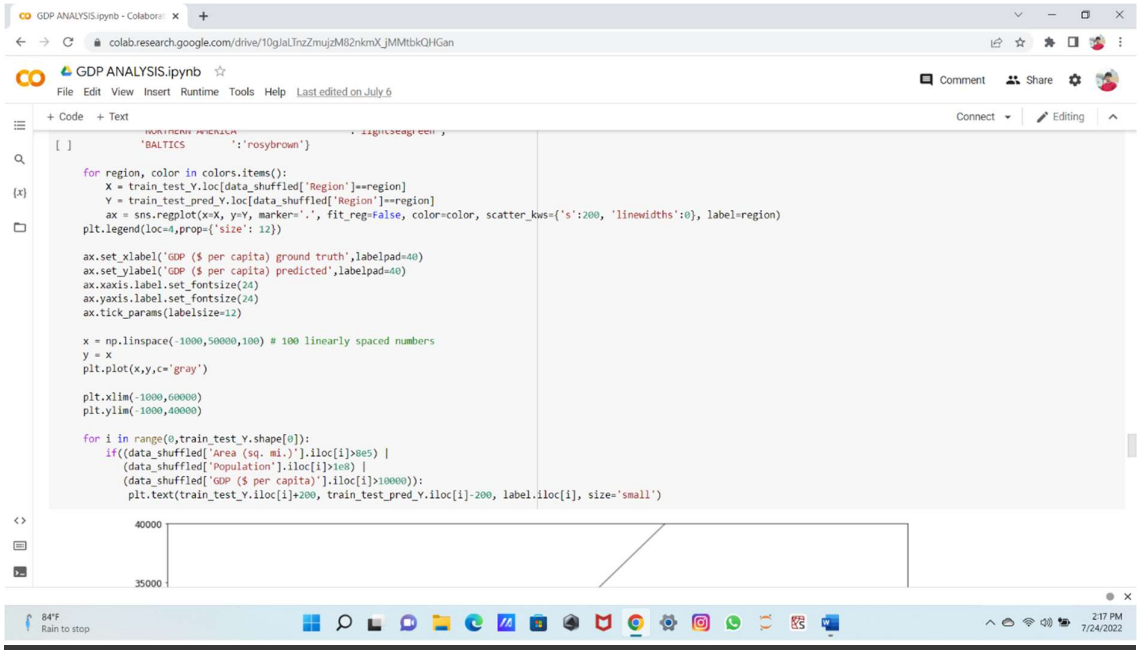
```
rmse_train: 4272.398089878158 msle_train: 5.803560248429717
rmse_test: 5665.143992454732 msle_test: 4.980030471022535
```

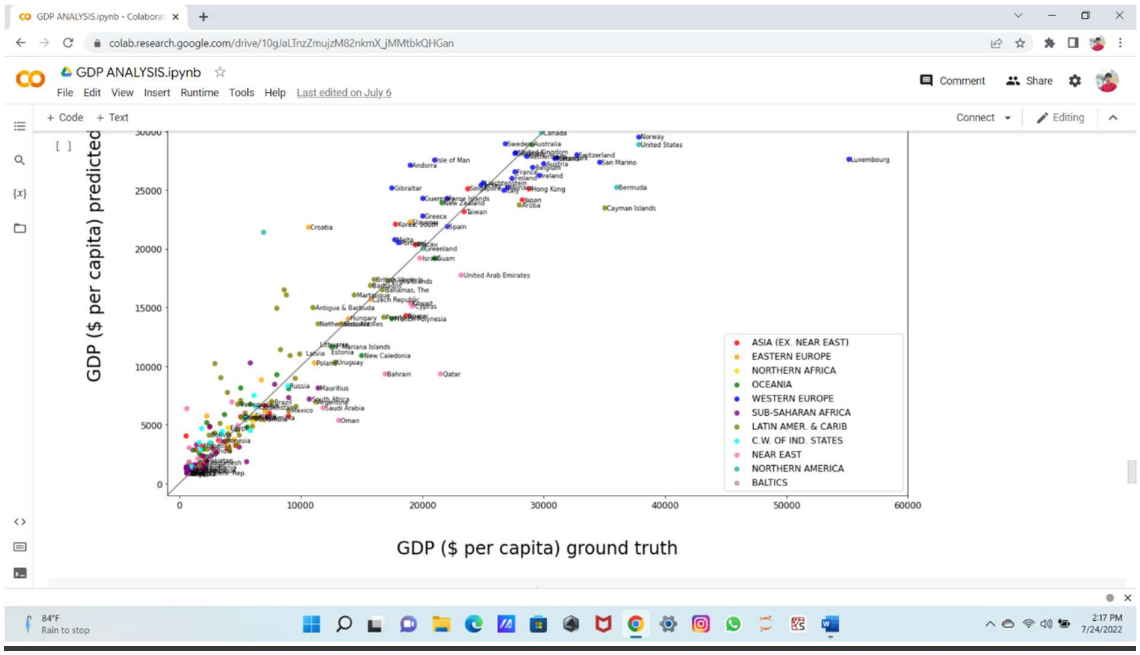
```
#trying non linear model
model = RandomForestRegressor(n_estimators = 50,
                              max_depth = 6,
                              min_weight_fraction_leaf = 0.05,
                              max_features = 0.8,
                              random_state = 42)
model.fit(train_X,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

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

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

```
rmse_train: 2711.696525592684 msle_train: 0.15291347344187928
rmse_test: 5100.404420897436 msle_test: 0.2793099440488309
```



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

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

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

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

for region, color in colors.items():
    X = train_test_Y.loc[data_shuffled['Region']==region]
    Y = train_test_pred_Y.loc[data_shuffled['Region']==region]
    ax = sns.regplot(x=X, y=Y, marker='.', fit_reg=False, color=color, scatter_kws={'s':200, 'linewidths':0}, label=region)
plt.legend(loc=4,prop={'size': 12})

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


GDP ANALYSIS.ipynb
File Edit View Insert Runtime Tools Help Last edited on July 6
Comment Share
+ Code + Text
Connect Editing
'BALTICS ':'rosybrown'}
for region, color in colors.items():
X = train_test_Y.loc[data_shuffled['Region']==region]
Y = train_test_pred_Y.loc[data_shuffled['Region']==region]
ax = sns.regplot(x=X, y=Y, marker='.', fit_reg=False, color=color, scatter_kws={'s':200, 'linewidths':0}, label=region)
plt.legend(loc=4,prop={'size': 12})
ax.set_xlabel('GDP ($ per capita) ground truth',labelpad=40)
ax.set_ylabel('GDP ($ per capita) predicted',labelpad=40)
ax.xaxis.label.set_fontsize(24)
ax.yaxis.label.set_fontsize(24)
ax.tick_params(labelsize=12)
x = np.linspace(-1000,50000,100) # 100 linearly spaced numbers
y = x
plt.plot(x,y,c='gray')
plt.xlim(-1000,60000)
plt.ylim(-1000,40000)
for i in range(0,train_test_Y.shape[0]):
if((data_shuffled['Area (sq. mi.)'].iloc[i]>8e5) |
(data_shuffled['Population'].iloc[i]>1e8) |
(data_shuffled['GDP ($ per capita)'].iloc[i]>10000)):
plt.text(train_test_Y.iloc[i]+200, train_test_pred_Y.iloc[i]-200, label.iloc[i], size='small')
40000
35000
84°F
Rain to stop
2:17 PM
7/24/2022


GDP ANALYSIS.ipynb
File Edit View Insert Runtime Tools Help Last edited on July 6
Comment Share
+ Code + Text
Connect Editing
GDP ($ per capita) predicted
GDP ($ per capita) ground truth
0
5000
10000
15000
20000
25000
0
10000
20000
30000
40000
50000
60000
ASIA (EX. NEAR EAST)
EASTERN EUROPE
NORTHERN AFRICA
OCEANIA
WESTERN EUROPE
SUB-SAHARAN AFRICA
LATIN AMER. & CARIB
C.W. OF IND. STATES
NEAR EAST
NORTHERN AMERICA
BALTICS
Luxembourg
Bermuda
Cayman Islands
United Arab Emirates
Qatar
Bahrain
84°F
Rain to stop
2:17 PM
7/24/2022

GDP ANALYSIS.ipynb - Colabora

colab.research.google.com/drive/10gJaLTnzZmujzM82nkmX_jMMtbkQHGan

GDP ANALYSIS.ipynb

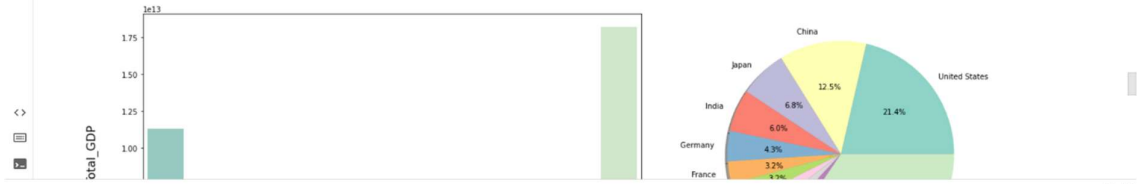
File Edit View Insert Runtime Tools Help Last edited on July 6

Comment Share

+ Code + Text

Connect Editing

```
[ ]
    #total 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 ($)'], labels=gdps['Country'],colors=colors,autopct='%1.1f%%',shadow=True)
    axes[1].axis('equal')
    plt.show()
```



84°F Rain to stop

2:19 PM 7/24/2022

GDP ANALYSIS.ipynb - Colabora

colab.research.google.com/drive/10gJaLTnzZmujzM82nkmX_jMMtbkQHGan

GDP ANALYSIS.ipynb

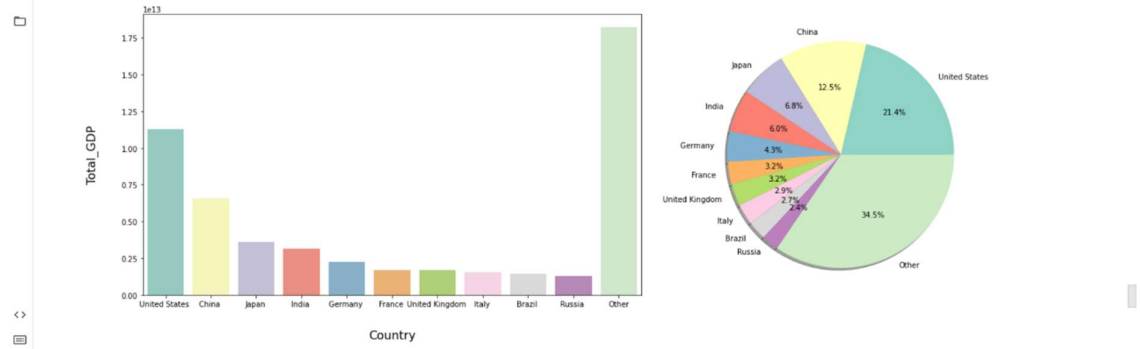
File Edit View Insert Runtime Tools Help Last edited on July 6

Comment Share

+ Code + Text

Connect Editing

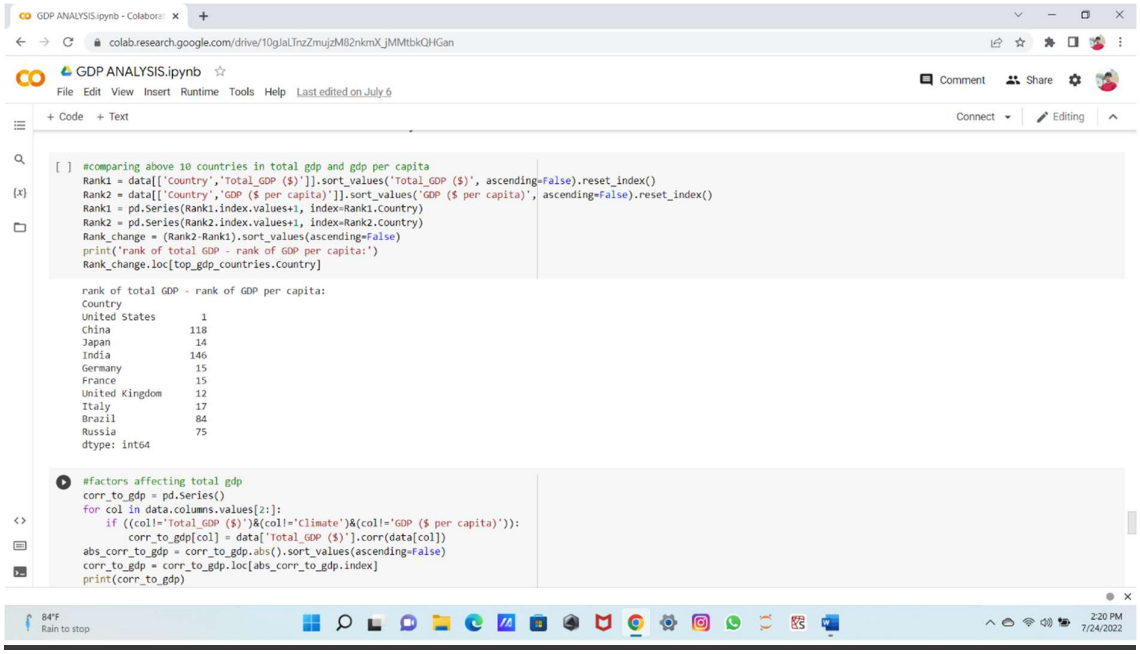
```
[ ] colors = sns.color_palette("Set3", gdps.shape[0]).as_hex()
    axes[1].pie(gdps['Total_GDP ($)'], labels=gdps['Country'],colors=colors,autopct='%1.1f%%',shadow=True)
    axes[1].axis('equal')
    plt.show()
```




```
[ ] #comparing above 10 countries in total gdp and gdp per capita
```

84°F Rain to stop

2:19 PM 7/24/2022



GDP ANALYSIS.ipynb - Colaborat

colab.research.google.com/drive/10gJaLTnzZmujzM82nkmX_jMMtbkQHGan

GDP ANALYSIS.ipynb

File Edit View Insert Runtime Tools Help Last edited on July 6

Comment Share

+ Code + Text

Connect Editing

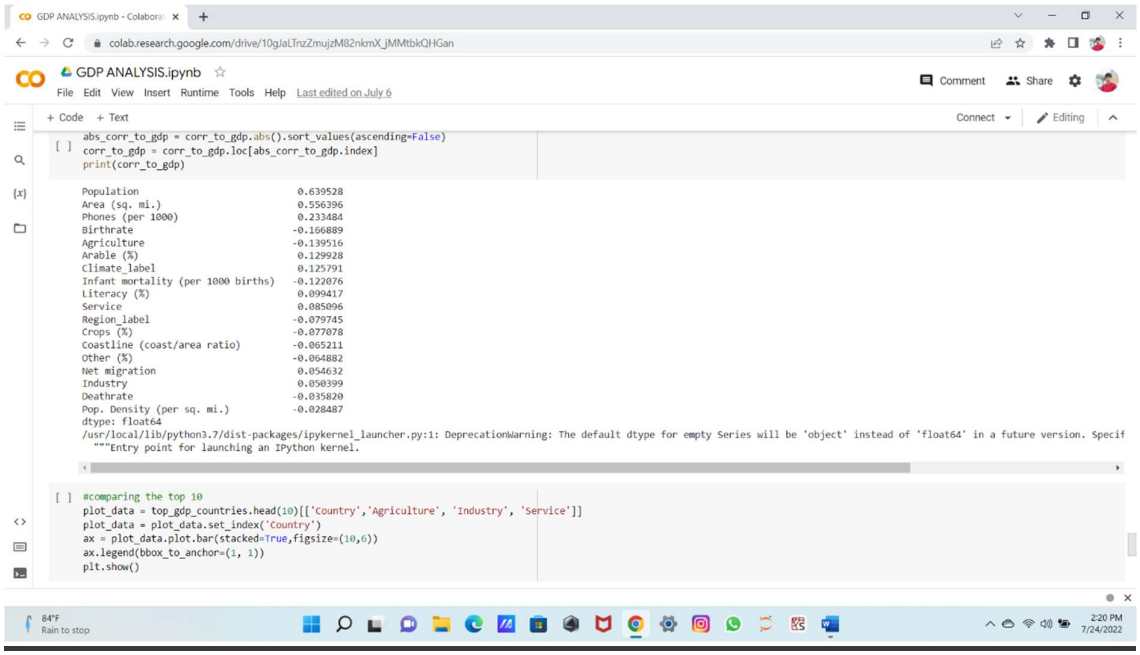
```
[ ] #comparing above 10 countries in total gdp and gdp per capita
    Rank1 = data[['Country','Total_GDP ($)']].sort_values('Total_GDP ($)', ascending=False).reset_index()
    Rank2 = data[['Country','GDP ($ per capita)']].sort_values('GDP ($ per capita)', ascending=False).reset_index()
    Rank1 = pd.Series(Rank1.index.values+1, index=Rank1.Country)
    Rank2 = pd.Series(Rank2.index.values+1, index=Rank2.Country)
    Rank_change = (Rank2-Rank1).sort_values(ascending=False)
    print('rank of total GDP - rank of GDP per capita:')
    Rank_change.loc[top_gdp_countries.Country]
```

```
rank of total GDP - rank of GDP per capita:
Country
United States       1
China             118
Japan              14
India             146
Germany            15
France             15
United Kingdom     12
Italy              17
Brazil             84
Russia             75
dtype: int64
```

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

84°F Rain to stop

2:20 PM 7/24/2022



GDP ANALYSIS.ipynb - Colaborat

colab.research.google.com/drive/10gJaLTnzZmujzM82nkmX_jMMtbkQHGan

GDP ANALYSIS.ipynb

File Edit View Insert Runtime Tools Help Last edited on July 6

Comment Share

+ Code + Text

Connect Editing

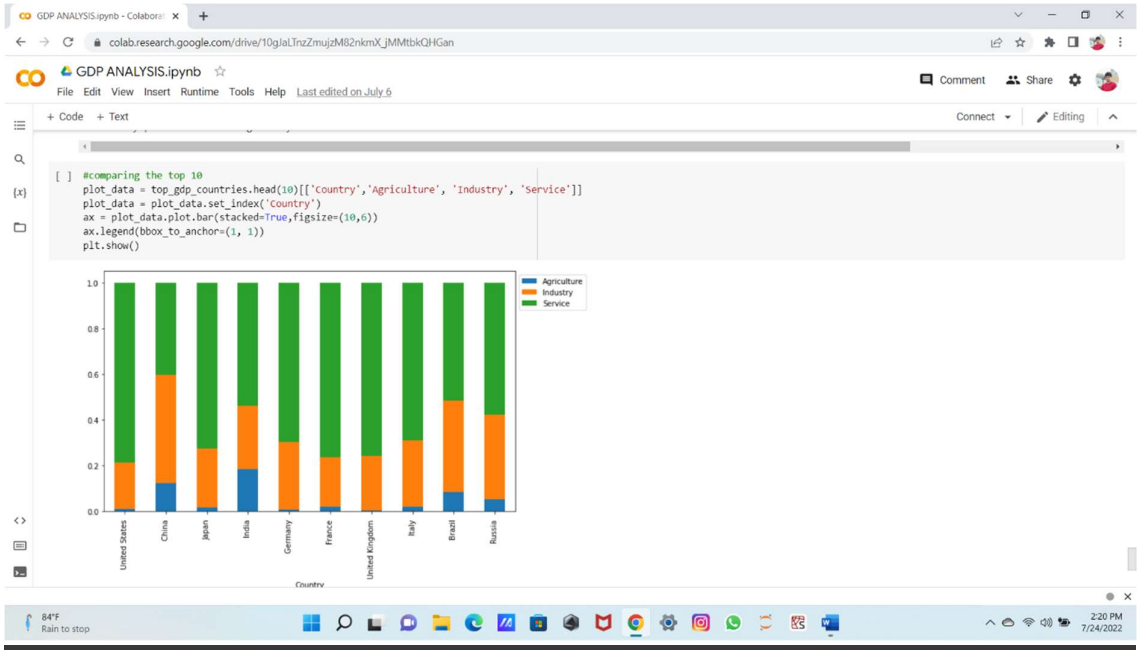
```
[ ] abs_corr_to_gdp = corr_to_gdp.abs().sort_values(ascending=False)
    corr_to_gdp = corr_to_gdp.loc[abs_corr_to_gdp.index]
    print(corr_to_gdp)
```

```
Population                          0.639528
Area (sq. mi.)                      0.556396
Phones (per 1000)                   0.233484
Birthrate                          -0.166889
Agriculture                        -0.139516
Arable (%)                          0.129928
Climate_label                       0.125791
Infant mortality (per 1000 births) -0.122076
Literacy (%)                        0.099417
Service                             0.085096
Region_label                       -0.079745
Crops (%)                          -0.077078
Coastline (coast/area ratio)       -0.065211
Other (%)                          -0.064882
Net migration                       0.054632
Industry                            0.050399
Deathrate                          -0.035820
Pop. Density (per sq. mi.)         -0.028487
dtype: float64
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: DeprecationWarning: The default dtype for empty Series will be 'object' instead of 'float64' in a future version. Specif
  """Entry point for launching an IPython kernel.
```

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

84°F Rain to stop

2:20 PM 7/24/2022



CONCLUSION

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

Bibilography:

[1] https://towardsdatascience.com/a-data-science-workflow-26c3f05a010e

[2] https://www.investopedia.com/terms/p/per-capita-gdp.asp

[3] https://thecleverprogrammer.com/2020/05/26/gdp-analysis-with-data-science/

[4] https://www.guru99.com/data-science-tutorial.html

THANK YOU