GROSS DOMESTIC PRODUCT PREDECTION USING MACHINE LEARNING

PROJECT REPORT

Submitted by,

CH. YASWANTH	**-**	**723921243011**
G. MAHENDRA	**-**	**723921243020**
CH. HEMANTH KUMAR	**-**	**723921243014**
V. ADARSH VARMA	**-**	**723921243056**

in partial fulfilment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

ARJUN COLLEGE OF TECHNOLOGY

COIMBATORE - 642 120

ANNA UNIVERSITY: CHENNAI 600 025

MAY 2025

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this Report titled **"Gross Domestic Product Prediction using Machine Learning"** is the bonafide work of **CH.YASWANTH (723921243011), G.MAHENDRA (723921243020), CH.HEMANTH KUMAR(723921243014), V.ADARSH VARMA (723921243056)** who carried out the work under my supervision. Certified further that to the best of my knowledge the work reported here in does not form part of any other project work on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE	**SIGNATURE**
SUPERVISOR	**HEAD OF THE DEPARTMENT**
Mr. BHASKARAN N A	**Dr. J. THILAGAVATHI MCA., Ph.D**
Associate Professor & Head	Associate Professor & Head
Department of IT,	Department of AI & DS,
Arjun College of Technology,	Arjun College of Technology,
Coimbatore - 642 120	Coimbatore - 642 120

Submitted for the University Project Viva-Voce held on _______________

Internal Examiner **External Examiner**

ACKNOWLEDGEMENT

We owe our sincere and heartfelt thanks to our chairman **Thiru. R. SURIYANARAYANAN,** and also we extend our profound thanks to our Secretary **Dr. R. SURESH KUMAR M.E., Ph.D.,** for their exuberance in motivating young minds.

Our deepest gratitude and thanks to our motivator and Principal **Dr. N. JANAKI MANOHAR M.E., Ph.D.,** who always helping us whenever we approach him during the course of our project.

We would also like to express our profound thanks to our Head of the Department **Dr. J. THILAGAVATHI MCA., Ph.D.,** Associate Professor, Department of Artificial Intelligence and Data Science, whose thoughtful words, advise and help to complete our project successfully.

We would also like to express our profound thanks to our Project Coordinator and guide **Mr. BHASKARAN N A,** Associate Professor & Head of the Department of Information Technology, whose continuous support and help to complete our project successfully.

We express our sincere thanks to all **Faculty Members and Skilled Assistants** of Computer Science and Engineering Department and our lovable **Friends** for their help and wishes for the successful completion of this project.

Finally, yet importantly, we would like to express our indebtedness to our beloved **Parents** for their affectionate blessing co-operation at all stages of this academic venture and also our well-wishers.

ABSTRACT

Gross Domestic Product is cited as vital and most widely accepted economic indicator which not only helps in diagnosing the problems related to the economy but also correcting it. The usage of the gross domestic product as a measure of the market price of ultimate services and product that are produced over a selected amount of time will definitely continue to owe an abundant to the producing age. To policy makers and statisticians especially, gross domestic product helps in conveying data about the economy in particular and thereby notifying about country’s economic health. This project makes an attempt to expedite the process of prediction of Gross Domestic Product. Machine Learning algorithms such as Linear Regression and Random Forest are used for prediction. The proposed method using machine learning model proves to be fruitful for financial management.

CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	TITLE PAGE	**i**
	BONAFIED CERTIFICATE	**ii**
	ACKNOWLEDGEMENT	**iii**
	ABSTRACT	**iv**
	LIST OF FIGURES	**v**
1	**INTRODUCTION**	**1**
1.1	INTRODUCTION ABOUT DOMAIN	2
1.1.1	MACHINE LEARNING METHODS	3
1.1.2	APPROACHES	5
2	**LITERATURE SURVEY**	**9**
3	**SYSTEM ANALYSIS**	**11**
3.1	EXISTING SYSTEM	11
3.1.1	DISADVANTAGES OF EXISTING SYSTEM	11
4	**PROPOSED SYSTEM**	**12**
4.1	ADVANTAGES OF PROPOSED SYSTEM	12
5	**SYSTEM SPECIFICATIONS**	**13**
5.1	HARDWARE REQUIREMENTS	13
5.2	SOFTWARE REQUIREMENTS	13
6	**SYSTEM STUDY**	**14**
6.1	FEASIBILITY STUDY	14
6.2	ECONOMICAL FEASIBILITY	14
6.3	TECHNICAL FEASIBILITY	15
6.4	SOCIAL FEASIBILITY	15
7	**SOFTWARE ENVIRONMENT**	**16**
7.1	INTRODUCTION	16
7.2	FEATURE OF PYTHON	17
7.3	PYTHON INSTALLATION	19
7.4	BASIC PRINCIPLES OF PYTHON	23

	7.5	EXCEPTION HANDLING	28
8		**INPUT AND OUTPUT DESIGNS**	**30**
	8.1	INPUT DESIGN	30
	8.1.1	OBJECTIVES	30
	8.2	OUTPUT DESIGN	31
9		**MODULES**	**32**
	9.1	ADMIN MODULE	32
10		**IMPLEMENTIATION**	**33**
	10.1	SYSTEM ARCHITECTER	33
	10.2	LINEAR REGRESSION	34
	10.3	RANDOM FOREST	34
11		**DATA FLOW DIAGRAMS**	**35**
12		**ACTIVITY DIAGRAM**	**35**
13		**CLASS DIAGRAM**	**36**
14		**SEQUENCE DIAGRAM**	**36**
15		**USE CASE DIAGRAM**	**37**
16		**SYSTEM TEST**	**38**
	16.1	TYPES OF TESTING’S	38
	16.2	TEST STRATEGIES AND APPROACHES	40
17		**RESULT**	**42**
18		**CONCLUSION**	**48**
		SOURCE CODE	**49**
		REFERENCES	**60**

LIST OF FIGURES

Fig No.	Title	Page No.
7.2	FEATURES OF PYTHON	17
10.1	ARCHITECTURE DIAGRAM	33
11.1	DATA FLOW DIAGRAM	35
12.1	ACTIVITY DIAGRAM	35
13.1	CLASS DIAGRAM	36
14.1	SEQUENCE DIAGRAM	36
15.1	USE CASE DIAGRAM	37

CHAPTER 1
INTRODUCTION

Gross Domestic Product (GDP) is the market price of all product and services that area unit made inside the country's national borders in a year. Gross domestic product could be a measure to assess overall economic performance of a country, it includes all product and services made by the economy as well as personal consumption, government purchase, non-public inventories, paid in construction prices and therefore the foreign trade gap. The topic of GDP became of high importance among the indicator of economy variables. Information on Gross Domestic Product is thought to be a crucial indicator for evaluating the national economic development and growth of entire macro economy.

GDP aggregates the complete economic motion. It is frequently used as a finest measure to calculate the performance of the economy. GDP is mostly measured in one of a 3 approaches. First, the Expenditure approach, it involves the worth of all domestic expenditures created on final product and services of the year, beside consumption expenditures, investment expenditures, government expenditures, and web exports. Second, the assembly approach, it's involving the summation of all additional activities at each a part of production by all industries inside the country, taxes and product's subsidies of the amount. Third is that the financial gain approach, it's the summation of all aspect of the financial gain created by production inside the economy as remuneration of workers, capital financial gain, and gross in operation surplus of enterprises i.e., profit, taxes on production and imports less grants of the quantity.

The aim of this study is to predict GDP, using linear regression and random forest for a particular period. Prediction of GDP involves application of applied mathematics and mathematical model to predict future developments within the economy. It permits to review previous economic movements and predict however current economic changes can amend the patterns of previous trend; therefore, a more accurate prediction would provide a significance facilitate to the government in setting up economic development goals, ways and policies. Consequently, a correct Gross Domestic Product prediction presents a number one insight associate an understanding for future economics' trend.

1.1 INTRODUCTION ABOUT DOMAIN MACHINE LEARNING

Machine learning is a subfield of Artificial Intelligence (AI). The goal of machine learning generally is to understand the structure of data and fit that data into models that can be understood and utilized by people.

Although machine learning is a field within computer science, it differs from traditional computational approaches. In traditional computing, algorithms are sets of explicitly programmed instructions used by computers to calculate or problem solve. Machine learning algorithms instead allow for computers to train on data inputs and use statistical analysis in order to output values that fall within a specific range. Because of this, machine learning facilitates computers in building models from sample data in order to automate decision-making processes based on data inputs.

Any technology user today has benefitted from machine learning. Facial recognition technology allows social media platforms to help users tag and share photos of friends.

Optical Character Recognition (OCR) technology converts images of text into movable type. Recommendation engines, powered by machine learning, suggest what movies or television shows to watch next based on user preferences. Self-driving cars that rely on machine learning to navigate may soon be available to consumers.
Machine learning is a continuously developing field. Because of this, there are some considerations to keep in mind as you work with machine learning methodologies, or analyze the impact of machine learning processes.

In this project, we’ll look into the common machine learning methods of supervised and unsupervised learning, and common algorithmic approaches in machine learning, including the k- nearest neighbor algorithm, decision tree learning, and deep learning. We'll explore which programming languages are most used in machine learning, providing you with some of the positive and negative attributes of each. Additionally, we'll discuss biases that are perpetuated by machine learning algorithms, and consider what can be kept in mind to prevent these biases when building algorithms.

1.1.1 Machine Learning Methods

In machine learning, tasks are generally classified into broad categories. These categories are based on how learning is received or how feedback on the learning is given to the system developed.

Two of the most widely adopted machine learning methods are **supervised learning** which trains algorithms based on example input and output data that is labeled by humans, and **unsupervised learning** which provides the algorithm with no labeled data in order to allow it to find structure within its input data. Let's explore these methods in more detail.

Supervised Learning

In supervised learning, the computer is provided with example inputs that are labeled with their desired outputs. The purpose of this method is for the algorithm to be able to "learn" by comparing its actual output with the "taught" outputs to find errors, and modify the model accordingly. Supervised learning therefore uses patterns to predict label values on additional unlabeled data.

For example, with supervised learning, an algorithm may be fed data with images of sharks labeled as fish and images of oceans labeled as water. By being trained on this data, the supervised learning algorithm should be able to later identify unlabeled shark images as fish and unlabeled ocean images as water.

A common use case of supervised learning is to use historical data to predict statistically likely future events. It may use historical stock market information to anticipate upcoming fluctuations, or be employed to filter out spam emails. In supervised learning, tagged photos of dogs can be used as input data to classify untagged photos of dogs.

Unsupervised Learning

In unsupervised learning, data is unlabeled, so the learning algorithm is left to find commonalities among its input data. As unlabeled data are more abundant than labeled data, machine learning methods that facilitate unsupervised learning are particularly valuable. The goal of unsupervised learning may be as straightforward as discovering hidden patterns within a dataset, but it may also have a goal of feature learning, which allows the computational machine to automatically discover the representations that are needed to classify raw data.

Unsupervised learning is commonly used for transactional data. You may have a large dataset of customers and their purchases, but as a human you will likely not be able to make sense of what similar attributes can be drawn from customer profiles and their types of purchases. With this data fed into an unsupervised learning algorithm, it may be determined that women of a certain age range who buy unscented soaps are likely to be pregnant, and therefore a marketing campaign related to pregnancy and baby products can be targeted to this audience in order to increase their number of purchases. Without being told a “correct” answer, unsupervised learning methods can look at complex data that is more expansive and seemingly unrelated in order to organize it in potentially meaningful ways. Unsupervised learning is often used for anomaly detection including for fraudulent credit card purchases, and recommender systems that recommend what products to buy next. In unsupervised learning, untagged photos of dogs can be used as input data for the algorithm to find likenesses and classify dog photos together.

1.1.2 Approaches

As a field, machine learning is closely related to computational statistics, so having a background knowledge in statistics is useful for understanding and leveraging machine learning algorithms. For those who may not have studied statistics, it can be helpful to first define correlation and regression, as they are commonly used techniques for investigating the relationship among quantitative variables.

Correlation is a measure of association between two variables that are not designated as either dependent or independent. **Regression** at a basic level is used to examine the relationship between one dependent and one independent variable. Because regression statistics can be used to anticipate the dependent variable when the independent variable is known, regression enables prediction capabilities.

Approaches to machine learning are continuously being developed. For our purposes, we'll go through a few of the popular approaches that are being used in machine learning at the time of writing.

k-nearest neighbor

The k-nearest neighbor algorithm is a pattern recognition model that can be used for classification as well as regression. Often abbreviated as k-NN, the **k** in k-nearest neighbor is a positive integer, which is typically small. In either classification or regression, the input will consist of the k closest training examples within a space.

We will focus on k-NN classification. In this method, the output is class membership. This will assign a new object to the class most common among its k nearest neighbors. In the case of k = 1, the object is assigned to the class of the single nearest neighbor.

Among the most basic of machine learning algorithms, k-nearest neighbor is considered to be a type of "lazy learning" as generalization beyond the training data does not occur until a query is made to the system.

Decision Tree Learning

For general use, decision trees are employed to visually represent decisions and show or inform decision making. When working with machine learning and data mining, decision trees are used as a predictive model. These models map observations about data to conclusions about the data's target valueThe goal of decision tree learning is to create a model that will predict the value of a target based on input variables.

In the predictive model, the data's attributes that are determined through observation are represented by the branches, while the conclusions about the data's target value are represented in the leaves.

When "learning" a tree, the source data is divided into subsets based on an attribute value test, which is repeated on each of the derived subsets recursively. Once the subset at a node has the equivalent value as its target value has, the recursion process will be complete.

Let's look at an example of various conditions that can determine whether or not someone should go fishing. This includes weather conditions as well as barometric pressure conditions.

A true classification tree data set would have a lot more features than what is outlined above, but relationships should be straightforward to determine. When working with decision tree learning, several determinations need to be made, including what features to choose, what conditions to use for splitting, and understanding when the decision tree has reached a clear ending.

Deep Learning

Deep learning attempts to imitate how the human brain can process light and sound stimuli into vision and hearing. A deep learning architecture is inspired by biological neural networks and consists of multiple layers in an artificial neural network made up of hardware and GPUs.

Deep learning uses a cascade of nonlinear processing unit layers in order to extract or transform features (or representations) of the data.

The output of one layer serves as the input of the successive layer. In deep learning, algorithms can be either supervised and serve to classify data, or unsupervised and perform pattern analysis.

Among the machine learning algorithms that are currently being used and developed, deep learning absorbs the most data and has been able to beat humans in some cognitive tasks. Because of these attributes, deep learning has become the approach with significant potential in the artificial intelligence space.
Computer vision and speech recognition have both realized significant advances from deep learning approaches. IBM Watson is a well-known example of a system that leverages deep learning.

Programming Languages

When choosing a language to specialize in with machine learning, you may want to consider the skills listed on current job advertisements as well as libraries available in various languages that can be used for machine learning processes.

From data taken from job ads on indeed.com in December 2016, it can be inferred that Python is the most sought-for programming language in the machine learning professional field. Python is followed by Java, then R, then C++.

Python's popularity may be due to the increased development of deep learning frameworks available for this language recently, including TensorFlow, PyTorch, and Keras. As a language that has readable syntax and the ability to be used as a scripting language, Python proves to be powerful and straightforward both for preprocessing data and working with data directly.

The scikit-learn machine learning library is built on top of several existing Python packages that Python developers may already be familiar with, namely NumPy, SciPy, and Matplotlib.

To get started with Python, you can read our tutorial series on "How To Code in Python 3," or read specifically on "How To Build a Machine Learning Classifier in

Python with scikit-learn” or “How To Perform Neural Style Transfer with Python 3 and PyTorch.”

Java is widely used in enterprise programming, and is generally used by front-end desktop application developers who are also working on machine learning at the enterprise level. Usually it is not the first choice for those new to programming who want to learn about machine learning, but is favored by those with a background in Java development to apply to machine learning. In terms of machine learning applications in industry, Java tends to be used more than Python for network security, including in cyberattack and fraud detection use cases.

Among machine learning libraries for Java are Deeplearning4j, an open-source and distributed deep-learning library written for both Java and Scala; MALLET (Machine Learning for Language Toolkit) allows for machine learning applications on text, including natural language processing, topic modeling, document classification, and clustering; and Weka, a collection of machine learning algorithms to use for data mining tasks.

R is an open-source language mainly used for statistical computing, popular in academia but growing in industry due to data science. It has key machine learning packages like caret for predictive modeling. C++ is widely used for machine learning in games and robotics, especially in embedded systems. Its efficiency makes it ideal for hardware developers and engineers. Some important C++ machine learning libraries are mlpack for scalability, Dlib for various algorithms, and Shark for modular open-source solutions.

CHAPTER 2
LITERATURE SURVEY

Gross Domestic Product’s growth rate is treated as a sign of the economic health of the country. A number of studies demonstrates the factors for prediction of GDP using various methodologies. The GDP data ranging from the year 1989 to 2007 of Anhui region in particular was studied by Gang Long [1]. The method depicts the comparative performance of the GASVM and RBF neural networks respectively. Jaehyun Yoon [2] explored the Gross Domestic Growth of Japan from the year 2001 to 2018.The data is collected from International Monetary Fund and Bank of Japan. The author worked with gradient boosting and random forest machine learning classifier. MAPE and RMSE method are taken into consideration for the purpose of measuring accuracy of the model. Further, cross validation and hyper parameter tuning are used for the creation of more accurate models. The vector machine was trained with genetic algorithm and henceforth used for GDP forecasting. Relative error method was used to evaluate the model performance. The author concluded that in SVM, optimal solution in short time was acquired by genetic algorithm which worked as a better approach in parameters selection of SVM.

For optimizing the support vector machine's parameters, Genetic algorithm was introduced. Various Economic Indicators play a vital role in Gross Domestic Product prediction. Consumption is normally the largest GDP component in the economy. John [3] coined Real Government Consumption Expenditures, Real Personal Consumption Expenditures and Gross Private Domestic Investment as more vital indicators for predicting GDP. Autoregressive approach predicts consistent future growth in terms of factors related to GDP but fails to overcome historic economic recession. Shelly and Wallace [4] studied the relation between M1 money, real GDP and inflation in Mexico. Annual data from the year 1944 to 1991 is studied.

This work indicates that a positive effect on real Gross Domestic Product growth is obtained by unpredictable increases in differenced inflation while predictable increases in differenced inflation results in negative impact on real Gross Domestic Product growth.

In order to produce short-term forecasts of real Austrian GDP, Schneider M. and Spitzer M [5] utilized a generalized dynamic factor model. Macro-Economic Variables has a great influence in country’s GDP. Amongst factors like service, agricultural and livestock sector, business sector and industrial sector proves to be dominating one as far as contribution to the GDP is concerned [6]. The influence of small medium enterprises was described by author Maciej Woźniak [7] stating small medium enterprises plays key social role as they reduce unemployment. Carlos Encinas- Ferrer [8] stated why Foreign Direct Investment does not show as an independent variable since FDI has small proportion within the national investment in the countries like Brazil, China, Peru, Mexico and therefore lead to low multiplier effect on the national economy.

CHAPTER 3
SYSTEM ANALYSIS

3.1 Existing System:

GDP is the most closely monitored and crucial economic indicator for economists as well as investors. GDP is the monetary value of all the manufactured commodities produced within a country's borders in a specific period by the country's citizens and foreigners. It is basically used to determine a country's economic health.

There are limited data modeling techniques in existing system to analyse large data set.

3.1.1 Disadvantages:

- Lack of analyzing models.
- Low accuracy of GDP analysis.

CHAPTER 4
PROPOSED SYSTEM

The aim of this study is to predict GDP, using linear regression and random forest for a particular period. Prediction of GDP involves application of applied mathematics and mathematical model to predict future developments within the economy.

4.1 Advantages:

It permits to review previous economic movements and predict however current economic changes can amend the patterns of previous trend; therefore, a more accurate prediction would provide a significance facilitate to the government in setting up economic development goals, ways and policies. Consequently, a correct Gross Domestic Product prediction presents a number one insight associate an understanding for future economics' trend.

CHAPTER 5
SYSTEM SPECIFICATION:

5.1 HARDWARE REQUIREMENTS:

• System	:	Pentium I3 processor/ Intel Core Processor
• Hard Disk	:	40 GB.
• Ram	:	512 Mb.

5.2 SOFTWARE REQUIREMENTS:

• Operating system	:	Windows family
• Coding Language	:	Python. Front-End

CHAPTER 6
SYSTEM STUDY

6.1 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are,

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- SOCIAL FEASIBILITY

6.2 ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had be purchased.

6.3 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modes requirement, as only minimal or null changes are required for implementing this system.

6.4 SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity.

The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

CHAPTER 7
SOFTWARE ENVIRONMENT

7.1 INTRODUCTION

Python is developed by **Guido van Rossum**. Guido van Rossum started implementing Python in 1989. Python is a very simple programming language so even if you are new to programming, you can learn python without facing any issues.

Python is a high-level, object-oriented programming language used in coding, created by Guido van Rossum in 1991.[1] Python puts readability at a high standard and this makes it great for both programmers and non-programmers to learn. Python is cross-platform, which means you can run it on all major platforms like Microsoft Windows, Linux, and Mac OS X. Python is open-source software and, as a result, has a large community of developers who help improve and contribute to the language. Currently, the main implementation of Python, C Python, is managed by the Python Software Foundation, a non-profit organization working to develop and maintain the Python standards.

Python Software Foundation, a non-profit organization working to develop and maintain the Python standards.

Python gives you the ability to rapidly develop projects, while being able to maintain them at the same time. Python usually uses less lines of codes than other object-oriented languages, like C++ and Java, and it has a simple and easy syntax. A simple "Hello, world!" can be done with just one line of code (actually this is called a command) and only four lines when recreating it as a GUI!

This course is part of the School of Computer Science. For a complete list of resources, see the Computer Science Course Listing.

Please help develop this resource by:

Contributing and giving your knowledge by editing this resource. Giving a teacher or contributor some feedback about this resource. Grammar and spell corrections are appreciated on this resource. Joining a parallel effort at Mat lab and Octave.

7.2 Features of Python programming language

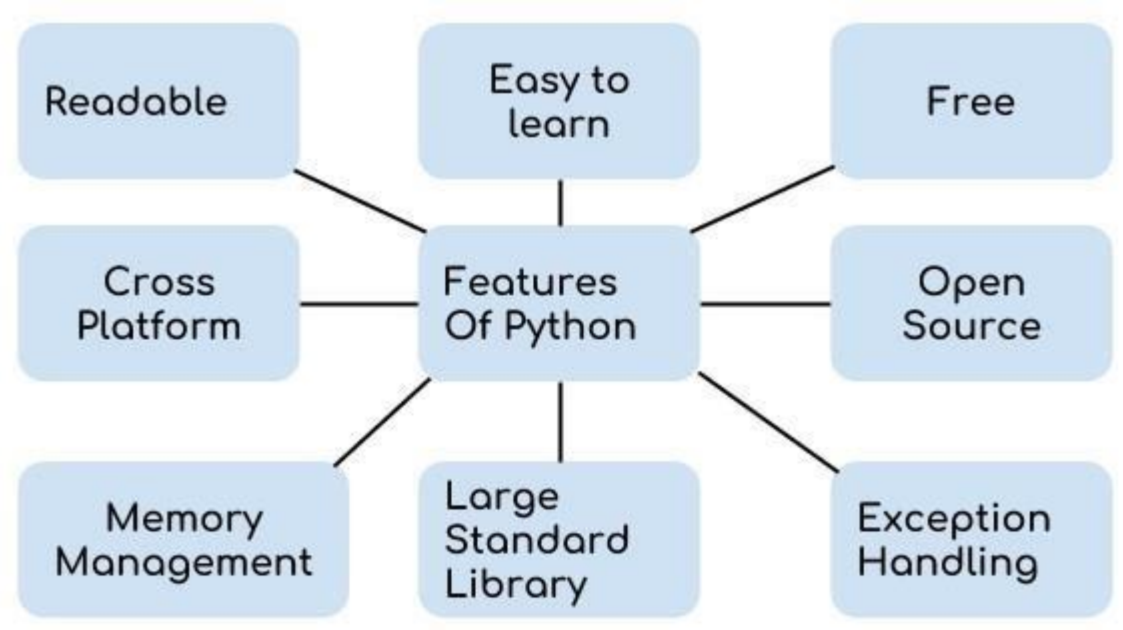


Fig No: 7.1(Features of Python)

Readable: Python is a very readable language.

Easy to Learn: Learning python is easy as this is a expressive and high level programming language, which means it is easy to understand the language and thus easy to learn.

Cross platform: Python is available and can run on various operating systems such as Mac, Windows, Linux, Unix etc. This makes it a cross platform and portable language.

Open Source: Python is an open-source programming language.

Large standard library: Python comes with a large standard library that has some handy codes and functions which we can use while writing code in Python.

Free: Python is free to download and use. This means you can download it for free and use it in your application. See: Open Source Python License. Python is an example of a FLOSS (Free/Libre Open Source Software), which means you can freely distribute copies of this software, read its source code and modify it.

Supports exception handling: If you are new, you may wonder what is an exception? An exception is an event that can occur during program exception and can disrupt the normal flow of program. Python supports exception handling which means we can write less error prone code and can test various scenarios that can cause an exception later on.

Advanced features: Supports generators and list comprehensions. We will cover these features later.

Automatic memory management: Python supports automatic memory management which
means the memory is cleared and freed automatically. You do not have to bother clearing the memory.

What Can You Do with Python?

You may be wondering what all are the applications of Python. There are so many applications of Python, here are some of the them.

1. Web development – Web framework like Django and Flask are based on Python. They help you write server-side code which helps you manage database, write backend programming logic, mapping urls etc.

2. Machine learning – There are many machine learning applications written in Python. Machine learning is a way to write a logic so that a machine can learn and solve a particular problem on its own. For example, products recommendation in websites like Amazon, Flipkart, eBay etc. is a machine learning algorithm that recognised user's interest. Face recognition and Voice recognition in your phone is another example of machine learning.

Data Analysis – Data analysis and data visualization in form of charts can also be developed using Python.

3. Scripting – Scripting is writing small programs to automate simple tasks such as sending automated response emails etc. Such type of applications can also be written in Python programming language.
4. Game development – You can develop games using Python.
5. You can develop Embedded applications in Python.
6. Desktop applications – You can develop desktop application in Python using library like TKinter or QT.

7.3 Python Installation

What is Python?

Python is a high-level scripting language which can be used for a wide variety of text processing, system administration and internet-related tasks. Unlike many similar languages, it's core language is very small and easy to mas-ter, while allowing the addition of modules to perform a virtual limitless variety of tasks. Python is a true object-oriented language, and is available on a wide variety of platforms. There's even a python interpreter written entirely in Java, further enhancing python's position as an excellent solution for internet-based problems.

Python was developed in the early 1990's by Guido van Rossum, then at CWI in Amsterdam, and currently at CNRI in Virginia. In some ways, python grew out of a project to design a computer language which would be easy for beginners to learn, yet would be powerful enough for even advanced users. This heritage is reflected in python's small, clean syntax and the thoroughness of the implementation of ideas like object-oriented programming, without eliminating the ability to program in a more traditional style. So python is an excellent choice as a first programming language without sacrificing the power and advanced capabilities that users will eventually need.

Although pictures of snakes often appear on python books and websites, the name is derived from Guido van Rossum's favorite TV show, "Monty Python's Flying Circus". For this reason, lots of online and print documentation for the language has a light and humorous touch. Interestingly, many experienced programmers report that python has brought back a lot of the fun they used to have programming, so van Rossum's inspiration may be well expressed in the language itself.

The very Basics of Python

There are a few features of python which are different than other programming languages, and which should be mentioned early on so that subsequent examples don't seem confusing. Further information on all of these features will be provided later, when the topics are covered in depth. Python statements do not need to end with a special character – the python interpreter knows that you are done with an individual statement by the presence of a newline, which will be generated when you press the "Return" key of your keyboard. If a statement spans more than one line, the safest course of action is to use a backslash (\) at the end of the line to let python know that you are going to continue the statement on the next line; you can continue using backslashes on additional continuation lines. (There are situations where the backslashes are not needed which will be discussed later.)

Python provides you with a certain level of freedom when composing a program, but there are some rules which must always be obeyed. One of these rules, which some people find very surprising, is that python uses in-dentation (that is, the

amount of white space before the statement itself) to indicate the presence of loops, instead of using delimiters like curly braces ({}) or keywords (like “begin” and “end”) as in many other languages. The amount of indentation you use is not important, but it must be consistent within a given depth of a loop, and statements which are not indented must begin in the first column.
Most python programmers prefer to use an edi-tor like emacs, which automatically provides consistent indentation; you will probably find it easier to maintain your programs if you use consistent in-dentation in every loop, at all depths, and an intelligent editor is very useful in achieving this.

Invoking Python

There are three ways to invoke python, each with its’ own uses. The first way is to type “python” at the shell command prompt. This brings up the python interpreter with a message similar to this one:

```
Python 2.2.1 (#2, Aug 27 2002, 09:01:47)

[GCC 2.95.4 20011002 (Debian prerelease)] on linux2

Type "help", "copyright", "credits" or "license" for more information.
```

The three greater-than signs (>>>) represent python’s prompt; you type your commands after the prompt, and hit return for python to execute them. If you’ve typed an executable statement, python will execute it immediately and display the results of the statement on the screen. For example, if I use python’s print statement to print the famous “Hello, world” greeting, I’ll immediately see a response:

```
>>>     print ’hello,world’ hello,world
```

The print statement automatically adds a newline at the end of the printed string.

This is true regardless of how python is invoked. (You can suppress the newline by following the string to be printed with a comma.)

When using the python interpreter this way, it executes statements im-mediately, and, unless the value of an expression is assigned to a variable (See Section 6.1), python will display the value of that expression as soon as it’s typed. This makes python a very handy calculator:

```
>>>     cost = 27.00
>>>      taxrate = .075

>>>      cost * taxrate 2.025

>>> 16+25+92*3 317
```

When you use python interactively and wish to use a loop, you must, as always, indent the body of the loop consistently when you type your statements. Python can’t execute your statements until the completion of the loop, and as a reminder, it changes its prompt from greater-than signs to periods. Here’s a trivial loop that prints each letter of a word on a separate line — notice the change in the prompt, and that python doesn’t respond until you enter a completely blank line.

```
>>>       word = ’python’ >>> for i in word:
```

...print i... python The need incompletely blank line is peculiar to the interactive use of python. In other settings, simply returning to the previous level of indentation informs python that you’re closing the loop.

You can terminate an interactive session by entering the end-of-file character appropriate to your system (control-Z for Windows, control-D for Unix), or by entering import sys sys.exit() or raise SystemExit at the python prompt.

For longer programs, you can compose your python code in the editor of your choice, and execute the program by either typing “python”, followed by the name of the file containing your program, or by clicking on the file’s icon, if you’ve associated the suffix of your python file with the python interpreter. The file extension most commonly used for python files is

“.py”. Under UNIX systems, a standard technique for running programs written in languages like python is to include a specially formed comment as the first line of the file, informing the shell where to find the interpreter for your program. Suppose that python is installed as /usr/local/bin/python on your system. (The UNIX command “which python” should tell you where python is installed if it’s not
in /usr/local/bin.) Then the first line of your python program, starting in column 1, should look like this:

```
#!/usr/local/bin/python
```

After creating a file, say myprogram.py, which contains the special comment as its first line, you would make the file executable (through the UNIX com-mand “chmod +x myprogram.py”), and then you could execute your pro-gram by simply typing “myprogram.py” at the UNIX prompt.
When you’re running python interactively, you can instruct python to ex-ecute files containing python programs with the execfile function. Suppose that you are using python interactively, and wish to run the program you’ve stored in the file myprog.py. You could enter the following statement:

```
execfile("myprog.py")
```

The file name, since it is not an internal python symbol (like a variable name or keyword), must be surrounded by quotes.

7.4 BASIC PRINCIPLES OF PYTHON

Python has many features that usually are found only in languages which are much more complex to learn and use. These features were designed into python from its very first beginnings, rather than being accumulated into an end result, as is the case with many other scripting languages. If you’re new to programming, even the basic descriptions which follow may seem intimidating. But don’t worry – all of these ideas will be made clearer in the chapters which follow. The idea of presenting these concepts now is to make you aware of how python works, and the general philosophy behind python programming. If some of the concepts that are introduced here seem abstract or overly complex, just try to get a general feel for the idea, and the details will be fleshed out later.

Basic Core Language

Python is designed so that there really isn’t that much to learn in the basic language. For example, there is only one basic structure for conditional programming (if/else/elif), two looping commands (while and for), and a consistent method of handling errors (try/except) which apply to all python programs. This doesn’t mean that the language is not flexible and powerful, however. It simply means that you’re not confronted with an overwhelming choice of options at every turn, which can make programming a much simpler task.

Modules

Python relies on modules, that is, self-contained programs which define a variety of functions and data types, that you can call in order to do tasks be-yond the scope of the basic core language by using the import command. For example, the core distribution of python contains modules for processing files, accessing your computer’s operating system and the internet, writing CGI scripts (which handle communicating with pages displayed in web browsers), string handling and many other tasks. Optional modules, available on the Python web site (http://www.python.org), can be used to create graphical user interfaces, communicate with data bases, process image files, and so on. This structure makes it easy to get started with python, learning specific skills only as you need them, as well as making python run more efficiently by not always including every capability in every program.

Object Oriented Programming

Python is a true object-oriented language. The term “object oriented” has become quite a popular buzzword; such high profile languages as C++ and Java are both object oriented by design. Many other languages add some object-oriented capabilities, but were not designed to be object oriented from the ground up as python was. Why is this feature important? Object oriented program allows you to focus on the data you’re interested in, whether it’s employee information, the results of a scientific experiment or survey, setlists for your favorite band, the contents of your CD collection, information entered by an internet user into a search form or shopping cart, and to develop methods to deal efficiently with your data. A basic concept of object-oriented programming is encapsulation, the ability to define an object that contains your data and all the information a program needs to operate on that data. In this way, when you call a function (known as a method in object-oriented lingo), you don’t need to specify a lot of details about your data, because your data object “knows” all about itself. In addition, objects can inherit from other objects, so if you or someone else has designed an object that’s very close to one you’re interested in, you only have to construct those methods which differ from the existing object, allowing you to save a lot of work.

Another nice feature of object-oriented programs is operator overloading. What this means is that the same operator can have different meanings

BASIC PRINCIPLES OF PYTHON

when used with different types of data. For example, in python, when you’re dealing with numbers, the plus sign (+) has its usual obvious meaning of addition. But when you’re dealing with strings, the plus sign means to join the two strings together. In addition to being able to use overloading for built-in types (like numbers and strings), python also allows you to define what operators mean for the data types you create yourself.

Perhaps the nicest feature of object-oriented programming in python is that you can use as much or as little of it as you want. Until you get comfortable with the ideas behind object-oriented programming, you can write more traditional programs in python without any problems.

Namespaces and Variable Scoping

When you type the name of a variable inside a script or interactive python session, python needs to figure out exactly what variable you're using. To prevent variables you create from overwriting or interfering with variables in python itself or in the modules you use, python uses the concept of multiple namespaces. Basically, this means that the same variable name can be used in different parts of a program without fear of destroying the value of a variable you're not concerned with. To keep its bookkeeping in order, python enforces what is known as the LGB rule. First, the local namespace is searched, then the global names-pace, then the namespace of python built-in functions and variables. A local namespace is automatically created whenever you write a function, or a module containing any of functions, class definitions, or methods. The global namespace consists primarily of the variables you create as part of the "top-level" program, like a script or an interactive session. Finally, the built-in namespace consists of the objects which are part of python's core. You can see the contents of any of the namespaces by using the dir command:

```
>>> dir()
['__builtins__', '__doc__', '__name__']
>>> dir(__builtins__)  '__doc__','__import__', '__name__', 'abs', 'apply', 'callable',
'chr', 'cmp', 'coerce',
'compile', 'complex', 'delattr', 'dir', 'divmod', 'eval', 'execfile', 'filter', 'float',
'getattr', 'globals', 'hasattr', 'hash', 'hex', 'id', 'input', 'int', 'intern', 'isinstance',
```

’issubclass’, ’len’, ’list’, ’locals’, ’long’, ’map’, ’max’, ’min’, ’oct’, ’open’, ’ord’, ’pow’,’range’, ’raw_input’, ’reduce’, ’reload’, ’repr’, ’round’, ’setattr’, ’slice’, ’str’, ’tuple’, ’type’, ’vars’, ’xrange’]

The_builtins__namespace contains all the functions, variables and ex-ceptions which are part of python’s core.

To give controlled access to other namespaces, python uses the import statement. There are three ways to use this statement. In its simplest form, you import the name of a module; this allows you to specify the various objects defined in that module by using a two level name, with the mod-ule’s name and the object’s name separated by a period. For example, the string module (Section 8.4) provides many functions useful for dealing with character strings. Suppose we want to use the split function of the string module to break up a sentence into a list containing separate words. We could use the following sequence of statements:

```
>>> import string
>>> string.split(’Welcome to the Ministry of Silly Walks’) [’Welcome’,’to’, ’the’, ’Ministry’, ’of’, ’Silly’, ’Walks’]
```

If we had tried to refer to this function as simply “split”, python would not be able to find it. That’s because we have only imported the string module into the local namespace, not all of the objects defined in the module. (See below for details of how to do that.)

The second form of the import statement is more specific; it specifies the individual objects from a module whose names we want imported into the local namespace. For example, if we only needed the two functions split and join for use in a program, we could import just those two names directly into the local namespace, allowing us to dispense with the string. prefix:

```
>>> from string import split, join
>>> split (’Welcome to the Ministry of Silly Walks’) [’Welcome’, ’to’,’the’, ’Ministry’, ’of’, ’Silly’, ’Walks’]
```

This technique reduces the amount of typing we need to do, and is an efficient way to bring just a few outside objects into the local environment.
Finally, some modules are designed so that you're expected to have top-level access to all of the functions in the module without having to use the module name as a prefix. In cases like this you can use a statement like:

```
>>> from string import *
```

Now all of the objects defined in the string module are available directly in the top-level environment, with no need for a prefix. You should use this technique with caution, because certain commonly used names from the module may override the names of your variables. In addition, it introduces lots of names into the local namespace, which could adversely affect python's efficiency.

7.5 Exception Handling

Regardless how carefully you write your programs, when you start using them in a variety of situations, errors are bound to occur. Python provides a consistent method of handling errors, a topic often refered to as exception handling. When you're performing an operation that might result in an error, you can surround it with a try loop, and provide an except clause to tell python what to do when a particular error arises. While this is a fairly advanced concept, usually found in more complex languages, you can start using it in even your earliest python programs.

As a simple example, consider dividing two numbers. If the divisor is zero, most programs (python included) will stop running, leaving the user back at a system shell prompt, or with nothing at all. Here's a little python program that illustrates this concept; assume we've saved it to a file called div.py:

```
#!/usr/local/bin/python x = 7
y = 0 print x/y
print "Now we're done!"
```

When we run this program, we don't get to the line which prints the message, because the division by zero is a "fatal" error:

```
% div.py
Traceback (innermost last):
File "div.py", line 5, in ? print x/y
ZeroDivisionError: integer division or modulo
```

While the message may look a little complicated, the main point to notice is that the last line of the message tells us the name of the exception that occured. This allows us to construct an except clause to handle the problem:

```
x = 7
y = 0 try:
print x/y
except ZeroDivisionError:
        print "Oops - I can’t divide by zero, sorry!" print "Now we’re done!"
```

Now when we run the program, it behaves a little more nicely:

```
% div.py
Oops - I can’t divide by zero, sorry! Now we’re done!
```

Since each exception in python has a name, it’s very easy to modify your program to handle errors whenever they’re discovered. And of course, if you can think ahead, you can construct try/except clauses to catch errors before they happen.

CHAPTER 8
INPUT AND OUTPUT DESIGN

8.1 INPUT DESIGN

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

8.2 OBJECTIVES

1) Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

2) It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

3) When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow.

8.3 OUTPUT DESIGN

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.
2. Select methods for presenting information.

3. Create document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following objectives.

- Convey information about past activities, current status or projections of the
- Future.
- Signal important events, opportunities, problems, or warnings.
- Trigger an action.
- Confirm an action.

CHAPTER 9
MODULES

9.1 ADMIN MODULE

- Admin will add dataset to the system for analysis.
- Data will be analysed and provided classification of GDP price valuation on difference factor of dataset.
- Prediction and Graphs will be generated by the system for easy analysis.

CHAPTER 10

IMPLEMENTATION

10.1 SYSTEM ARCHITECTURE

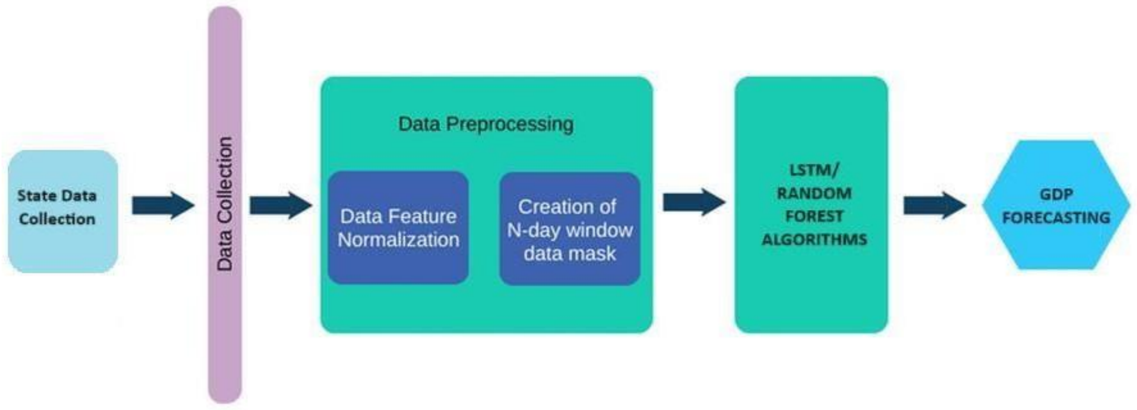

State Data Collection
Data Collection
Data Preprocessing
Data Feature Normalization
Creation of N-day window data mask
LSTM/ RANDOM FOREST ALGORITHMS
GDP FORECASTING

Fig: 10.1 ARCHITECTURE DIAGRAM

A. Data Collection

Data for predicting factors influencing the growth of GDP is taken from Kaggle. The dataset consists of 227 countries data with 20 different factors namely literacy, net migration, population etc.

B. Preprocessing of Data

The goal is to pre-process the data for consecutive step. Therefore, a method is used to handle missing values and outliers. The feature selection was done manually on the basis of literature survey and research. Data cleaning was done by removing the null values followed by removal of outliers by using Interquartile Range (IQR) method. Fig.2. shows the data set after the data pre- processing.

C. Modelling

During this step, various error metrics are used in order to calculate the accuracy of the forecasting model.

D. Output and Usage

This step presents the accuracy graph of the model.

10.2 Linear Regression

It is supervised machine learning algorithm, the most basic type of regression. Basically, it is the mathematical model that analyses the linear relationship between a dependent variable with given set of independent variables(s). In the project the simple linear regression was used to predict the individual attribute of the dataset. For this 80% of the dataset was the training dataset
i.e., used for training the model and remaining 20% was used to test the dataset.

10.3 Random Forest

Random Forest is one of the well-known machine learning algorithms that belongs to supervised learning technique. Random Forest is used both for regression and classification problems in machine learning. Ensemble Learning is a concept in which multiple classifiers are integrated in order to resolve a complicated drawback and hence it improves the performance of the model.
Random Forest relies on the concept of ensemble learning.

As the name itself suggests, "Random Forest is basically a classifier that consists of decision trees of the given dataset on varied subsets. Further, the random forest takes the average in order to improve the forecasting accuracy." Predictions from every tree that is formed are taken into consideration instead of just relying on a single decision tree and after that; based on majority votes of prediction, output is predicted.

11. DATA FLOW DIAGRAM

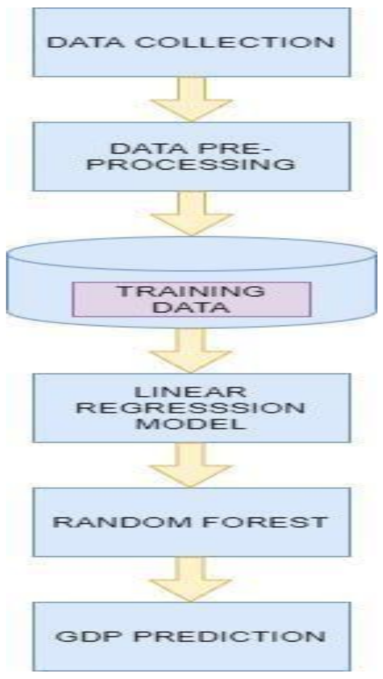


Fig No: 11.1

12. ACTIVITY DIAGRAM

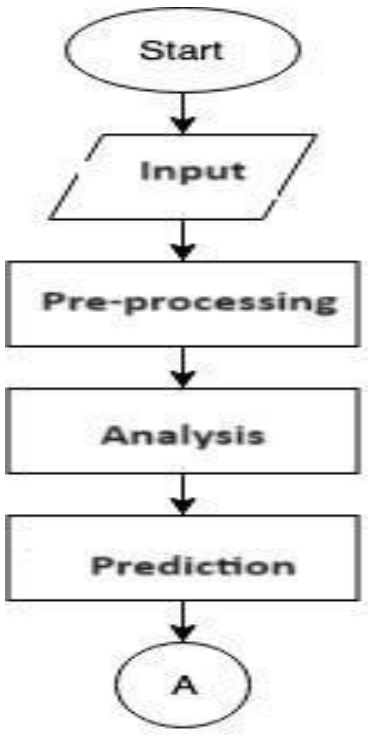


Fig No: 12.1

13. CLASS DIAGRAM

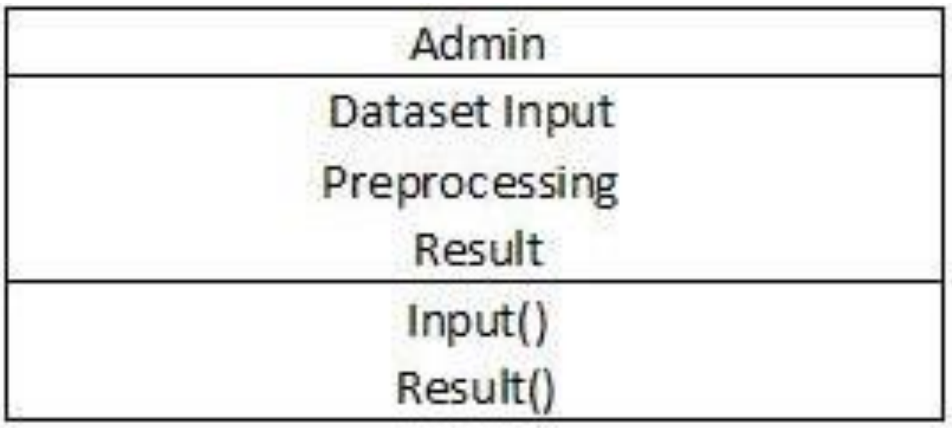


Fig No: 13.1

14. SEQUENCE DIAGRAM

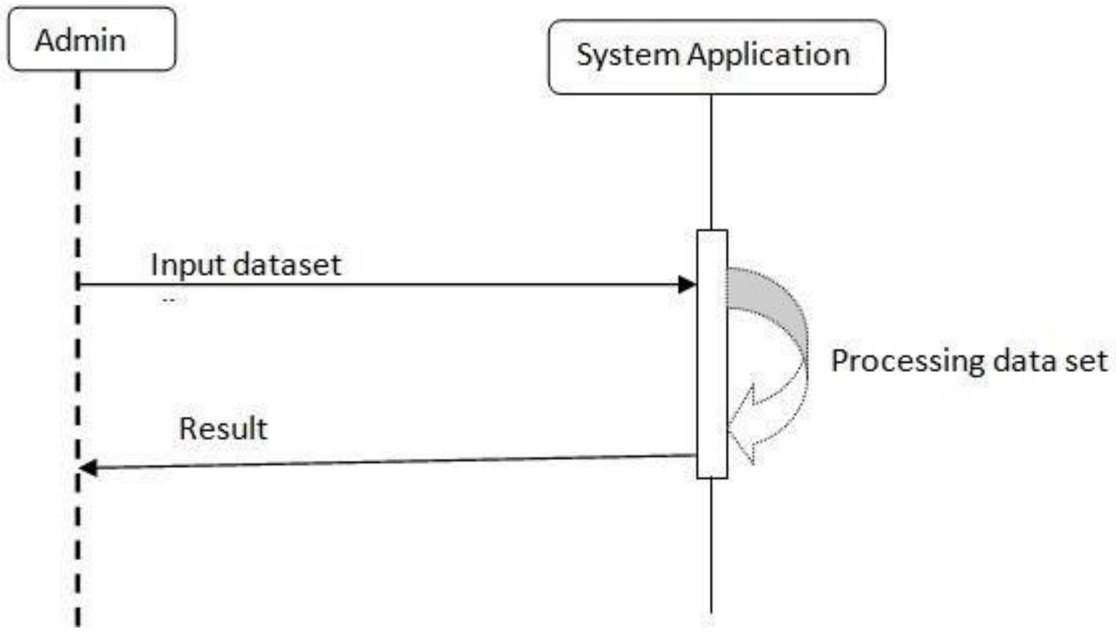


Fig No: 14.1

15. Use case diagram

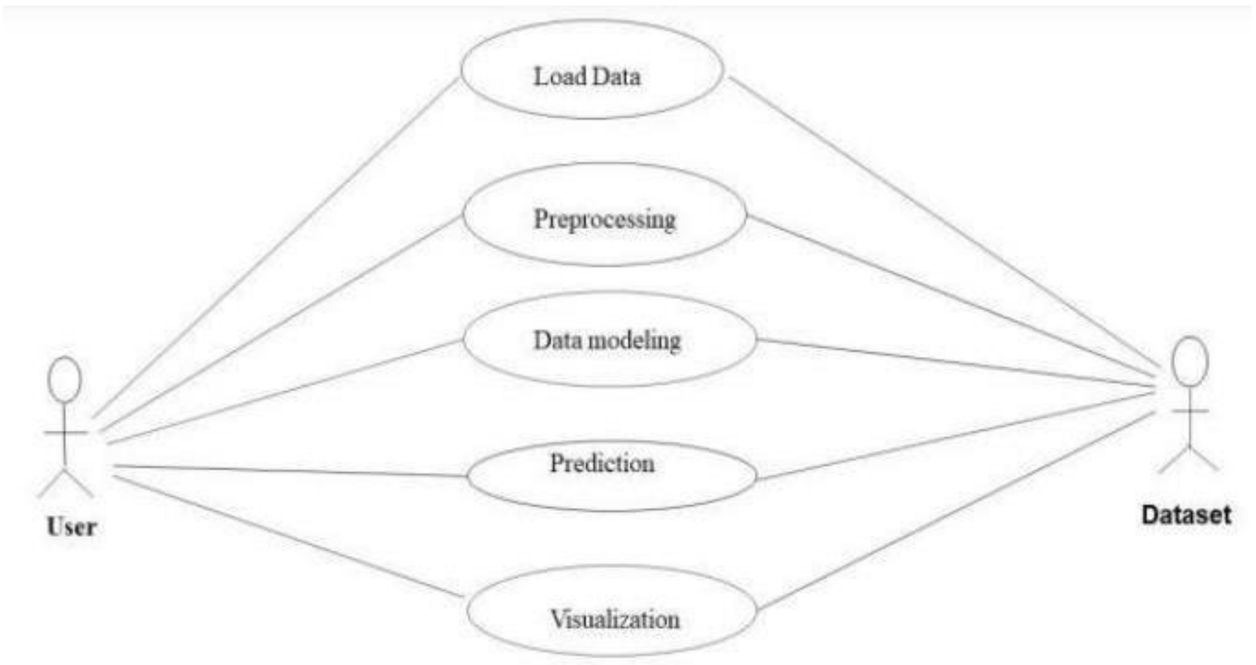


Fig No: 15.1

CHAPTER 16
SYSTEM TEST

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

16.1 TYPES OF TESTS

Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected
results.

Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

<u>Functional test</u>

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input	:	identified classes of valid input must be accepted.
Invalid Input	:	identified classes of invalid input must be rejected.
Functions	:	identified functions must be exercised.
Output	:	identified classes of application outputs must be exercised.
Systems/Procedures	:	interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

<u>System Test</u>

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose.
It is purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box. you cannot "see" into it. The test provides inputs and responds to outputs without considering how the software works.

Unit Testing

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

16.2 TEST STRATEGIES AND APPROACHES

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

E. All field entries must work properly.
F. Pages must be activated from the identified link.
G. The entry screen, messages and responses must not be delayed.

Features to be tested

H. Verify that the entries are of the correct format

I. No duplicate entries should be allowed
J. All links should take the user to the correct page.

Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

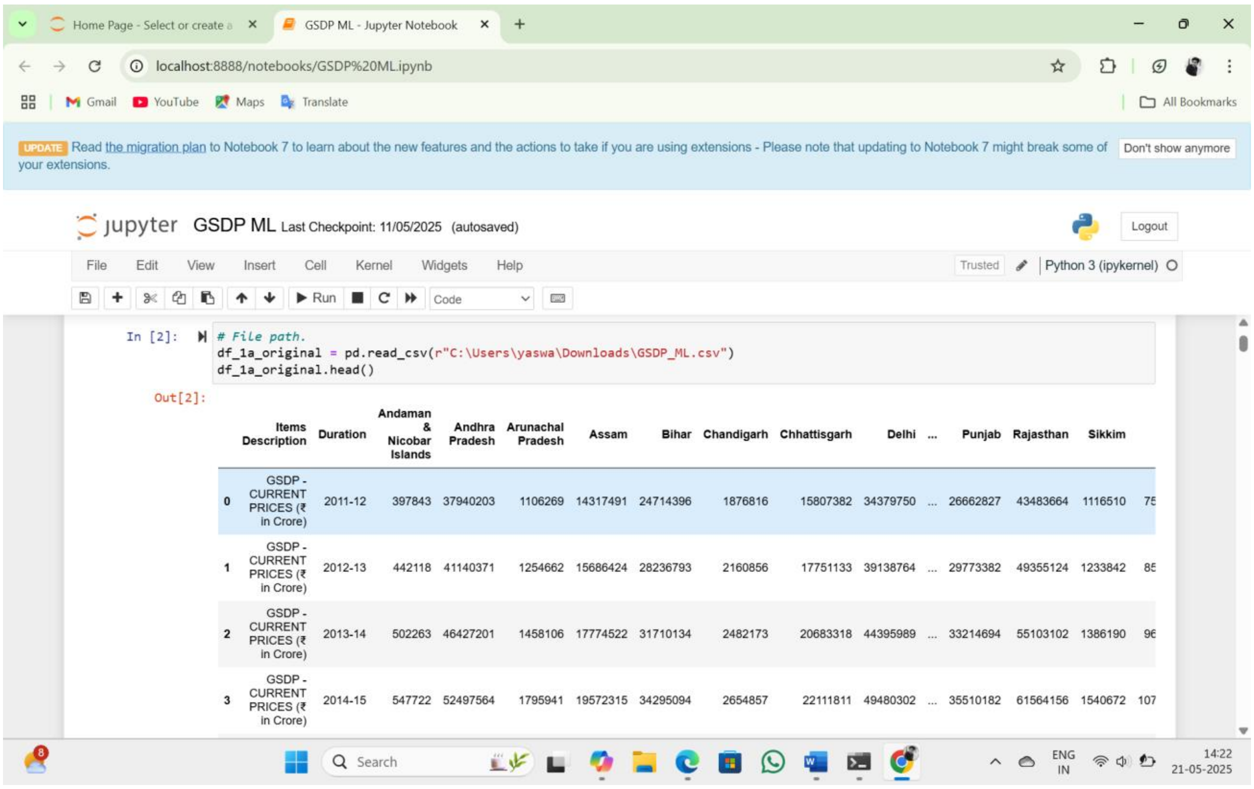
Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

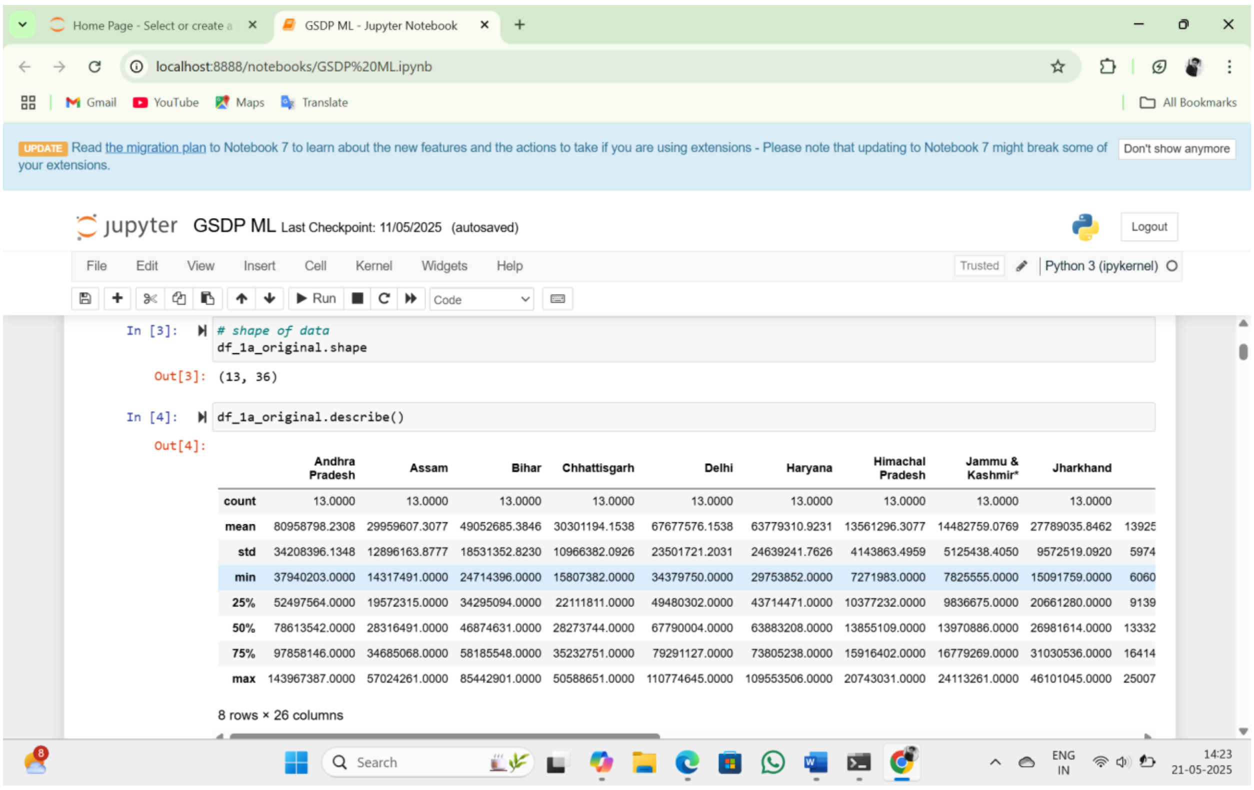
CHAPTER 17

RESULT


Home Page - Select or create a
GSDP ML - Jupyter Notebook
localhost:8888/notebooks/GSDP%20ML.ipynb
Gmail
YouTube
Maps
Translate
All Bookmarks
UPDATE Read the migration plan to Notebook 7 to learn about the new features and the actions to take if you are using extensions - Please note that updating to Notebook 7 might break some of your extensions.
Don't show anymore
jupyter GSDP ML Last Checkpoint: 11/05/2025 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
Search
ENG IN
14:22 21-05-2025

```
In [2]:  # File path.
         df_1a_original = pd.read_csv(r"C:\Users\yaswa\Downloads\GSDP_ML.csv")
         df_1a_original.head()
Out[2]:
```

	Items Description	Duration	Andaman & Nicobar Islands	Andhra Pradesh	Arunachal Pradesh	Assam	Bihar	Chandigarh	Chhattisgarh	Delhi	...	Punjab	Rajasthan	Sikkim	
0	GSDP - CURRENT PRICES (₹ in Crore)	2011-12	397843	37940203	1106269	14317491	24714396	1876816	15807382	34379750	...	26662827	43483664	1116510	75
1	GSDP - CURRENT PRICES (₹ in Crore)	2012-13	442118	41140371	1254662	15686424	28236793	2160856	17751133	39138764	...	29773382	49355124	1233842	85
2	GSDP - CURRENT PRICES (₹ in Crore)	2013-14	502263	46427201	1458106	17774522	31710134	2482173	20683318	44395989	...	33214694	55103102	1386190	96
3	GSDP - CURRENT PRICES (₹ in Crore)	2014-15	547722	52497564	1795941	19572315	34295094	2654857	22111811	49480302	...	35510182	61564156	1540672	107

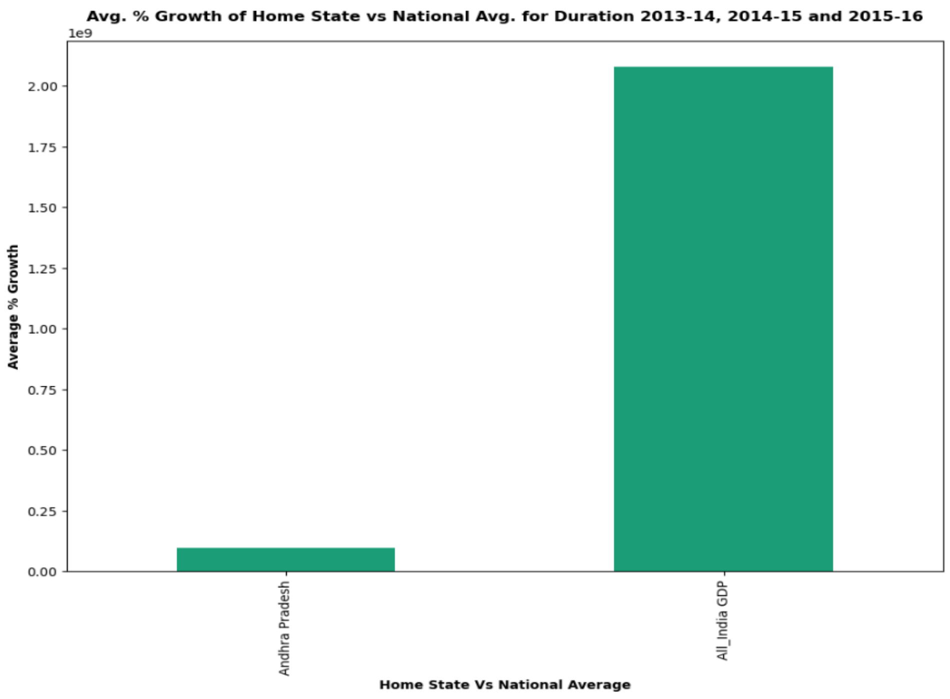

Home Page - Select or create a
GSDP ML - Jupyter Notebook
localhost:8888/notebooks/GSDP%20ML.ipynb
Gmail
YouTube
Maps
Translate
All Bookmarks
UPDATE Read the migration plan to Notebook 7 to learn about the new features and the actions to take if you are using extensions - Please note that updating to Notebook 7 might break some of your extensions.
Don't show anymore
jupyter GSDP ML Last Checkpoint: 11/05/2025 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
Search
ENG IN
14:23 21-05-2025

```
In [3]:  # shape of data
         df_1a_original.shape
Out[3]: (13, 36)

In [4]:  df_1a_original.describe()
Out[4]:
```

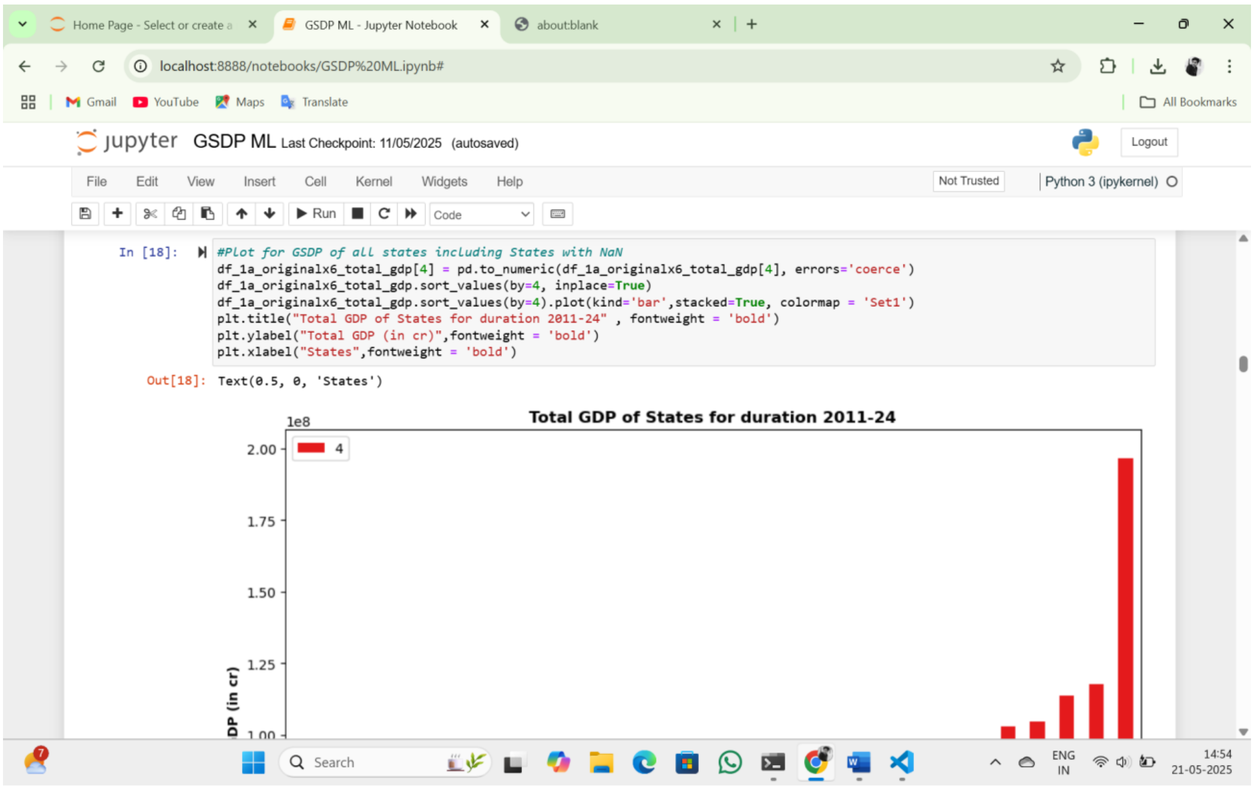
	Andhra Pradesh	Assam	Bihar	Chhattisgarh	Delhi	Haryana	Himachal Pradesh	Jammu & Kashmir*	Jharkhand	
count	13.0000	13.0000	13.0000	13.0000	13.0000	13.0000	13.0000	13.0000	13.0000	
mean	80958798.2308	29959607.3077	49052685.3846	30301194.1538	67677576.1538	63779310.9231	13561296.3077	14482759.0769	27789035.8462	13925
std	34208396.1348	12896163.8777	18531352.8230	10966382.0926	23501721.2031	24639241.7626	4143863.4959	5125438.4050	9572519.0920	5974
min	37940203.0000	14317491.0000	24714396.0000	15807382.0000	34379750.0000	29753852.0000	7271983.0000	7825555.0000	15091759.0000	6060
25%	52497564.0000	19572315.0000	34295094.0000	22111811.0000	49480302.0000	43714471.0000	10377232.0000	9836675.0000	20661280.0000	9139
50%	78613542.0000	28316491.0000	46874631.0000	28273744.0000	67790004.0000	63883208.0000	13855109.0000	13970886.0000	26981614.0000	13332
75%	97858146.0000	34685068.0000	58185548.0000	35232751.0000	79291127.0000	73805238.0000	15916402.0000	16779269.0000	31030536.0000	16414
max	143967387.0000	57024261.0000	85442901.0000	50588651.0000	110774645.0000	109553506.0000	20743031.0000	24113261.0000	46101045.0000	25007

8 rows × 26 columns

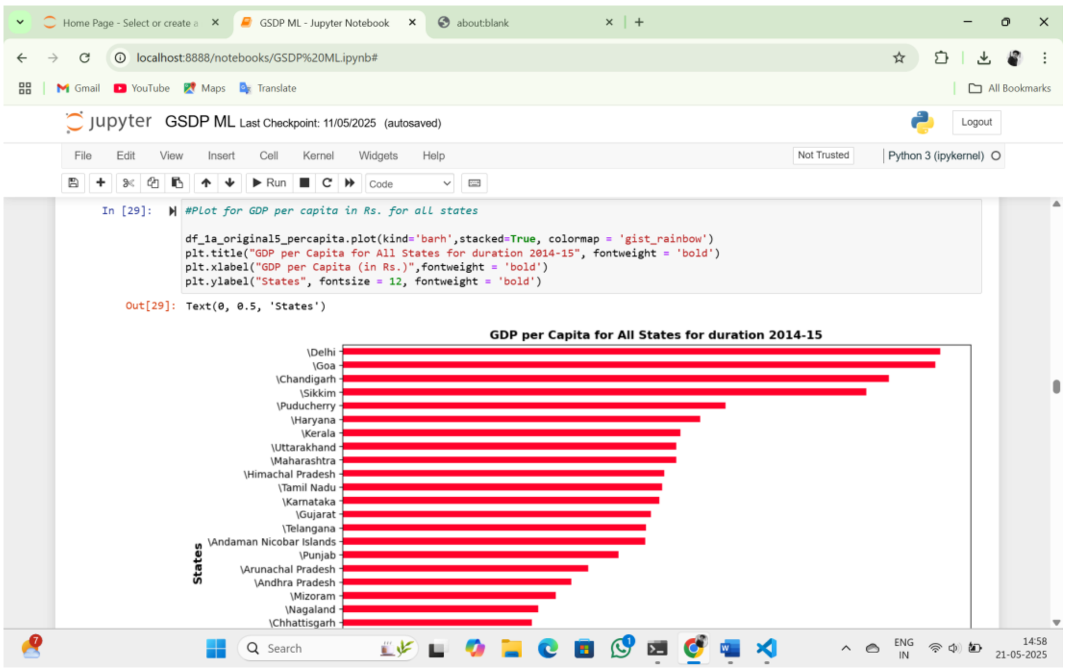
```
In [15]:  df_1a_originalx4_myhome.plot(kind='bar',stacked=True, colormap = 'Dark2')
          plt.title("Avg. % Growth of Home State vs National Avg. for Duration 2013-14,
          plt.ylabel("Average % Growth", fontweight = 'bold')
          plt.xlabel("Home State Vs National Average", fontweight = 'bold')

Out[15]:  Text(0.5, 0, 'Home State Vs National Average')
```



```
In [16]:  #Selecting the GSDP for year 2015-16

          df_1a_originalx5_total_gdp = df_1a_originalx4.iloc[2:,4:5]
```

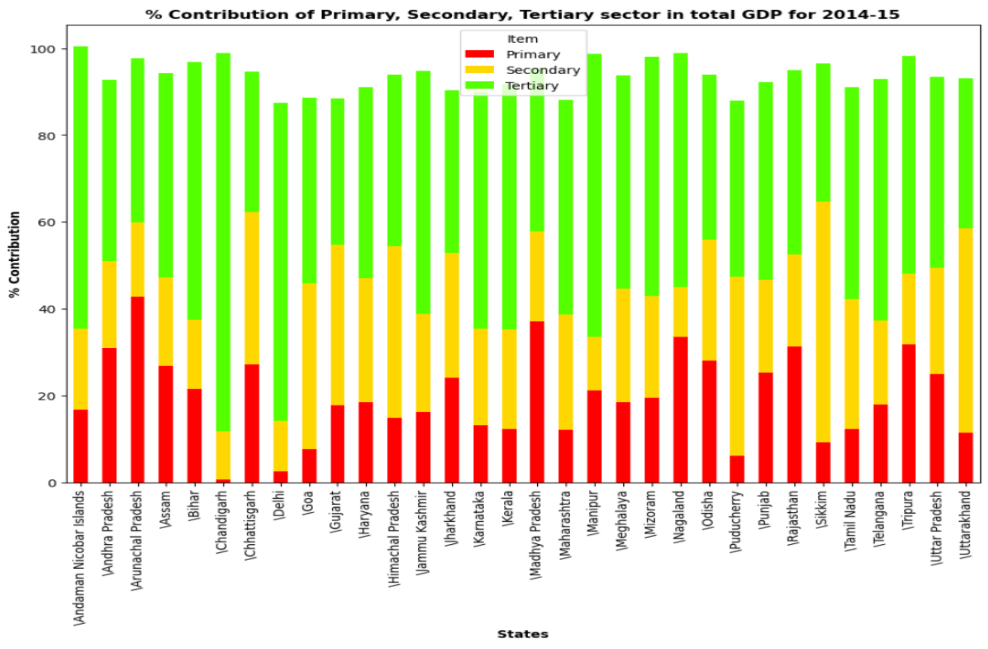


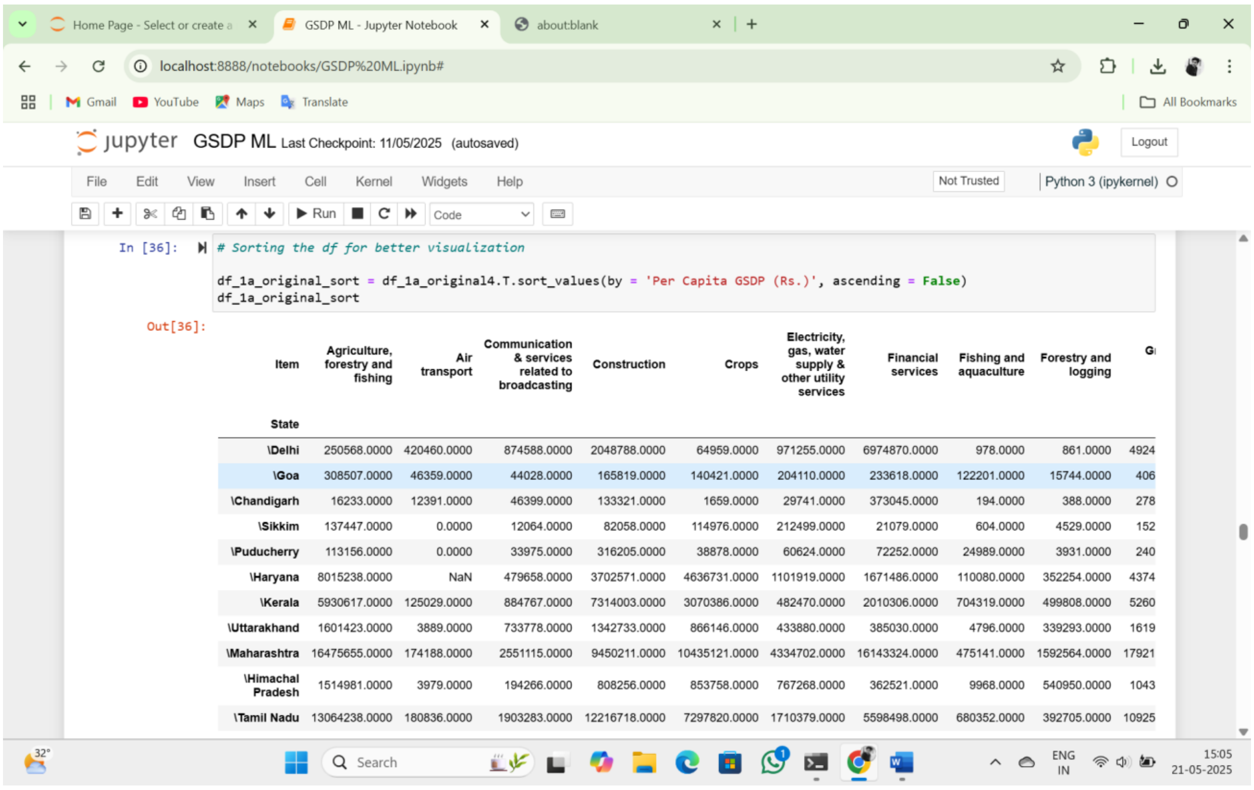
```
In [18]:  #Plot for GSDP of all states including States with NaN
          df_1a_originalx6_total_gdp[4] = pd.to_numeric(df_1a_originalx6_total_gdp[4], errors='coerce')
          df_1a_originalx6_total_gdp.sort_values(by=4, inplace=True)
          df_1a_originalx6_total_gdp.sort_values(by=4).plot(kind='bar',stacked=True, colormap = 'Set1')
          plt.title("Total GDP of States for duration 2011-24" , fontweight = 'bold')
          plt.ylabel("Total GDP (in cr)",fontweight = 'bold')
          plt.xlabel("States",fontweight = 'bold')

Out[18]:  Text(0.5, 0, 'States')
```



```
In [29]:  #Plot for GDP per capita in Rs. for all states

          df_1a_original5_percapita.plot(kind='barh',stacked=True, colormap = 'gist_rainbow')
          plt.title("GDP per Capita for All States for duration 2014-15", fontweight = 'bold')
          plt.xlabel("GDP per Capita (in Rs.)",fontweight = 'bold')
          plt.ylabel("States", fontsize = 12, fontweight = 'bold')

Out[29]:  Text(0, 0.5, 'States')
```

```
In [35]:  # Plot for % contribution of sectors in total GDP

          df_1a_original_gdp_percon.plot(kind='bar',stacked=True, colormap = 'prism')
          plt.title("% Contribution of Primary, Secondary, Tertiary sector in total GDP
          plt.ylabel("% Contribution", fontweight = 'bold')
          plt.xlabel("States", fontweight = 'bold')

Out[35]:  Text(0.5, 0, 'States')
```

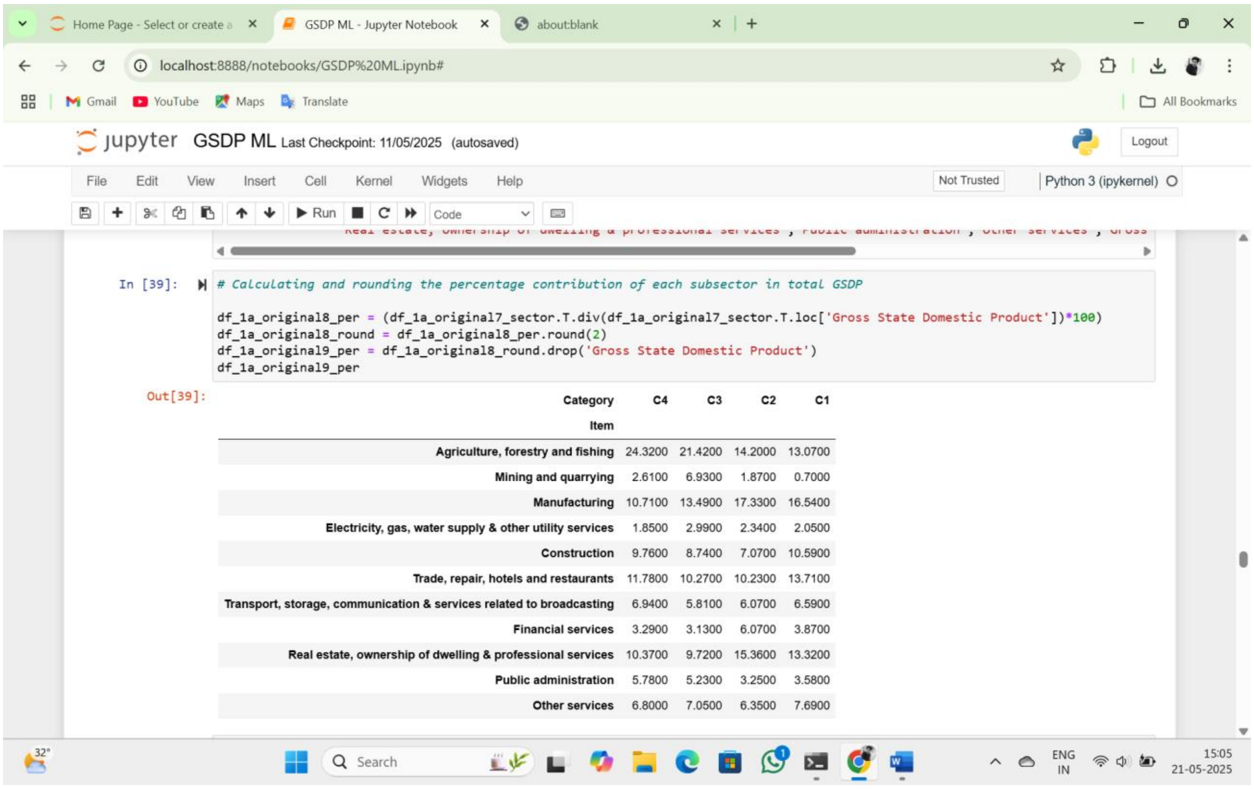




```
In [36]: # Sorting the df for better visualization

         df_1a_original_sort = df_1a_original4.T.sort_values(by = 'Per Capita GSDP (Rs.)', ascending = False)
         df_1a_original_sort
```

Out[36]:

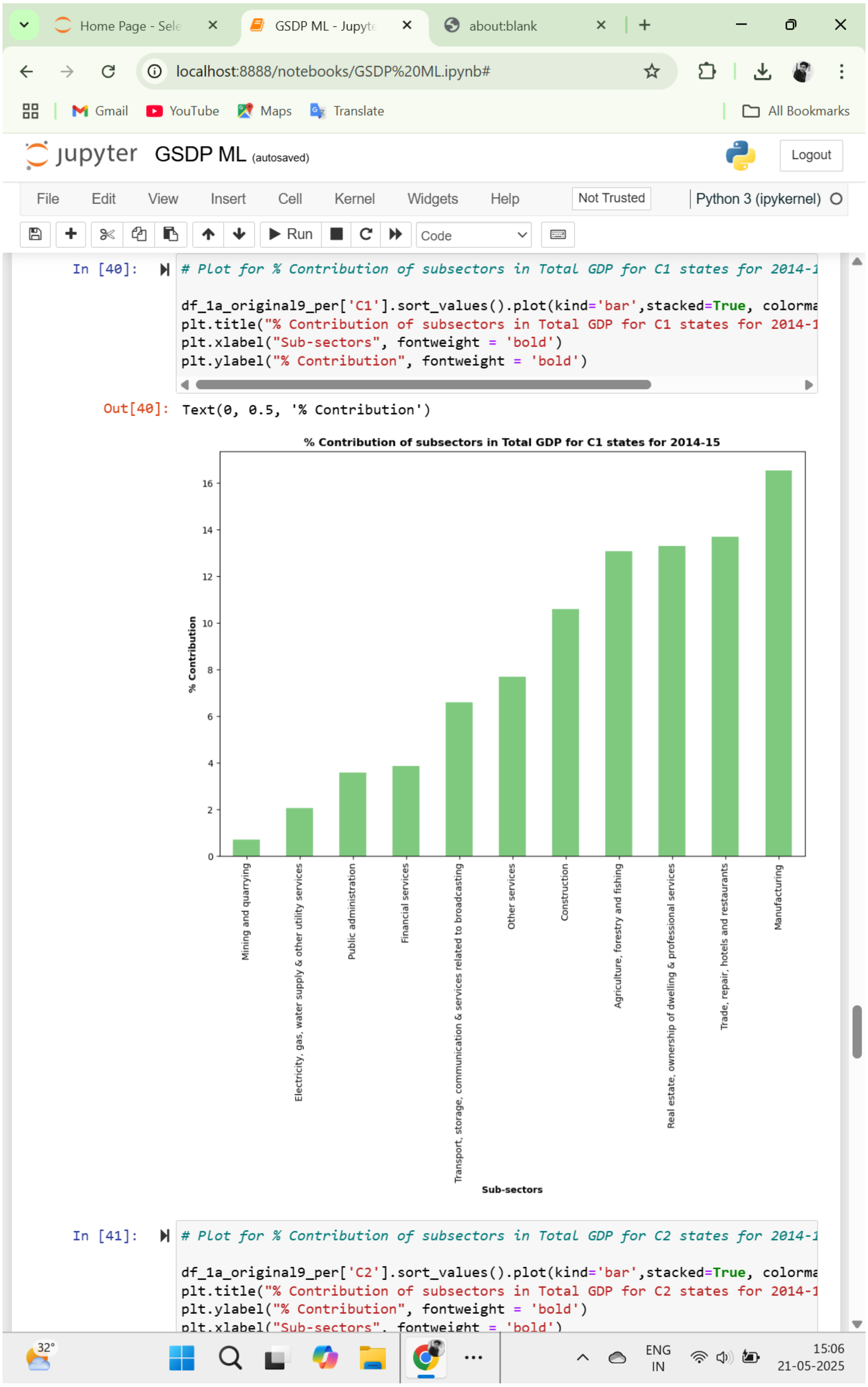
Item	Agriculture, forestry and fishing	Air transport	Communication & services related to broadcasting	Construction	Crops	Electricity, gas, water supply & other utility services	Financial services	Fishing and aquaculture	Forestry and logging	G
State										
\Delhi	250568.0000	420460.0000	874588.0000	2048788.0000	64959.0000	971255.0000	6974870.0000	978.0000	861.0000	4924
\Goa	308507.0000	46359.0000	44028.0000	165819.0000	140421.0000	204110.0000	233618.0000	122201.0000	15744.0000	406
\Chandigarh	16233.0000	12391.0000	46399.0000	133321.0000	1659.0000	29741.0000	373045.0000	194.0000	388.0000	278
\Sikkim	137447.0000	0.0000	12064.0000	82058.0000	114976.0000	212499.0000	21079.0000	604.0000	4529.0000	152
\Puducherry	113156.0000	0.0000	33975.0000	316205.0000	38878.0000	60624.0000	72252.0000	24989.0000	3931.0000	240
\Haryana	8015238.0000	NaN	479658.0000	3702571.0000	4636731.0000	1101919.0000	1671486.0000	110080.0000	352254.0000	4374
\Kerala	5930617.0000	125029.0000	884767.0000	7314003.0000	3070386.0000	482470.0000	2010306.0000	704319.0000	499808.0000	5260
\Uttarakhand	1601423.0000	3889.0000	733778.0000	1342733.0000	866146.0000	433880.0000	385030.0000	4796.0000	339293.0000	1619
\Maharashtra	16475655.0000	174188.0000	2551115.0000	9450211.0000	10435121.0000	4334702.0000	16143324.0000	475141.0000	1592564.0000	17921
\Himachal Pradesh	1514981.0000	3979.0000	194266.0000	808256.0000	853758.0000	767268.0000	362521.0000	9968.0000	540950.0000	1043
\Tamil Nadu	13064238.0000	180836.0000	1903283.0000	12216718.0000	7297820.0000	1710379.0000	5598498.0000	680352.0000	392705.0000	10925



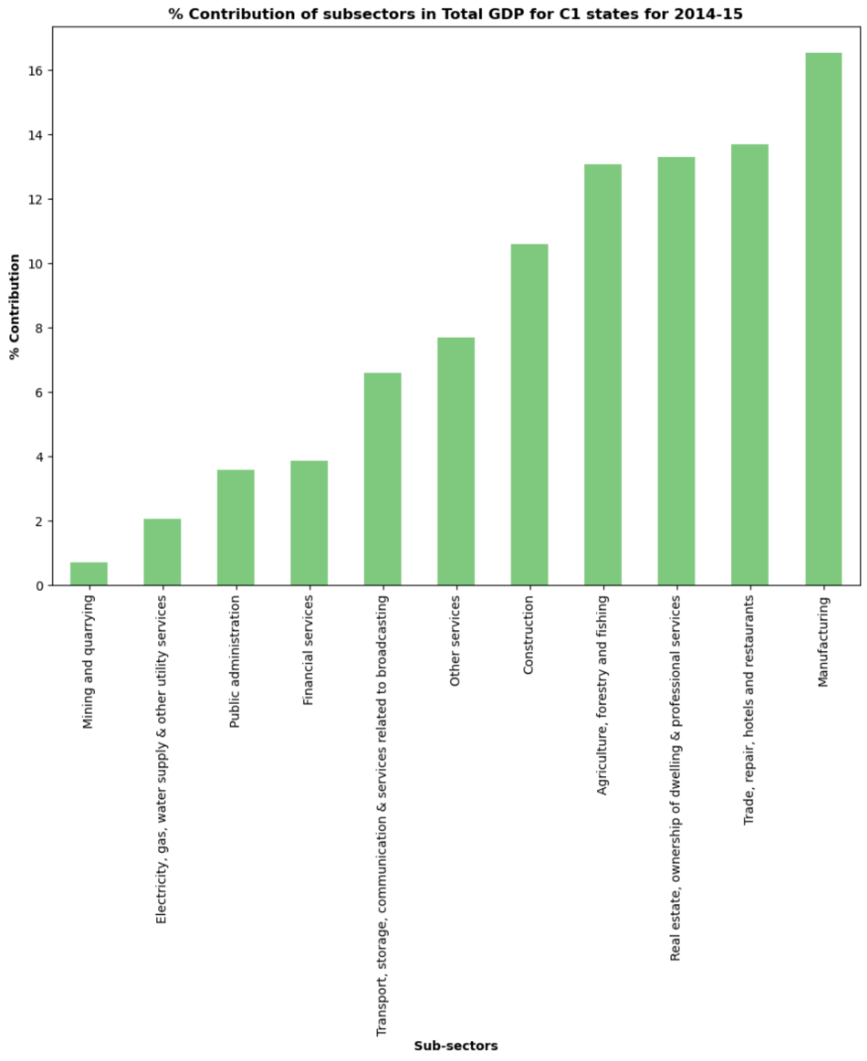
```
In [39]: # Calculating and rounding the percentage contribution of each subsector in total GSDP

         df_1a_original8_per = (df_1a_original7_sector.T.div(df_1a_original7_sector.T.loc['Gross State Domestic Product'])*100)
         df_1a_original8_round = df_1a_original8_per.round(2)
         df_1a_original9_per = df_1a_original8_round.drop('Gross State Domestic Product')
         df_1a_original9_per
```

Out[39]:

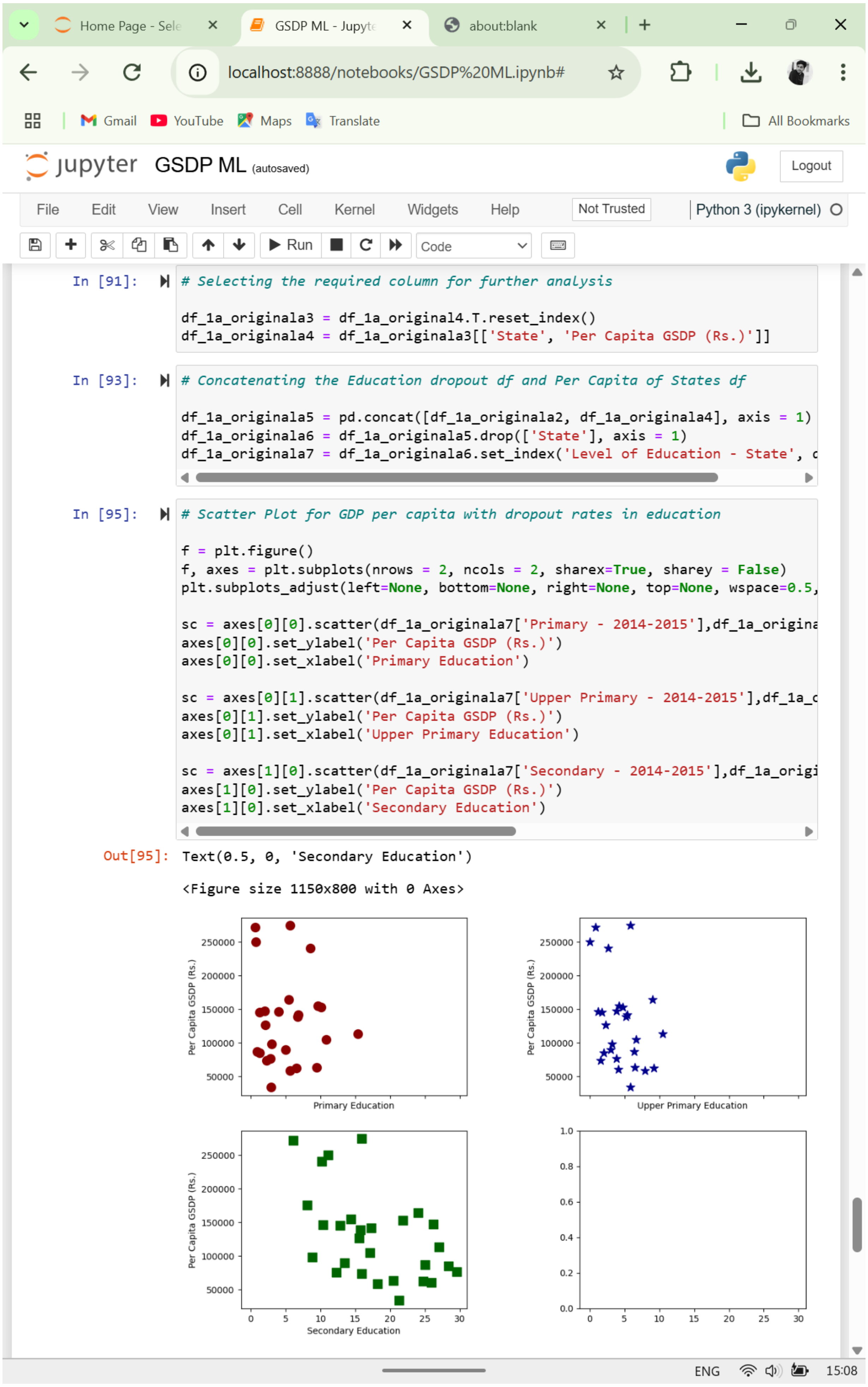
Category	C4	C3	C2	C1
Item				
Agriculture, forestry and fishing	24.3200	21.4200	14.2000	13.0700
Mining and quarrying	2.6100	6.9300	1.8700	0.7000
Manufacturing	10.7100	13.4900	17.3300	16.5400
Electricity, gas, water supply & other utility services	1.8500	2.9900	2.3400	2.0500
Construction	9.7600	8.7400	7.0700	10.5900
Trade, repair, hotels and restaurants	11.7800	10.2700	10.2300	13.7100
Transport, storage, communication & services related to broadcasting	6.9400	5.8100	6.0700	6.5900
Financial services	3.2900	3.1300	6.0700	3.8700
Real estate, ownership of dwelling & professional services	10.3700	9.7200	15.3600	13.3200
Public administration	5.7800	5.2300	3.2500	3.5800
Other services	6.8000	7.0500	6.3500	7.6900



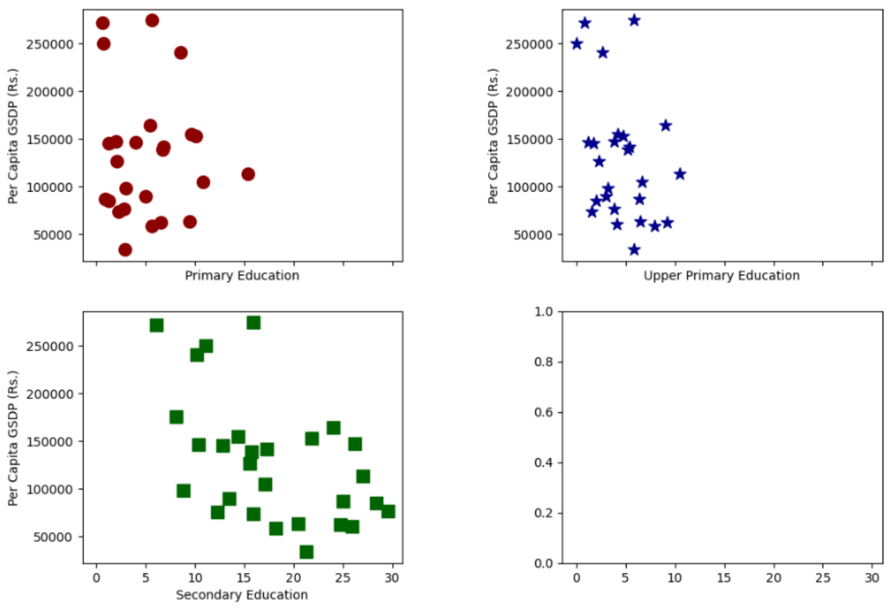
```
In [40]:  # Plot for % Contribution of subsectors in Total GDP for C1 states for 2014-1

          df_1a_original9_per['C1'].sort_values().plot(kind='bar',stacked=True, colorma
          plt.title("% Contribution of subsectors in Total GDP for C1 states for 2014-1
          plt.xlabel("Sub-sectors", fontweight = 'bold')
          plt.ylabel("% Contribution", fontweight = 'bold')

Out[40]:  Text(0, 0.5, '% Contribution')
```



```
In [41]:  # Plot for % Contribution of subsectors in Total GDP for C2 states for 2014-1

          df_1a_original9_per['C2'].sort_values().plot(kind='bar',stacked=True, colorma
          plt.title("% Contribution of subsectors in Total GDP for C2 states for 2014-1
          plt.ylabel("% Contribution", fontweight = 'bold')
          plt.xlabel("Sub-sectors", fontweight = 'bold')
```



```
In [91]:  # Selecting the required column for further analysis

          df_1a_originala3 = df_1a_original4.T.reset_index()
          df_1a_originala4 = df_1a_originala3[['State', 'Per Capita GSDP (Rs.)']]

In [93]:  # Concatenating the Education dropout df and Per Capita of States df

          df_1a_originala5 = pd.concat([df_1a_originala2, df_1a_originala4], axis = 1)
          df_1a_originala6 = df_1a_originala5.drop(['State'], axis = 1)
          df_1a_originala7 = df_1a_originala6.set_index('Level of Education - State', c

In [95]:  # Scatter Plot for GDP per capita with dropout rates in education

          f = plt.figure()
          f, axes = plt.subplots(nrows = 2, ncols = 2, sharex=True, sharey = False)
          plt.subplots_adjust(left=None, bottom=None, right=None, top=None, wspace=0.5,

          sc = axes[0][0].scatter(df_1a_originala7['Primary - 2014-2015'],df_1a_origina
          axes[0][0].set_ylabel('Per Capita GSDP (Rs.)')
          axes[0][0].set_xlabel('Primary Education')

          sc = axes[0][1].scatter(df_1a_originala7['Upper Primary - 2014-2015'],df_1a_c
          axes[0][1].set_ylabel('Per Capita GSDP (Rs.)')
          axes[0][1].set_xlabel('Upper Primary Education')

          sc = axes[1][0].scatter(df_1a_originala7['Secondary - 2014-2015'],df_1a_origi
          axes[1][0].set_ylabel('Per Capita GSDP (Rs.)')
          axes[1][0].set_xlabel('Secondary Education')

Out[95]:  Text(0.5, 0, 'Secondary Education')

          <Figure size 1150x800 with 0 Axes>
```



CHAPTER 18
CONCLUSION

The resultant study encourages the utilization of machine learning classifiers namely linear regression and Random Forest in macroeconomic data forecasting. On the basis of optimization process, the machine learning algorithm "Random Forest" utilized during this study worked well with the accuracy 86 percentage in order to predict the true GDP per capita. Random Forest Classifier produces more accurate forecasts as compared to the linear regression. The main focus of traditional economics models is mainly on explanations of relationships whereas machine learning classifiers target predictions. Though it may seem as Machine learning models do not turn out to be good performers while discovering the impact of independent variable on the dependent variable or analyzing a causal relationship.

SOURCE CODE

```
# Import the required Libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import glob
from functools import reduce
from itertools import cycle, islice
pd.options.display.float_format='{:.4f}'.format
plt.rcParams['figure.figsize'] = [11.5,8]
pd.set_option('display.max_colwidth', 1000)  # Example width
pd.set_option('display.max_colwidth', None)

#Creating the dataframe(Please change the path of the file below before executing the
cell)

df_1a_original = pd.read_csv(r"C:\Users\yaswa\Downloads\GSDP_ML.csv")
df_1a_original.head()

# Reading the relevant file on which Analysis needs to be done

file = path + 'SGDP.csv'
dfx = pd.read_csv(file)
dfx.head(4)

# shape of data

dfx.shape

# Data Information

df_1a_original.info()

# Data Information

dfx.info()

## Data Cleansing and Preparation

# Calculating the Missing Values % contribution in DF
df_1a_original_null=df_1a_original.isna().mean().round(4) * 100
```

```
df_1a_original_null

# Dropping columns where all rows are NaN

df_1a_originalx1 = df_1a_original.dropna(axis = 1, how = 'all')

# Dropping the data for Duration 2016-17 as it will not be used in Analysis

df_1a_originalx2 = df_1a_originalx1[df_1a_originalx1.Duration != '2016-17']

# Dropping the UT as it is not needed for Analysis

df_1a_originalx3 = df_1a_originalx2.T
df_1a_originalx4 = df_1a_originalx3.drop(labels = ['Andaman & Nicobar
Islands','Chandigarh','Delhi','Puducherry'])
#dfx3

# Mean of the row (% Growth over previous year) for duration 2013-14, 2014-15 and
2015-16

df_1a_originalx4_mean = df_1a_originalx4.iloc[2:,
6:10].mean(axis=1).fillna(0).round(2).sort_values()
df_1a_originalx4_mean

## Data Visualization and Insights Extraction

# Bar Plot for Average growth rates of the various states for duration 2013-14, 2014-
15 and 2015-16

plt.rcParams['figure.figsize'] = [11.5,8]
df_1a_originalx4_mean.plot(kind='barh',stacked=True, colormap = 'Set1')
plt.title("Avg.% Growth of States for Duration 2013-14, 2014-15 and 2015-16",
fontweight = 'bold')
plt.xlabel("Avg. % Growth", fontweight = 'bold')
plt.ylabel("States", fontweight = 'bold')

# Average growth rate of my home state against the National average Growth rate

df_1a_originalx4_myhome = df_1a_originalx4_mean[['Andhra Pradesh', 'All_India
GDP']]


df_1a_originalx4_myhome.plot(kind='bar',stacked=True, colormap = 'Dark2')
plt.title("Avg. % Growth of Home State vs National Avg. for Duration 2013-14, 2014-
15 and 2015-16", fontweight = 'bold')
plt.ylabel("Average % Growth", fontweight = 'bold')
plt.xlabel("Home State Vs National Average", fontweight = 'bold')
```

### Insights from the above Plot considering the average growth rates of my Home state Vs National Average for duration 2013-2016
* Average growth rate of my home state Madhya Pradesh(~14%) is greater than the National Average growth rate(~12%).Performance of my state is better as compared to most of rest states because the state is rich in natural resources, fuels, minerals, agriculture and biodiversity *

```
#Selecting the GSDP for year 2015-16

df_1a_originalx5_total_gdp = df_1a_originalx4.iloc[2:,4:5]

# Dropping the GSDP of All_India as it will not be included in the plot

df_1a_originalx6_total_gdp = df_1a_originalx5_total_gdp.drop(labels = ['All_India GDP'])

#Plot for GSDP of all states including States with NaN

df_1a_originalx6_total_gdp[4] = pd.to_numeric(df_1a_originalx6_total_gdp[4], errors='coerce')
df_1a_originalx6_total_gdp.sort_values(by=4, inplace=True)
df_1a_originalx6_total_gdp.sort_values(by=4).plot(kind='bar',stacked=True,
colormap = 'Set1')
plt.title("Total GDP of States for duration 2011-24" , fontweight = 'bold')
plt.ylabel("Total GDP (in cr)",fontweight = 'bold')
plt.xlabel("States",fontweight = 'bold')

# Dropping the States whose GSDP in NaN for year 2015-16

df_1a_originalx7_total_gdp = df_1a_originalx6_total_gdp.dropna().sort_values(by = 4)

#Plot for GSDP of all states excluding States with NaN

df_1a_originalx7_total_gdp.plot(kind='bar',stacked=True, colormap = 'autumn')
plt.title("Total GDP of States for duration 2015-16" , fontweight = 'bold')
plt.ylabel("Total GDP (in cr)",fontweight = 'bold')
plt.xlabel("States",fontweight = 'bold')
```

```
df_1a_originalx7_total_gdp.shape

### Insights from the above Plot considering the GSDP of various states for duration 2015-16
*  GSDP of bigger states like TN and UP is higher as compared to smaller states like Sikkim and Arunachal Pradesh.*
*  GSDP of southern states like TN,Karnataka, Kerlala are better as compared to rest of the India.*
*  GSDP of most populous state Uttar Pradesh stands at position 2.
*  India's Silicon Valley Bangalore assisting Karnataka secure position 3.


# GSDP of Top 5 States
df_1a_originalx7_total_gdp.tail(5).plot(kind='bar',stacked=True, colormap = 'Dark2')
plt.title("Total GDP of top 5 States for 2015-16", fontweight = 'bold')
plt.ylabel("Total GDP (in cr)",fontweight = 'bold')
plt.xlabel("States",fontweight = 'bold')


# GSDP of Bottom 5 States
df_1a_originalx7_total_gdp.head(5).plot(kind='bar',stacked=True, colormap = 'Set1')
plt.title("Total GDP of bottom 5 States for 2015-16", fontweight = 'bold')
plt.ylabel("Total GDP (in cr)",fontweight = 'bold')
plt.xlabel("States",fontweight = 'bold')

# Reading all the csv files using glob functionality from a directory for further analysis


path = r'C:\Users\yaswa\Downloads\N'  # Define path properly
files = glob.glob(path + "/*.csv")  # Ensure files are selected

data = pd.DataFrame()

for f in files:
   dfs = pd.read_csv(f, encoding='unicode_escape')
   dfs['State'] = f.replace(path, '').replace('NAD-', '').replace('-GSVA_cur_2016-17.csv', '') \
                 .replace('-GSVA_cur_2015-16.csv', '').replace('-GSVA_cur_2014-15.csv', '').replace('_', ' ')
   data = pd.concat([data, dfs], ignore_index=True)  # Use concat instead of append

data = data.iloc[:, ::-1]  # Reverse columns if needed
data.sort_values(by="State", inplace=True)  # Sort properly if required
```

```
# Selecting the required columns for the Analysis

df_1a_original = data[['State', 'Item', '2014-15']]
df_1a_original1 = df_1a_original.reset_index(drop = True)

# Cleansing the columns name

df_1a_original1['Item'] = df_1a_original1['Item'].map(lambda x: x.rstrip('*') if
isinstance(x, str) else x)
df_1a_original1 = df_1a_original1.set_index('State')

# Pivoting the df for enhanced analysis of data

df_1a_original2 = pd.pivot_table(df_1a_original1, values = '2014-15', index=['Item'],
columns = 'State').reset_index()
df_1a_original3 = df_1a_original2.set_index('Item',drop=True)
#df3

# Dropping the UT as it will not be used in further analysis

df_1a_original4 = df_1a_original3.drop(['Andaman Nicobar Islands', 'Chandigarh',
'Delhi', 'Puducherry'], axis=1, errors='ignore')

df_1a_original5_percapita = df_1a_original4.loc['Per Capita GSDP
(Rs.)'].sort_values()

#Plot for GDP per capita in Rs. for all states

df_1a_original5_percapita.plot(kind='barh',stacked=True, colormap = 'gist_rainbow')
plt.title("GDP per Capita for All States for duration 2014-15", fontweight = 'bold')
plt.xlabel("GDP per Capita (in Rs.)",fontweight = 'bold')
plt.ylabel("States", fontsize = 12, fontweight = 'bold')



#Plot for GDP per Capita of top 5 States for 2014-15

df_1a_original5_percapita.tail(5).plot(kind='bar',stacked=True, colormap = 'winter')
plt.title("GDP per Capita of top 5 States for 2014-15", fontweight = 'bold')
plt.ylabel("GDP per Capita (in Rs.)", fontweight = 'bold')
plt.xlabel("States", fontsize = 12, fontweight = 'bold')
```

```
#Plot for GDP per Capita of bottom 5 States for 2014-15

df_1a_original5_percapita.head(5).plot(kind='bar',stacked=True, colormap = 'Set1')
plt.title("GDP per Capita of bottom 5 States for 2014-15", fontweight = 'bold')
plt.ylabel("GDP per Capita (in Rs.)", fontweight = 'bold')
plt.xlabel("States", fontweight = 'bold')


def safe_get(series, index_name):
   return series.loc[index_name] if index_name in series.index else None

Goa_percapita = safe_get(df_1a_original5_percapita, 'Goa') / df_1a_original5_percapita.sum() * 100 if 'Goa' in df_1a_original5_percapita.index else None
Sikkim_percapita = safe_get(df_1a_original5_percapita, 'Sikkim') / df_1a_original5_percapita.sum() * 100 if 'Sikkim' in df_1a_original5_percapita.index else None
Bihar_percapita = safe_get(df_1a_original5_percapita, 'Bihar') / df_1a_original5_percapita.sum() * 100 if 'Bihar' in df_1a_original5_percapita.index else None
UP_percapita = safe_get(df_1a_original5_percapita, 'Uttar Pradesh') / df_1a_original5_percapita.sum() * 100 if 'Uttar Pradesh' in df_1a_original5_percapita.index else None

# Ratio of the highest per capita GDP to the lowest per capita GDP

h_percapita = df_1a_original5_percapita.iloc[-1]
l_percapita = df_1a_original5_percapita.iloc[0]
percapita_ratio = (h_percapita/l_percapita).round(3)

percapita_ratio

# Selecting Primary Secondary and Tertiary sector for percentage contribution in total GDP

df_1a_original_gdp_con = df_1a_original4.loc[['Primary', 'Secondary', 'Tertiary','Gross State Domestic Product']]
df_1a_original_gdp_percon = (df_1a_original_gdp_con.div(df_1a_original_gdp_con.loc['Gross State Domestic Product'])*100).round(2)
df_1a_original_gdp_percon =df_1a_original_gdp_percon.T.iloc[:,:3]

# Plot for % contribution of sectors in total GDP
```

```
df_1a_original_gdp_percon.plot(kind='bar',stacked=True, colormap = 'prism')
plt.title("% Contribution of Primary, Secondary, Tertiary sector in total GDP for 2014-15",fontweight = 'bold')
plt.ylabel("% Contribution", fontweight = 'bold')
plt.xlabel("States", fontweight = 'bold')

# Sorting the df for better visualization

df_1a_original_sort = df_1a_original4.T.sort_values(by = 'Per Capita GSDP (Rs.)', ascending = False)
df_1a_original_sort

# Define the quantile values and bins for categorisation

df_1a_original_sort.quantile([0.2,0.5,0.85,1], axis = 0)
bins = [0, 67385, 101332, 153064.85, 271793]
labels = ["C4", "C3", "C2", "C1"]
df_1a_original_sort['Category'] = pd.cut(df_1a_original_sort['Per Capita GSDP (Rs.)'], bins = bins, labels = labels)
df_1a_original_index = df_1a_original_sort.set_index('Category')
df_1a_original_sum = df_1a_original_index.groupby(['Category']).sum()
df_1a_original_rename = df_1a_original_sum.rename(columns = {"Population ('00)" : "Population (00)"})

# Selecting the sub sectors which will be used for further analysis

df_1a_original7_sector = df_1a_original_rename[['Agriculture, forestry and fishing','Mining and quarrying','Manufacturing','Electricity, gas, water supply & other utility services',
            'Construction','Trade, repair, hotels and restaurants','Transport, storage, communication & services related to broadcasting','Financial services',
           'Real estate, ownership of dwelling & professional services','Public administration','Other services','Gross State Domestic Product']]

# Calculating and rounding the percentage contribution of each subsector in total GSDP

df_1a_original8_per = (df_1a_original7_sector.T.div(df_1a_original7_sector.T.loc['Gross State Domestic Product'])*100)
df_1a_original8_round = df_1a_original8_per.round(2)
df_1a_original9_per = df_1a_original8_round.drop('Gross State Domestic Product')
df_1a_original9_per
```

```
# Plot for % Contribution of subsectors in Total GDP for C1 states for 2014-15

df_1a_original9_per['C1'].sort_values().plot(kind='bar',stacked=True, colormap =
'Accent')
plt.title("% Contribution of subsectors in Total GDP for C1 states for 2014-15",
fontweight = 'bold')
plt.xlabel("Sub-sectors", fontweight = 'bold')
plt.ylabel("% Contribution", fontweight = 'bold')

# Plot for % Contribution of subsectors in Total GDP for C2 states for 2014-15

df_1a_original9_per['C2'].sort_values().plot(kind='bar',stacked=True, colormap =
'Accent')
plt.title("% Contribution of subsectors in Total GDP for C2 states for 2014-15",
fontweight = 'bold')
plt.ylabel("% Contribution", fontweight = 'bold')
plt.xlabel("Sub-sectors", fontweight = 'bold')

# Plot for % Contribution of subsectors in Total GDP for C3 states for 2014-15

df_1a_original9_per['C3'].sort_values().plot(kind='bar',stacked=True, colormap =
'Accent')
plt.title("% Contribution of subsectors in Total GDP for C3 states for 2014-15",
fontweight = 'bold')
plt.ylabel("% Contribution", fontweight = 'bold')
plt.xlabel("Sub-sectors", fontweight = 'bold')


# Plot for % Contribution of subsectors in Total GDP for C4 states for 2014-15

df_1a_original9_per['C4'].sort_values().plot(kind='bar',stacked=True, colormap =
'Accent')
plt.title("% Contribution of subsectors in Total GDP for C4 states for 2014-15",
fontweight = 'bold')
plt.ylabel("% Contribution", fontweight = 'bold')
plt.xlabel("Sub-sectors", fontweight = 'bold')

### Plot for top 3/4/5/6 sub-sectors that contribute to approximately 80% of the GSDP
of each category.

# 80% Contribution by top subsectors in Total GSDP for C1/C2/C3/C4 States 2014-
15

fig, axes = plt.subplots(2,2, figsize=(15,12))
```

```
fig.tight_layout()
plt.subplots_adjust(left=None, bottom=None, right=None, top=None, wspace=0.8,
hspace=1.8)


df_1a_original9 = df_1a_original9_per.sort_values(by = ['C1', 'C2', 'C3', 'C4'],
ascending = False)
topsubsector = df_1a_original9[df_1a_original9.C1.cumsum() <= 80]
top_c1 = topsubsector[['C1']]
top_c1.plot(kind='bar',stacked=True, colormap = 'Dark2',ax=axes[0][0])


df_1a_original9 = df_1a_original9_per.sort_values(by = ['C2', 'C3', 'C4','C1'],
ascending = False)
topsubsector = df_1a_original9[df_1a_original9.C2.cumsum() <= 80]
top_c2 = topsubsector[['C2']]
top_c2.plot(kind='bar',stacked=True, colormap = 'Dark2',ax=axes[0][1])


df_1a_original9 = df_1a_original9_per.sort_values(by = ['C3', 'C4','C1','C2'],
ascending = False)
topsubsector = df_1a_original9[df_1a_original9.C3.cumsum() <= 80]
top_c3 = topsubsector[['C3']]
top_c3.plot(kind='bar',stacked=True, colormap = 'prism',ax=axes[1][0])


df9 = df_1a_original9_per.sort_values(by = ['C4','C1','C2', 'C3'], ascending = False)
topsubsector = df_1a_original9[df_1a_original9.C4.cumsum() <= 80]
top_c4 = topsubsector[['C4']]
top_c4.plot(kind='bar',stacked=True, colormap = 'prism',ax=axes[1][1])


### GDP and Education Dropout Rates Relationship

# Reading the relevant file on which Analysis needs to be done

file1 = r"C:\Users\yaswa\Downloads\Dropout rate dataset.csv"
df_1a_original_dropout = pd.read_csv(file1)

# Renaming the columns which are incorrect

df_1a_original_rename = df_1a_original_dropout.rename(columns = {'Primary -
2014-2015' : 'Primary - 2013-2014','Primary - 2014-2015.1' : 'Primary - 2014-2015'})
```

```
# Selecting the columns which will be used for further analysis

df_1a_originala = df_1a_original_rename[['Level of Education - State','Primary - 2014-2015','Upper Primary - 2014-2015','Secondary - 2014-2015']]

# Dropping the union territory because it will not be used in further analysis

df_1a_originala1 = df_1a_originala.drop([0, 5, 7, 8, 9, 18, 26, 35, 36])
df_1a_originala2 = df_1a_originala1.reset_index(drop=True)

# Calculating the Missing Values % contribution in DF

df_1a_originala2.isna().mean().round(2) * 100

# Selecting the required column for further analysis

df_1a_originala3 = df_1a_original4.T.reset_index()
df_1a_originala4 = df_1a_originala3[['State', 'Per Capita GSDP (Rs.)']]

# Concatenating the Education dropout df and Per Capita of States df

df_1a_originala5 = pd.concat([df_1a_originala2, df_1a_originala4], axis = 1)
df_1a_originala6 = df_1a_originala5.drop(['State'], axis = 1)
df_1a_originala7 = df_1a_originala6.set_index('Level of Education - State', drop = True)

# Scatter Plot for GDP per capita with dropout rates in education

f = plt.figure()
f, axes = plt.subplots(nrows = 2, ncols = 2, sharex=True, sharey = False)
plt.subplots_adjust(left=None, bottom=None, right=None, top=None, wspace=0.5, hspace=None)

sc = axes[0][0].scatter(df_1a_originala7['Primary - 2014-2015'],df_1a_originala7['Per Capita GSDP (Rs.)'], s=100, c='DarkRed',marker="o")
axes[0][0].set_ylabel('Per Capita GSDP (Rs.)')
axes[0][0].set_xlabel('Primary Education')

sc = axes[0][1].scatter(df_1a_originala7['Upper Primary - 2014-2015'],df_1a_originala7['Per Capita GSDP (Rs.)'], s=100, c='DarkBlue',marker="*")
axes[0][1].set_ylabel('Per Capita GSDP (Rs.)')
axes[0][1].set_xlabel('Upper Primary Education')

sc = axes[1][0].scatter(df_1a_originala7['Secondary - 2014-
2015'],df_1a_originala7['Per Capita GSDP (Rs.)'], s=100, c='DarkGreen',marker="s")
axes[1][0].set_ylabel('Per Capita GSDP (Rs.)')
axes[1][0].set_xlabel('Secondary Education')
```

```
df_1a_originala7.plot(kind='scatter',x='Primary - 2014-2015',y='Per Capita GSDP (Rs.)', s=150, c='DarkRed',marker="o")
```

```
df_1a_originala7.plot(kind='scatter',x='Upper Primary - 2014-2015',y='Per Capita GSDP (Rs.)', s=150, c='DarkRed',marker="*")
```

```
df_1a_originala7.plot(kind='scatter',x='Secondary - 2014-2015',y='Per Capita GSDP (Rs.)', s=150, c='DarkRed',marker="s")
```

REFERENCES

[1] Martin Schneider, Martin Spitzer, "Forecasting Austrian GDP using the generalized dynamic factor model," 17 September 2004.

[2] Gary L. Shelley, Frederick H. Wallace, "Inflation, money, and real GDP in Mexico: a causality analysis," Applied Economics Letters, vol. 11, no. 4, p. 223–225, 2004.

[3] Long Gang, "GDP Prediction by Support Vector Machine Trained with Genetic Algorithm," in 2nd International Conference on Signal Processing Systems (ICSPS), 2010.

[4] Anwar Ali Shah G.Syed, Faiz Muhammad Shaikh, "Effects of Macroeconomic Variables on Gross Domestic Product (GDP) in Pakistan," in International Conference on Applied Economics (ICOAE), 2013.

[5] Carlos Encinas-Ferrer, Eddie Villegas-Zermeño, "Foreign direct investment and gross domestic product growth," in International Conference on Applied Economics, ICOAE , Kazan, Russia, 2-4 July 2015.

[6] Gourav Kalbalia, Vivek Tambi, "Forecasting GDP: A Linear Regression Model," DU Journal of Undergraduate Research and Innovation, vol. 2, no. 2, pp. 41-46, 2016.

[7] John Roush, Keith Siopes, Gongzhu Hu, "Predicting Gross Domestic Product Using Autoregressive Models," in IEEE SERA, London, UK, June 7-9, 2017.

[8] Maciej Woźniak, Joanna Duda, Aleksandra Gąsior,Tomasz Bernat, "Relations of GDP growth and development of SMEs in Poland," in 23rd International Conference on Knowledge-Based and Intelligent Information & Engineering Systems, 2019.

[9] Luca Coscieme, Lars F. Mortensen , Sharolyn Anderson, James Ward , Ian Donohue,Paul C. Sutton, "Going beyond Gross Domestic Product as an indicator to bring coherence to the Sustainable Development Goals," Journal of Cleaner Production, vol. 248, 2019.

[10] Jaehyun Yoon, "Forecasting of Real GDP Growth Using Machine Learning Models: Gradient Boosting and Random Forest Approach," Springer Science+Business Media, LLC, part of Springer Nature, 2020.