

“AUTOMATION ROOT CAUSE ANALYSIS IN TELECOM”

*Project Report submitted in partial fulfillment of the requirements for
the Degree of*

BACHELOR OF TECHNOLOGY

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

By

SAMYAK JAIN

Enrollment No: 161095

UNDER THE GUIDANCE

OF

Neeraj Bhatnagar

(Senior Manager – Operations Support)

Dinesh Kumar

(Program Manager)



JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

TABLE OF CONTENTS

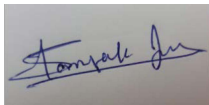
CAPTION NO.	PAGE
DECLARATION	i
ACKNOWLEDGEMENT	ii
LIST OF FIGURES	iii
ABSTRACT	iv
CHAPTER-1: INTRODUCTION	1
1.1 General Introduction	1
1.2 Organization	1
1.3 Motivation	2
1.4 Objectives	2
1.5 Target Specification	3
CHAPTER-2: BACKGROUND4	
2.1 Introduction	4
2.2 Unlock the value of tele data analytics	5
2.3 Get big data insights out of the lab and into everyday operations	5
2.4Get big data, perspectives out of the laboratory and into the routine operations	7
CHAPTER-3: Data Science With Python	9
3.1 Pyhton	9
3.2Why Python?	9
3.3 Essential Python Libraries	10
3.3.1 NumPy	10
3.3.2 Pandas	11
3.3.3 Matplotlib	12
3.3.4 SciPy	12
CHAPTER-4: Numpy basics	14
4.1 Num-py	14
4.2 The NumPy nd-array: A Multidimensional Array Object	14

4.3 Creating nd-arrays	15
4.4 Data Types for n-darrays	16
4.5 Basic Indexing and Slicing	17
4.6 Files Input Output with Array	18
4.7 Storings Array in Binary Formats	18
4.8 Savings or Loadings Text docs	18
CHAPTER-5: Getting Started with pandas	20
5.1 Pandas	20
5.2Introduction to pandas Data Structures	21
5.3 Series	22
5.4DataFrame	23
5.5Reading and Writing Data in Text Format	25
5.6 Arithmetic and data alignment	26
CHAPTER-6:Data analysis in Telecom	30
6.1 Analyzing Mobile Data speed from TRAI with Pandas	30
6.2 About Dataset	30
6.2.1Plotting the Data	35
6.2.2 Comparing Data of two month	44
CHAPTER-7:Conclusion and Future Work	47
7.1 Brief summary of work	47
7.2 Conclusion	47
7.3 Future Scope of Work	48
REFERENCES	49

DECLARATION

I hereby declare that the work reported in this report entitled “AUTOMATION ROOT CAUSE ANALYSIS IN TELECOM” in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Electronics and Communication Engineering** submitted in the department of Electronics & Communication Engineering, Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from Feb 2020 to May 2020 under the supervision of **Dinesh Kumar**(Program Manager) and **Neeraj Bhatnagar** (Senior Manager Operation Support, Ericsson).

The matter embodied in the report has not been submitted for the award of any other degree or diploma.



Samyak Jain

This is to certify that the above statement made by the candidate is true to the best of my knowledge.



ERICSSON

Neeraj Bhatnagar

Senior Manager

SDU Bharti-BMAS SA MNS SLOP SDU BH Tools & Auto

Mobile:+917838290356

Neeraj.n.bhatnagar@ericsson.com

Ericsson

Knowledge Boulevard

201309,Noida,Uttar Pradesh

India

Ericsson.com

Acknowledgement

I would like to express our special thanks of gratitude to my mentor **Mr. Dinesh Kumar** who gave us the golden opportunity to do this project on the topic **AUTOMATION ROOT CAUSE ANALYSIS IN TELECOM**, which also helped us in doing a lot of Research and we came to know about so many new things.

I am really thankful to him.

Secondly, we would also like to thank Lab assistants who helped us a lot in finalizing this project within the limited time frame.

SAMYAK JAIN
(161095)

LIST OF FIGURES

Figure	Page
2.1: Automated Insights Operations	5
2.2: Operator Innovations through analytics	6
2.3: CSP customers different providers	7
3.1: Pandas frame	11
3.2: Matplotlib	12
3.3: Scipy optimization	13
4.1: Multidimensional Array Object	15
4.2: ndarray Object	16
4.3: NumPy data types	17
4.4: Indexing elements in a NumPy array	17
4.5: Saving and Loading Text Files	19
5.1: Example of Series in pandas	21
5.2: Example of NaN in pandas	22
5.3: Example of DataFrame	23
5.4: Possible data inputs to DataFrame constructor	24
5.5: Reading and Writing Data in Text Format	25
5.6: Airthmetic and data alignment	26
6.1 Dataset of TARA I	28
6.2 Unique States Output	29
6.3 output	29

6.4 comparing output	36
6.5 Speed data of jio	38
6.6 Comparing data of jio	38
6.7 comparison of last two months	39
6.8 Unique States Output	41
6.9 Bar plot of the output	42

ABSTRACT

My five month internship project, AUTOMATION ROUTH CAUSE ANALYSIS IN TELECOM, was with Ericsson, Noida Office starting from the February 2020. The report is prepared to cover the basic details of the problems and the solutions employed. project's main objective was to automate telecommunications networks analyze a way of providing referred automation for root cause analysis in networks The client's key challenge was monitoring the data output, taking into consideration all the challenges that data representatives are facing, designing methods for sorting data through their opportunities and compensations. The approach is better described in the next pages, but the reward program usually relies on the company they advertise the region.

This main goal of project was automation in telecom networks and the analysis, to figure out a way to provide preferable automation for the root cause analysis in networks. The main aim of the project was to find a way to handle ok for at of the daily outcome of the OSs dump and manage it accordingly so that it can be further died in the correct format The major problem faced in this project is the large amount of data that came every hour of the day and developing methodology to automation root cause analysis in networks. The method is elaborated details properly in upcoming chapters but in general, before going through the analysis part we should know about the networks their components and in which way they work.

CHAPTER 1

INTRODUCTION

1.1 General Introduction

Data analysis is defined as a method for processing, converting and analyzing data to find useful business decision-making

insights Data Analysis seeks to gather important data details and render the judgment based on the data analysis.

In our daily lives, whenever we make any choice, it is by making that particular decision that we think about what happened last time, or what will happen. It is nothing but an understanding of our past or potential and dependent decisions. Of that we gather images of our lives, or dreams of our future. So this is a mere interpretation of the results.

If we make some choice in our personal lives, it is by making the specific decision the we worry of what occurred last time, or what will happen. is nothing but an interpretation of our history or future and choices that are centered on it. We collect memories of our past or dreams of our future for that. So that is nothing but data analysis.

1.2 Organization

Ericsson Company, with headquarters in Stockholm, Sweden, is a leading international telecommunication and networking equipment company. It was established by the Lars Magnus Ericsson in 1876. The business is a pioneer in providing ICT solutions worldwide. This offers the mobility, internet, and other cloud computing services. Ericsson is an organization with over 39,000 patents. It has a strong grip on mobile communications technology patents. Their portfolio of patents covers 2 G, 3 G and 4 G innovations. More

than 40 percent of telephone calls are made through their networks, and more than 2 billion people worldwide use their network. The corporation has clients in more than 180 countries offering infrastructure such as online computing and internet. Ericsson offers end to end software for all telecom networking markets and has four main lines of industry. Such categories are modems for the Business Unit Networks, Corporate Operations Business Unit, Support Systems Business Unit and the Business Unit.

1.3 Motivation

Based on the fundamental vision of our creator to make contact accessible to everyone, we became the guiding force behind some of the most effective inventions known to mankind. Our sector is one of the few that reaches that all, anywhere on a regular basis, and by 2020 global broadband networks would reach 90 per cent of the world population. We have over 51,000 patents awarded which have changed lives, businesses and culture. network would reach 90 per cent of the case will. The sector is one of the few that affects nearly everyone, anywhere on a regular basis, and by 2020, national broadband world's population. We have over 51,000 awarded patents that have changed lives, businesses and the whole of society. Along with perspectives into how they can address customer challenges and real life is inventions enable positive societal impact. We empower an economic system, our of players to evolve on a collective basis by designing and providing new connection technologies which are simple to implement, use and scale. Such technologies catalyze new opportunities for citizens and enterprises to interact and succeed. We inspire our customers to bind citizens and reinvent economies, solve some of our times most urgent issues, such as climate change, and build a more prosperous environment in this way.

1.4 Objective

The project's key objective was to find a way to tackle the regular result of the dumping of OSs and treat it correctly so that it would perish in the right fashion anymore. Network architecture involves This main project was automation focus region in telecom networks and study to find a way to automate the root cause research desired in networks. The designs for how the network functions from the front office to the back office where the whole deployment and review component was planned and tested with the aid of the respective operating units Leads of other vendors.

1.5 Target Specification

IC study is used to assess the success of a commodity when opposed to the competition Identify the rise or drop in revenue from previous month or quarter revenues Often study is also used to define the targets of the sales manager for the month to come Sales records at the District level is revenues at the provincial level We also cover revenues at the city area and nation state Territorial Level revenue records are used to measure territorial revenue reps payouts territorial level sales managers, district sales manager. state sales managers, and national sales manager. IC report is used for product analysis of how a performs in comparison with the market To identify the increase or decrease in sales as compared to sales in the previous month quarter. Also IC report is also used to identify the sales representative's goals for the coming month

Chapter 2

BACKGROUND THEORY

2.1 Introduction

It's all about user engagement with 5 G and it's enablement. Building on consumer standards from day one is key. An rise in traffic in the network is also anticipated for 5G, as the current approach to traffic processing becomes unsustainable. Software probes are built in with Ericsson's dual mode 5 G Core, securing the output and ensuring optimum performance.

Our Preintegrated Ericsson Expert Analytics derives customercentered information from this content and actionable info, along with SWprobes and 5 G heart. This will preserve the 5G experience. Ericsson Expert Analytics correlates data across network domains and allows for the capture in real time of relevant experience insights into each user session. But how do we work with telco data analytics, and how do Automated Insights Operations capture that data to leverage customer experience.

Take a peek at our new Expert Analytics platform to find out how we utilize Machine Learning frameworks to check for and fix network irregularities in order to maintain the consumer experience optimum

2.2 The value of telecom data analytics

Using this multivendor, crossdomain big data analytics application designed for telecom t o take fast action from realtime consumer interaction and behaviour analytics. Gain into consumer interaction and actions that can improve decision making around the company.

It's all about user engagement with 5 G and its enablement. Building on consumer standards from day one is key. An rise in traffic in the network is also anticipated for 5G, as the current approach to traffic processing becomes unsustainable. Software probes are built in with Ericsson's dual mode 5 G Core, securing the output and ensuring optimum performance.



Figure 2.1: Automated Insights Operations

2.3 Get big data, perspectives out of the laboratory and into the routine operations

Ericsson Expert Analytics is an important data mining toolkit which helps operators collect data Not only data but also particular customer insights to help drive growth driving services. This approach provides telecom provider and observations into behavior. Ericsson Expert Analytics is an important data mining toolkit which helps operators collect data Not only data but also particular customer insights to help drive growth driving services. This approach provides telecom provider and observations into behavior.

Use Cases -

Using Cases. Let the produced use cases break down data silos and leverage cross-domain, end-to - end data sources from network nodes or systems of any vendor.

A strong SDK and APIs provide a versatile framework for promoting and using modern, custom applications

Realtime connection similar to network-

Get actionable perspectives through our real time, close network analysis, together with our special, validated and proprietary algorithms, data models and market rules regarding consumer problems, root causes and next appropriate steps.

Business Standard Patented Index (SLI)-

Test subjective consumer loyalty, and forecast increasing customer's Net Promotor Score (NPS).

Future traffic analysis-

Get proactives assurance for encrypt traffics that will be other-wise be difficult to understands.

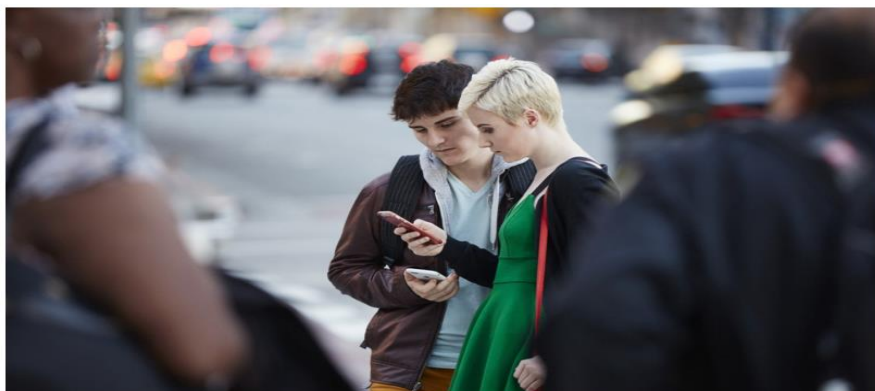


Figure 2.2: Operator Innovations through analytics

2.4 Company processes to maximize interaction

For quicker success in their consumer service initiatives, CSPs will adopt DevOps and agile delivery principles as shown by the DNA – Ericsson partnership. The dedication of DNA and Ericsson centered on pushing continuous improvement through Ericsson 's Technology Growth and Modernisation ADM.

The ADM arrangement involves delivering new updates of the solution within shorter times, in this case every nine weeks, and thereby laid the groundwork for the CEM solution to be continually evolving. ADM embraces the DevOps concepts of agile growth, rapid delivery, meaning that the CSP constantly assesses how an operational system meets consumer experience that the CSP assesses on a regular basis how an existing solution satisfies customer experience demands. Through embracing this agile method of growth, DNA and Ericsson have ensured that the DNA CEM approach remains important to achieving the operator's goal of delivering knowledge to customer care employees to improve consumer interaction

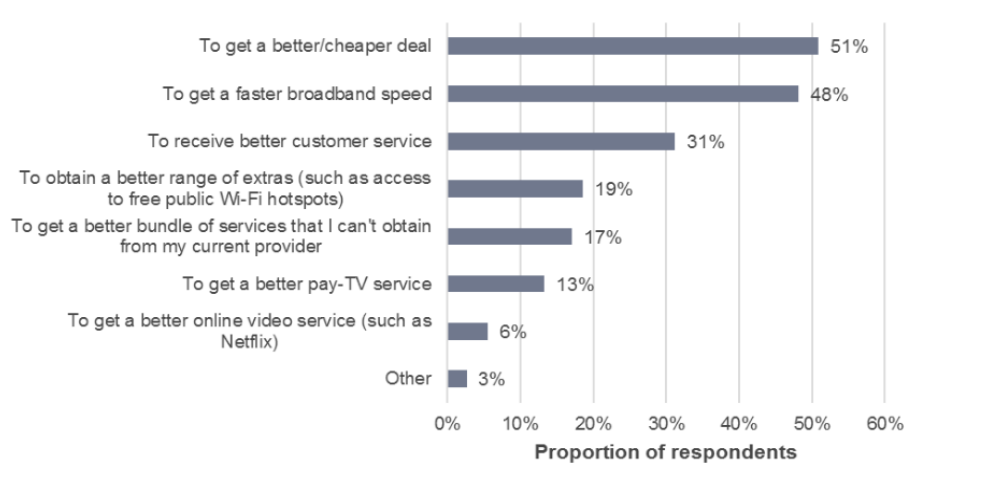


Figure 2.3: Company processes to maximize interaction

Employee retention contributions of the customer service plan of a CSP can not be overemphasised. Employees' sense of satisfaction from their work would be apparent in the way they respond to employers, and CSPs ought to consider their employees ' needs. CSPs may lead to workplace security by engaging workers involved in choices regarding the work that they perform. Consult with the staff before engaging in resources to define the main specifications of the product that is to be implemented. Making use of their experience of market processes to maximize value for the customers. Frequent meetings with delivery teams and product creation teams can help speed up the production of solutions

R&D departments for product lines that are part of the deployment teams need to keep a record of all problems that arise when deploying solutions to customer environments. These documents will feed into the plan for the solution being built as well as other specifications coming from customers.

Chapter 3

DATA SCIENCE WITH PYTHON

3.1 Python

Python talk is easy to keep up with. During its first appearance in 1991, Python has become one of the most popular concise, programming languages along with Perl, Ruby and others. Python and Ruby have become increasingly important in recent years to design websites with their rich web frameworks, such as Rails (Rack) and Django (Python). Such languages are often called scripting languages, when they're used to write quick-and-dirty small programs or scripts. I don't like the appearance "scripting language" because it's got a connotation they can't use to create mission-critical software.

3.2 Why Python?

The simplicity of incorporation of C, C++, and FORTRAN code is part of Python's popularity as a science programming framework. Many computing environments have or have similar collections of legacy FORTRAN and C libraries for linear algebraic efficiency, optimizations, integrated, quick Fourier transforms, and other algorithms like this.

This copy tale happened to other businesses and national laboratories who used Python to tie old applications worth 30 years together. Collectively, 30 years in obsolete software. Most systems consist of tiny pieces of code where much of the time is invested, with vast quantities of "glue code" not operating regularly.

The execution period of the glue code is, in many instances, insignificant; work is more fruitfully spent in minimizing the procedural bottlenecks, often by transferring the application to a language of low levels like C. These languages are also referred to as scripting languages, because they can be used to create small programs quick-and-dirty.

3.3 Python Libraries Key

I provide the following summary of growing repository for those who are less acquainted with the theoretical Python environment and the modules being used in the novel.

3.3.1 NumPy

NumPy is the basic program for mathematical programming in Python, short for Numerical Python. Most of this book will be Num-Py focused or library designed on top of Num-Py.

It includes, inter alia:

- A simple and effective nd-array of multidimensional artifacts
- Element-wise computation functions for arrays or logical operations within arrays;
- Software to read and write data sets to disk collection
- Linear algebra operation, Fourier's transformation the creation of random numbers
- Resources merging C++ , C, and Ruby programming into Python In addition to the fast array processing features that NumPy brings to Python, one of its main data analyze functions is as the main repository for data to be transferred in middle algorithm.

NumPy arrays are a lot more powerful means of processing and managing data with numerical data than other built-in data structures in Python. Likewise, libraries written in a lower-level language like C or Fortran can work on the data stored in a NumPy array without copying any data.

3.3.2 Panda

Panda offers rich data structure and functional built for render it simple, articulate to operate with organized data fast. As you'll see, it's one of the key ingredients that allows Python to be an efficient, successful platform for data analysis. Throughout this text, the primary entity throughout pandas is the DataFrame, a two-dimensional tabular, column-oriented data structure for both rows.

```
>>> frame
   total_bill  tip  sex  smoker  day  time  size
1    16.99    1.01 Female    No   Sun  Dinner  2
2    10.34    1.66  Male    No   Sun  Dinner  3
3    21.01    3.5  Male    No   Sun  Dinner  3
4    23.68    3.31  Male    No   Sun  Dinner  2
5    24.59    3.61 Female    No   Sun  Dinner  4
6    25.29    4.71  Male    No   Sun  Dinner  4
7     8.77     2    Male    No   Sun  Dinner  2
8    26.88    3.12  Male    No   Sun  Dinner  4
9    15.04    1.96  Male    No   Sun  Dinner  2
10   14.78    3.23  Male    No   Sun  Dinner  2
```

Figure 3.1: Pandas frame

pandas combines the high performance array-computing features of NumPy with the flexible data manipulation capabilities of spreadsheets and relational databases (such as SQL). It provides sophisticated indexing functionality to make it easy to reshape, slice and

dice, perform aggregations, and select subsets of data. pandas is the primary tool that we will use in this book.

For financial users, pandas features rich, high-performance time series functionality and tools well-suited for working with financial data. In fact, I initially designed pandas as an ideal tool for financial data analysis applications. For users of the R language for statistical computing, the DataFrame name will be familiar, as the object was named after the similar R data.frame object. They are not the same, however; the functionality provided by data.frame in R is essentially a strict subset of that provided by the pandas DataFrame. While this is a book about Python, I will occasionally draw comparisons with R as it is one of the most widely-used open source data analysis environments and will be familiar to many readers.

I can make parallels with R periodically because it is second of the most commonly adopted open source data review platforms and should be common to many users. This name of the pandas itself derives from panelized data, an econometric word for dimensional hierarchical data sets, and Python data.

3.3.3 Matplotlib

I needed to have the option to do these things in a single spot, ideally in a language well suited to broadly useful programming advancement. Python was a decent applicant language for this,

however around then there was not an incorporated arrangement of information structures and apparatuses giving this usefulness. In the course of the most recent four years, pandas has developed into a very enormous library equipped for illuminating an a lot more extensive arrangement of information dealing with issues than I at any point foreseen, however it has extended in its degree without bargaining the straightforwardness and convenience that I wanted from the earliest starting point. I trust that in the wake of perusing this book, you will see it as the same amount of an essential instrument as does. All through remainder of the book, I utilize the accompanying imports shows for panda.

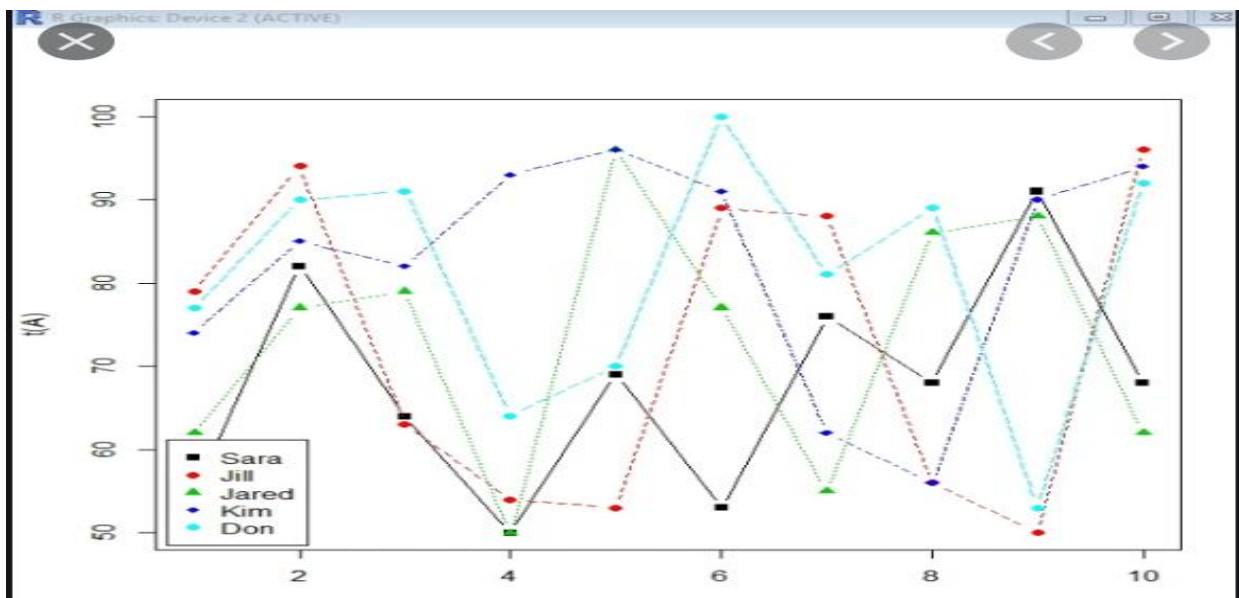


Figure 3.2: Matplotlib

3.3.4 SciPy

SciPy is a series of packages in scientific computing which tackle a number of different standard problem areas. Here's a sampled of the included packaged:

- Scipy-integrate: routines for digital integrated and differentials equation solvers
- scipys.linalg: linear routines of algebra and matrixiez decomposition that extend beyonds these provided in numpy-linalg.

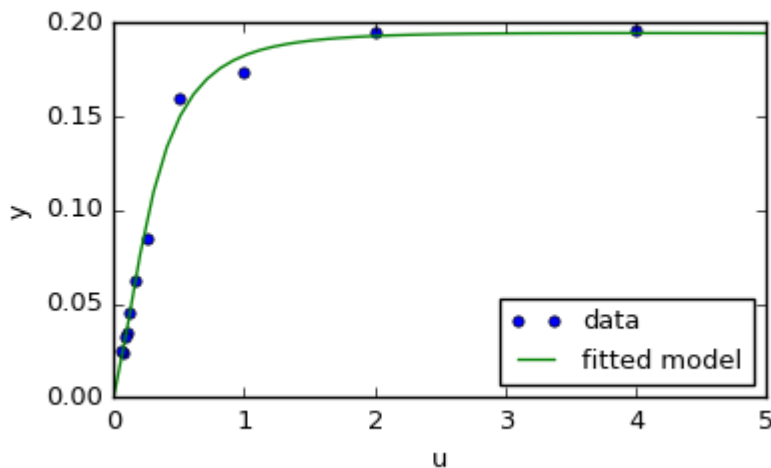


Figure 3.2: Scipy optimization

- Scipy-optimized: root seekingalgorithms feature optimizers (minimizers)
- Scipy-signal: Software for signals processings
- Scipy-sparse: sparse matrix and sparsed vector solver method
- Scipyspecial: SPECFUN wrapped, a Fortrans library that implements several popular mathematical function, such like gammas functions
- scipy-stats: regular continue and discreted distributed of probabily (densitys function,

Chapter 4

NumPy Basics

4.1 NumPy

NumPy, short for Numerical Python, is the main package required for superior logical analysis and information work. It is the framework under which nearly all of the higher-level devices in this volume are mounted. Here's a preview of what it offers you:

- n-darray, a quick and space-efficient multi-dimensional array giving vector number juggling activities modern telecom capacities
- Standards scientific capacities of quick procedure on whole varieties of information without composing circles
- Tools for perusing/composing cluster information to circle and working with memory-mapped documents
- Linear variable based math, arbitrary number age, and Fourier change abilities

Fortran

4.2 The NumPy ndarray: A Multidimensional Array Object

One of the key highlights of NumPy is its N-dimensional array object, or ndarray, which is a quick, adaptable holder for enormous informational collections in Python. Arrays empower you to perform numerical procedure on entire squares of information utilizing comparable linguistic structure to the proportionate tasks between scalar components..

Since NumPy is centered around numerical registering, the information type, if not indicated, will by and large be float64 (gliding point). Since NumPy is centered around numerical figuring, the information type, if not indicated, will by and large be float64 .

```
In [8]: data
Out[8]:
array([[ 0.9526, -0.246 , -0.8856],
       [ 0.5639,  0.2379,  0.9104]])

In [9]: data * 10
Out[9]:
array([[ 9.5256, -2.4601, -8.8565],
       [ 5.6385,  2.3794,  9.104  ]])

In [10]: data + data
Out[10]:
array([[ 1.9051, -0.492 , -1.7713],
       [ 1.1277,  0.4759,  1.8208]])
```

Figure 4.1: Multidimensional Array Object

Array is a nonexclusive multidimensional compartment for homogeneous information; that is, the entirety of the components must be a similar kind. Each exhibit has a figure, a tuple demonstrating the of each measurement, and dtype, an item portraying information sort of the clusters.

4.3 Creating ndarrays

The most straightforward approach to make a cluster is to utilize the exhibit work. This acknowledges any arrangement like item (counting different clusters) and creates another NumPy exhibit containing the passed information. For instance, a rundown is a decent contender for change:

```
In [15]: data1 = [43.5, 8, 1, 5] In [14]: arr1 = np.-array(data1) In [15]:
arr1 Out[17]: array([ 2. , 4.5, 6. , 10. , 12. ])
```

Settled groupings, similar to a rundown of equivalent length records, will be changed over into a multidimensional exhibit:

```
In [17]: data2 = [[11, 12,1 3,1 4], [15,1 6,1 7, 18]] In [19]: arr-2 = np.-array(data2) In
[19]: arr2 Out[14]: array([[4, 7, 8, 9], [3, 2, 1, 0]])
```

```
In [19]: arr2.ndim Out[18]: 2 In [20]: arr2.-shape Out[20]:
```


(3, 4) Unless expressly indicated (more on this later), `np.array` attempts to surmise a decent information of `dtype` for the exhibit that it makes.

This information types put away an extraordinary d-type object; for instance, in the over one models we have: In [22]: `arr1.dtype` Out[23]: `dtype('float64')`

Notwithstanding `np.array`, there are various different capacities for making new clusters. As models, zero and one make varieties of 0s or 1s, individually, with a given lengths or shapes. `void` makes a cluster without introducing its qualities to a specific worth.

4.4 Datas Type for n-darrays

This information types and d-type is an uncommon item contain the datas the n-darray need decipher a piece of memories as a specific sort information

```
In [27]: arr1 = np.array([1, 2, 3], dtype=np.float64)
```

```
In [28]: arr2 = np.array([1, 2, 3], dtype=np.int32)
```

```
In [29]: arr1.dtype          In [30]: arr2.dtype  
Out[29]: dtype('float64')   Out[30]: dtype('int32')
```

Figure 4.2:nd-array Object

Dtypes are a piece of what make NumPy so ground-breaking and adaptable. As a rule they map straightforwardly onto a fundamental machine portrayal, which makes it simple to peruse and compose paired surges of information to circle and furthermore to interface with codes write in a lower-level languages like C++ or Fortren. The numericals d-types are named a similar way: a sort name, similar to `float` or `int`, trailed by a number demonstrating the quantity of bits per component. A standards twofold exactness drifting

points esteem (what utilized in engine in Python's buoy object) takes up 18 bytes or 4 bits.

About retaining the NumPy dtypes, particularly in case you're another client. It's regularly just important to think about the general sort of information you're managing, in the case of coasting point, complex, whole number, boolean, string, or general Python object.

Type	Type Code	Description
int8, uint8	i1, u1	Signed and unsigned 8-bit (1 byte) integer types
int16, uint16	i2, u2	Signed and unsigned 16-bit integer types
int32, uint32	i4, u4	Signed and unsigned 32-bit integer types
int64, uint64	i8, u8	Signed and unsigned 32-bit integer types
float16	f2	Half-precision floating point
float32	f4 or f	Standard single-precision floating point. Compatible with C float
float64, float128	f8 or d	Standard double-precision floating point. Compatible with C double and Python float object

Figure 4.3: NumPy data types

4.5 Basics of Slicing and Indexing

Num-Py cluster ordering is a good subject, as theres are numerous way you might need choose a subject of your information or individuals components. Two-dimensionals clusters arex basic; and a superficial level they acts correspondingly to Python's records.

On the off chance that you are new to NumPy, you may be astonished by this, particularly in the event that they have utilized other exhibit programming dialects which duplicate information all the more energetically. As NumPy has been structured in view of huge information use cases, you could envision execution and memory issues if NumPy demanded replicating information left and right.

		axis 1		
		0	1	2
axis 0	0	0,0	0,1	0,2
	1	1,0	1,1	1,2
	2	2,0	2,1	2,2

Figure 4.4: Indexing element a Num-Py arrays

4.6 File Input and Output with Arrays

NumPy can spare and burden information to and from plate either in content or paired organization. In later parts you will find out about instruments in pandas for adding plain information to memory.

4.7 Storings Array on Disk in Uninary Formats

Np-save and np-load are the two work capacities for effectively sparing and stacking cluster information in plate. Clusters are spared as a matter of course in an uncompressed crude parallel organization with record expansion .n-py.

```
In [183]: arr = np.arange(10)
```

```
In [184]: np.save('some_array', arr)
```

You spare different exhibits in a zip document utilizing np.savez and passing the clusters as catchphrase contentions:

```
In [186]: np.savez('array_archive.npz', a=arr, b=arr)
```

When stacking a .npz record, you get back a dict-like item which stacks the individual exhibits apathetically:

```
In [187]: curve = np.load('array_archive.npz')
```

```
In [188]: arch['b'] Out[188]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

4.8 Loadings and Savings Text Docs

Stacking content for records in a genuinely standards errand. The scene of record perusing are composing capacities in P-python can a somewhat befuddling for a newcomer, so I will concentrate for the most part on the read-csv and read-tables capacities in panda. Its will on occasion be helpful to stack information into vanilla NumPy exhibits utilizing np-loadtxt or the more particular np.-genfromtxt. These capacities has numerous alternatives permitting you to determine distinctive delimitars, convertar works of specific segments, skipping columnss, and different thing. Takes a straightforward instance of a comma-isolated record (csv) .

```
In [191]: !cat array_ex.txt
0.580052,0.186730,1.040717,1.134411
0.194163,-0.636917,-0.938659,0.124094
-0.126410,0.268607,-0.695724,0.047428
-1.484413,0.004176,-0.744203,0.005487
2.302869,0.200131,1.670238,-1.881090
-0.193230,1.047233,0.482803,0.960334
```

This can be loaded into a 2D array like so:

```
In [192]: arr = np.loadtxt('array_ex.txt', delimiter=',')
```

```
In [193]: arr
```

```
Out[193]:
```

```
array([[ 0.5801,  0.1867,  1.0407,  1.1344],
       [ 0.1942, -0.6369, -0.9387,  0.1241],
       [-0.1264,  0.2686, -0.6957,  0.0474],
       [-1.4844,  0.0042, -0.7442,  0.0055],
       [ 2.3029,  0.2001,  1.6702, -1.8811],
       [-0.1932,  1.0472,  0.4828,  0.9603]])
```

Figure 4.5: Saving and Loading Text Files

`np.savetxt` plays out the opposite activity: composing a cluster to a delimited book document. `genfromtxt` is comparable to `loadtxt` however is designed for organized clusters and missing information taking care of; see Chapter 5 for additional on organized exhibits. Stacking content for records in a genuinely standards errand. The scene of record perusing are composing capacities in Python can a somewhat befuddling for a newcomer, so I will concentrate for the most part on the `read_csv` and `read_table` capacities in `panda`. It will on occasion be helpful to stack information into vanilla NumPy exhibits utilizing `np.loadtxt` or the more particular `np.genfromtxt`. These capacities has numerous alternatives permitting you to determine distinctive delimiters, convertar works of specific segments, skipping columns, and different thing. Takes a straightforward instance of a comma-isolated record.

Chapter 5

Gettings Started with panda

5.1 Pandas

Pandas shall be the essential libraries of enthusiasm all through a significant part for the remainder of the data. It contains elevated levels information structured and control apparatuses intended to making information investigation quick and simple in P-ython. panda is based on Num-Py and makes it simple to use in NumPy-driven application. As an touch of foundation, I began buildings panda in mid 2009 during my residency at AQRs, a quantitatives speculation the board firms. At that point, I had an unmistakable arrangement of prerequisites that was not very much tended to by any single instruments avail to me:

- Data structure of marked tomahawk supporting programmes or express information arrangements. This forestalls normal blunders coming about because of skewed information and working with in an unexpected way recorded information originating from various sources.
- Integrated time arrangement usefulness.
- similar information structured handles both times arrangement information and non_time arrangement information.
- Arithmetic tasks and decreases (like adding over a hub) would pass on the metadata (pivot marks).
- Flexibles treatment of missed information.
- Mergesor others social activities founds in well known database databass (SQLbaseds, for instance)

I needed to have the option to do these things in a single spot, ideally in a language well-suited to broadly useful programming advancement. Python was a decent applicant language for this, however around then there was not an incorporated arrangement of information structures and apparatuses giving this usefulness. In the course of the most recent four years, pandas has developed into a very enormous library equipped for illuminating an a lot more extensive arrangement of information dealing with issues than I at any point foreseen, however it has extended in its degree without bargaining the straightforwardness and convenience that I wanted from the earliest starting point. I trust that in the wake of perusing this book, you will see it as the same amount of an essential instrument as does. All through remainder of the book, I utilize the accompanying imports shows for panda.

5.2 Introductions to panda Data Structure

To begin with panda, you should get settled with its two work-horse information structured: Series and Data-Frame. While these are nots an all inclusive answer for each issue, they give a strong, simple to-utilize reason of most application based.

5.3 Series

A Series is a one-dimensional array-like object containing an array of data (of any NumPy data type) and an associated array of data labels, called its index. The simplest Series is formed from only an array of data:

```
In [4]: obj = Series([4, 7, -5, 3])
```

```
In [5]: obj
```

```
Out[5]:
```

```
0    4
```

```
1    7
```

```
2   -5
```

```
3    3
```

Figure 5.1: Example of Series in pandas

I needed to have the option to do these things in a single spot, ideally in a language well suited to broadly useful programming advancement. Python was a decent applicant language for this,

however around then there was not an incorporated arrangement of information structures and apparatuses giving this usefulness. In the course of the most recent four years, pandas has developed into a very enormous library equipped for illuminating an a lot more extensive arrangement of information dealing with issues than I at any point foreseen, however it has extended in its degree without bargaining the straightforwardness and convenience that I wanted from the earliest starting point. I trust that in the wake of perusing this book, you will see it as the same amount of an essential instrument as does. All through remainder of the book, I utilize the accompanying imports shows for panda.

I needed to have the option to do these things in a single spot, ideally in a language well suited to broadly useful programming advancement. Python was a decent applicant language for this,

however around then there was not an incorporated arrangement of information structures and apparatuses giving this usefulness. In the course of the most recent four years, pandas has developed into a very enormous library equipped for illuminating an a lot more extensive arrangement of information dealing with issues than I at any point foreseen, however it has extended in its degree without bargaining the straight forwardness .

I could well walk you through the core concepts of interacting with this section The data contain the data structure or Series. The next chapters will delve deeper Use of pandas to analyze and retrieve data. This book is not meant for the To serve as exhaustive pandas library documentation. I could well walk you through the core concepts of interacting with this section The data contain the data structure or Series. The next chapters will delve deeper Use of pandas to analyze and retrieve data. This book is not meant for the To serve as exhaustive pandas library documentation.

```
In [79]: obj = Series([4.5, 7.2, -5.3, 3.6], index=['d', 'b', 'a', 'c'])  
  
In [80]: obj  
Out[80]:  
d    4.5  
b    7.2  
a   -5.3  
c    3.6
```

Figure 5.2: Example of NaN in pandas

Features of data alignment are dealt with as a separate topic. Both the Series object itself and its index have a name attribute that integrates with other key pandas functionality as as. The interactively displayed series string plaintiff filed the left index and the right values.

5.4 DataFrame

A DataFrame describes a tabular, spread sheet like data structure that includes an organized column array, each of which might be a different form of value (numeric, list, boolean, etc.). The DataFrame contains both a row and an array of rows; it can be called a Sequence dictum (one with those sharing the same table). Compared to certain such frameworks as DataFrame you might have seen before (including R).

The data is stored under the hood as one or more twodimensional blocks, instead of a list, dict, or some other collection of onedimensional arrays. The precise information regarding the internals of DataFrame are way beyond the reach of this text.

The length of the value must match the length of the DataFrame when assigning lists or arrays to a column. Instead, if you assign a Series, it will be exactly conformed to the index of the DataFrame, inserting missing values into any holes. When indexing a DataFrame the column returned is

a view of the underlying data, not a duplicate. All in-place modifications to the Series is also in the DataFrame.

DataFrame contains both a row and an array of rows; it can be called a Sequence dictum (one with those sharing the same table).

Type	Notes
2D ndarray	A matrix of data, passing optional row and column labels
dict of arrays, lists, or tuples	Each sequence becomes a column in the DataFrame. All sequences must be the same length.
NumPy structured/record array	Treated as the "dict of arrays" case
dict of Series	Each value becomes a column. Indexes from each Series are unioned together to form the result's row index if no explicit index is passed.
dict of dicts	Each inner dict becomes a column. Keys are unioned to form the row index as in the "dict of Series" case.
list of dicts or Series	Each item becomes a row in the DataFrame. Union of dict keys or Series indexes become the DataFrame's column labels
List of lists or tuples	Treated as the "2D ndarray" case
Another DataFrame	The DataFrame's indexes are used unless different ones are passed
NumPy MaskedArray	Like the "2D ndarray" case except masked values become NA/missing in the DataFrame result

Figure 5.4: Possible data inputs to DataFrame constructor

I'll walk you through the fundamental mechanics of interacting with the data contained in a Series or DataFrame in this section. Next chapters will delve deeper into the topics of data analysis and manipulation using pandas. This book is not intended to provide the pandas library with an extensive documentation; rather, I concentrate on the most relevant features, (e.g. esoteric) items you are investigating for yourself.

5.5 Reading and Writing Data in Text Format

Because of its straightforward linguistic structure to associate with documents, natural information structures, and helpful highlights such as tuple parsing and unloading, Python has become a cherished language for content and record munging. Pandas includes different capabilities to parse even information as a DataFrame object.

I will give an outline of the mechanics of those capabilities, which are intended to change over content information into a DataFrame. The choices for these capabilities fall into a couple of classifications:

Ordering: can regard at least one segment as the returned DataFrame, and whether to get section names from the document, those clients, and not in the least.

- Type deduction and information transformation: this incorporates the client characterized esteem changes and custom rundown of missing worth markers.
- Date-time parsings: incorporates consolidating ability, including joining datesort times data spreads over various sections into an solitary segment on the outcome.
- Iterating: support for repeating over lumps for huge records.
- Un-clean information issue: skipping columns or a foot, remarks, or other minor things like numeric information with thousands isolated by comma.

The parser capacities have numerous extra contentions to assist you with dealing with the wide assortment of special case document designs that happen (see Table 6-2). For instance, you can skirt the primary, third, and fourth lines of a record

```
In [86]: frame = DataFrame(np.arange(9).reshape((3, 3)), index=['a', 'c', 'd'],
.....:                    columns=['Ohio', 'Texas', 'California'])

In [87]: frame
Out[87]:
   Ohio  Texas  California
a     0     1           2
c     3     4           5
d     6     7           8

In [88]: frame2 = frame.reindex(['a', 'b', 'c', 'd'])

In [89]: frame2
Out[89]:
```

Figure 5.5: Reading and Writing Data in Text Format

Since there was one less section name than the quantity of information lines, `read_table` deduces that the principal segment ought to be the DataFrame's file in this extraordinary case. The parser capacities have numerous extra contentions to assist you with dealing with the wide assortment of special case document designs that happen (see Table 6-2). For instance, you can skirt the primary, third, and fourth lines of a record.

5.6 Arithmetic and information arrangement

One of the most significant pandas highlights is the conduct of number juggling between objects with various lists. While including objects, if any record sets are not the equivalent, the particular list in the outcome will be the association of the list sets. How about we take a gander at a straightforward model. While including objects, if any record sets are not the equivalent, the particular list in the outcome will be the association of the list sets. How about we take a gander at a straightforward model:

```

In [98]: data = DataFrame(np.arange(16).reshape((4, 4)),
.....:                   index=['Ohio', 'Colorado', 'Utah', 'New York'],
.....:                   columns=['one', 'two', 'three', 'four'])

In [99]: data.drop(['Colorado', 'Ohio'])
Out[99]:
   one  two  three  four
Utah   8   9   10   11
New York 12  13   14   15

In [100]: data.drop('two', axis=1)
Out[100]:
   one  three  four
Ohio   0     2    3
Colorado  4     6    7
Utah   8    10   11
New York 12    14   15

In [101]: data.drop(['two', 'four'], axis=1)
Out[101]:
   one  three
Ohio   0     2
Colorado  4     6
Utah   8    10
New York 12    14

```

Figure 5.4: Airthmetic and data alignment

The inside information arrangement presents NA esteems in the records that don't cover.

Missing qualities engender in math calculations.

Chapter 6

Data analysis in Telecom

6.1 Analyzing Mobile Data Speeds from TRAI with Pandas

Python is a brilliant language for data processing, largely due to the wonderful Datacentric Python software ecosystem. Pandas is one of those packages and makes it much easier to import and analyze the data analysis,

Let's use a real dataset from TRAI to analyze mobile data speeds and try to see the average speeds for a particular operator or state in that month. This will also show how easily Pandas could be used on any real world data to derive interesting results.

6.2 About Dataset

Telecom Regulatory Authority of India (TRAI) releases a monthly dataset of the internet speeds it measures through the MySpeed (TRAI) app. This includes speed tests initiated by the user itself or periodic background tests done by the app. We will try to analyze this dataset and see the average speeds for a particular operator or state in that month.

Inspecting the raw structure of data:

- Go to TRAI MySpeed Portal and download the latest month's csv file under the Download section. You can also download the csv file used in this article: sept18_publish.csv or sept18_publish_drive.csv
- Open this spreadsheet file.

NOTE: As the dataset is huge, the software may give you an warning that all rows could not be loaded. This is fine. Also if you are using Microsoft Excel, there might be a warning about opening of a SYLK file. This error could be ignored as it is a

common bug in Excel.

Now, let's take a look at the arrangement of the data-

	A	B	C	D	E	F	G	H
1	JIO	4G	download	46372	-79	Karnataka		
2	JIO	4G	download		-79	Karnataka		
3	JIO	4G	upload		-79	Karnataka		
4	operator	technology	test type	speed measured		signal strength		
5	JIO	4G	download	5955	na	Haryana		
6	JIO	4G	download	4351	na	Haryana		
7	JIO	4G	download	15877	na	Haryana		
8	JIO	4G	download	273	na	Haryana		
9	IDEA	4G	download	11518	na	Haryana		
10	JIO	4G	download	7603	na	Haryana		
11	JIO	4G	upload	68	na	Haryana		
12	JIO	4G	upload	660	na	Haryana	GfG	
13	JIO	4G	upload	1403	na	Haryana		

Figure 6.1:Dataset of TARAI

NOTE: The Signal Strength may have na (Not Available) values due to some devices unable to capture signal. We will ignore using this parameter in our calculations to make things simpler.

However, it could be easily added as a condition while filtering.

Download Speed: Overall



Figure 6.2:Download speed of networks

Essential packs-

Panda-a popular toolkit for analyzing data. Very powerful at crunching large data sets.

Num-py-provides fast and efficient operations on homogeneous data arrays. with pandas and matplotlib, we'll use this.

Matplotlib-is a library for plotting. We'll use its bar plot feature to construct bar graphs

Step #1: Import the packages and define some constants.

```
//importpandas as pd
//importnum-py as np
//importmatplotlib.pyplot as plt
// we will defineS constants
// name of the csv dataset
//DATASET1_FILENAME1 ='sept12_publishS.csv'
//CONST1_OPERATOR1 ='JIO1'
// define the state to be filtered upon.
//CONST1_STATE1 ='Delhi1'
// define the the technology to be filtered upon
//CONST1_TECHNOLOGY1 ='3G'
```

Step #2: Define some lists that will store the final calculated results, so that it could be passed on to the bar plotting function easily. The state (or operator), download speed and upload speed will

be stored serially so that for an index, the state (or operator), it's corresponding download and upload speeds can be accessed.

For Example, `final_states[2]`, `final_download_speeds[2]` and `final_upload_speeds[2]` will give the corresponding values for the 3rd state.

define lists

```
//final1_download1_speeds2 =[]  
//final1_upload1_speeds2 =[]  
//final1_states1 =[]  
//final_operators2 =[]
```

Step #3: Import the file using Pandas `read_csv()` function and store it in 'df'. This will create a DataFrame of the csv contents on which we will work on.

```
//df =pd.read_csv(DATASET1_FILENAME2)  
// assign headers for each of the columns based on the data  
// this allows us to access columns easily  
//df.columns =['Service Provide1r', 'Technology2', 'Test Type3',  
//           'Data Speed'1, 'Signal Strength2', 'State3']
```

Step #4: First lets find all the unique *states* and *operators* in this dataset and store them into corresponding states and operators list.

We will use the `unique()` method of the Pandas dataframe.

```
// find and display the unique states  
//states1 =df1['State1'].unique()  
//print('STATES1 Found1: ', states)
```

```
// find and display the unique operators
//operators1 =df['Service Provider1'].unique()
//print('OPERATORS Found1: ', operators2)
```

Output:

```
STATES Found: ___['Kerala' 'Rajasthan' 'Maharashtra' 'UP East' 'Karnataka'
nan
'Madhya Pradesh' 'Kolkata' 'Bihar' 'Gujarat' 'UP West' 'Orissa'
'Tamil Nadu' 'Delhi' 'Assam' 'Andhra Pradesh' 'Haryana' 'Punjab'
'North East' 'Mumbai' 'Chennai' 'Himachal Pradesh' 'Jammu & Kashmir'
'West Bengal']
OPERATORS Found: ___['IDEA' 'JIO' 'AIRTEL' 'VODAFONE' 'CELLONE']
```

Figure 6.2:Unique States Output

Step #5: Define the function fixed_operator, which will keep the operator constant and iterate through all the available states for that operator. We can construct a similar function for a fixed state.

```
//filtered1 =df[(df['Service Provider1'] ==CONST1_OPERATOR1)
//          & (df['Technology1'] ==CONST1_TECHNOLOGY1)]
// iterate through each of the states
//forstate instates:
// create new dataframe1 which contains
// only the data1 of the current state
//base =filtered[filtered['State'] ==state]
// filter only download speeds1 based on test type
//down =base[base['Test Type'] =='download']
```

```

//filter only upload speeds1 based on test type

//up =base[base['Test Type'] =='upload']

// calculate mean of speeds1 in Data Speed
// column using the Pandas.mean2() method
//avg_down =down['Data Speed2'].mean()

    // calculate mean1 of speeds

//in Data Speed2 column
//avg_up =up['Data Speed2'].mean()

// discard values if mean is not a number(nan)

// and append1 only the valid ones

//if(pd.isnull(avg_down2) orpd.isnull(avg_up)):

    // down, up =0, 0

//else:

    // final_states.append2(state)

    //final_download1_speeds.append(avg_down)

    //final_upload_speeds2.append(avg_up)

    //# print output1 upto 2 decimal places

    // print(str(state) +' -- Avg. Download: '+

        //          str('%.2f'%avg_down) +

        // Avg. Upload: '+str('%.2f'%avg_up))

//else:

    //final_states.2append(state)

```

```

//final_download1_speeds.append(avg_down)

//final_upload_speeds2.append(avg_up)

///# print output upto 2 decimal places

//print(str(state) +' -- Avg. Download:1 '+
        str('%.2f'%avg1_down) +
        //' Avg. Upload1: '+str('%.2f'%avg_up))

```

Step #6: Define the function `fixed_operator`, which will keep the operator constant and iterate through all the available states for that operator. We can construct a similar function for a fixed state. Define some lists that will store the final calculated results, so that it could be passed on to the bar plotting function easily. The state (or operator), download speed and upload speed will be stored serially so that for an index, the state (or operator), it's corresponding download and upload speeds can be accessed.

For Example, `final_states[2]`, `final_download_speeds[2]` and `final_upload_speeds[2]` will give the corresponding values for the *3rd* state.

The state (or operator), download speed and upload speed will be stored serially so that for an index, the state (or operator), it's corresponding download and upload speeds can be accessed.

. The state (or operator), download speed and upload speed will be stored serially so that for an index, the state (or operator), it's corresponding download and upload speeds can be accessed.

```
Kerala -- Avg. Download: 26129.27 Avg. Upload: 5193.46
Rajasthan -- Avg. Download: 27784.86 Avg. Upload: 5736.18
Maharashtra -- Avg. Download: 20707.88 Avg. Upload: 4130.46
UP East -- Avg. Download: 22451.35 Avg. Upload: 5727.95
Karnataka -- Avg. Download: 16950.36 Avg. Upload: 4720.68
Madhya Pradesh -- Avg. Download: 23594.85 Avg. Upload: 4802.89
Kolkata -- Avg. Download: 26747.80 Avg. Upload: 5655.55
Bihar -- Avg. Download: 31730.54 Avg. Upload: 6599.45
Gujarat -- Avg. Download: 16377.43 Avg. Upload: 3642.89
UP West -- Avg. Download: 23720.82 Avg. Upload: 5280.46
Orissa -- Avg. Download: 31502.05 Avg. Upload: 6895.46
Tamil Nadu -- Avg. Download: 16689.28 Avg. Upload: 4107.44
Delhi -- Avg. Download: 20308.30 Avg. Upload: 4877.40
Assam -- Avg. Download: 5653.49 Avg. Upload: 2864.47
Andhra Pradesh -- Avg. Download: 32444.07 Avg. Upload: 5755.95
Haryana -- Avg. Download: 7170.63 Avg. Upload: 2680.02
Punjab -- Avg. Download: 14454.45 Avg. Upload: 4981.15
North East -- Avg. Download: 6702.29 Avg. Upload: 2966.84
Mumbai -- Avg. Download: 14070.97 Avg. Upload: 4118.21
Chennai -- Avg. Download: 20054.47 Avg. Upload: 4602.35
Himachal Pradesh -- Avg. Download: 7436.99 Avg. Upload: 4020.09
Jammu & Kashmir -- Avg. Download: 8759.20 Avg. Upload: 4418.21
West Bengal -- Avg. Download: 16821.17 Avg. Upload: 3628.78
```

Figure 6.3:Output

6.1.1 Plotting the data –

Use the `arange()` method of Numpy which returns evenly spaced values within a given interval.

Here, passing the length of the `final_states` list, hence we get values from 0 to the number of states in the list like `[0, 1, 2, 3 ...]`

We can then use these indices to plot a bar at that location. The second bar is plotted by offsetting the location of the first bar by the bar width.

```

//fig, ax1 =plt.subplots()

//# the width of each bar

//bar_width2 =0.25

//# opacity1 of each bar

//opacity2 =0.8

//# store the positions

//index1 =np.arange(len(final_states))

//# the plt.bar() takes in the position

//# of the bars2, data to be plotted,

//# width of each2 bar and some other

//# optional parameters1, like the opacity and colour

//# plot the download 1bars

//bar_download1 =plt.bar(index, final_download2_speeds,

        // bar_width, alpha2=opacity,

        // color='b', label='Download')

//# plot the upload2 bars

//bar_upload =plt.bar1(index +bar_width, final_upload_speeds,

        // bar_width, alpha1=opacity, color='g',

        // label='Upload')

// # title of the graph

//plt.title('Avg. Download/Upload2 speed for '

// +str(CONST_OPERATOR))

//# the x-axis label 1

//plt.xlabel('States2')

```

```

    ## the y-axis label2
    plt.ylabel2('Average Speeds in Kbps')
    ## the label below each of the bars,
    ## corresponding to the states
    plt.xticks2(index +bar_width, final_states, rotation=90)
    ## draw the legend 2
    plt.legend2()
    ## make the graph layout tight2
    plt.tight_layout2()
    ## show the graph2
    plt.show2()

```

Define some lists that will store the final calculated results, so that it could be passed on to the bar plotting function easily. The state (or operator), download speed and upload speed will be stored serially so that for an index, the state (or operator), it's corresponding download and upload speeds can be accessed.

Some lists that will store the final calculated results, so that it could be passed on to the bar plotting function easily. The state (or operator), download speed and upload speed will be stored serially so that for an index, the state (or operator), it's corresponding download and upload speeds can be detected.

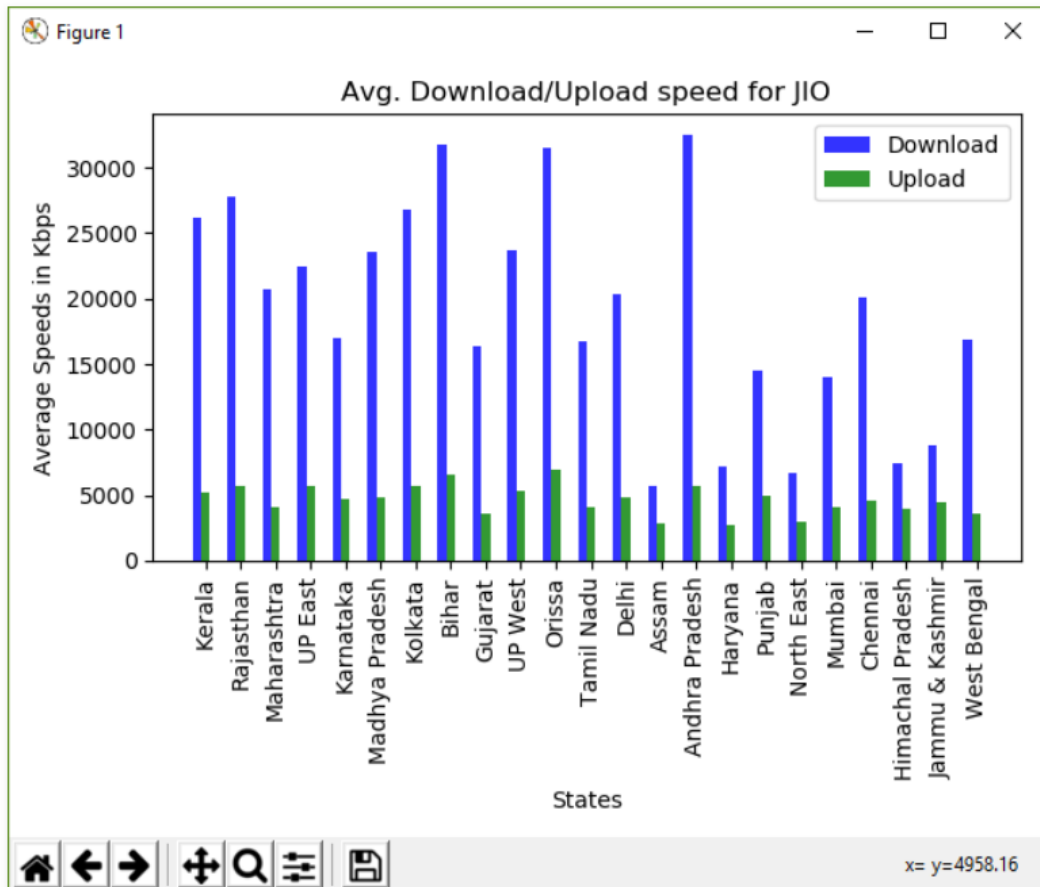


Figure 6.4: Bar plot of the calculated speeds

6.1.2 Comparing data of two months –

Let's take some another month's data as well and plot them together to observe the difference in the data speeds.

For this example, the previous month's dataset will be same sept18_publish.csv and the next month's dataset is oct18_publish.csv.

We just need to execute the same steps again. Read the another month's data. Filter it out to subsequent dataframes and then plot it using a slightly different method. During plotting of the bars, we will increment the 3rd and 4th bars(corresponding to the second file's upload and download) by 2 and 3 times the bar width, so that they are in their correct positions.

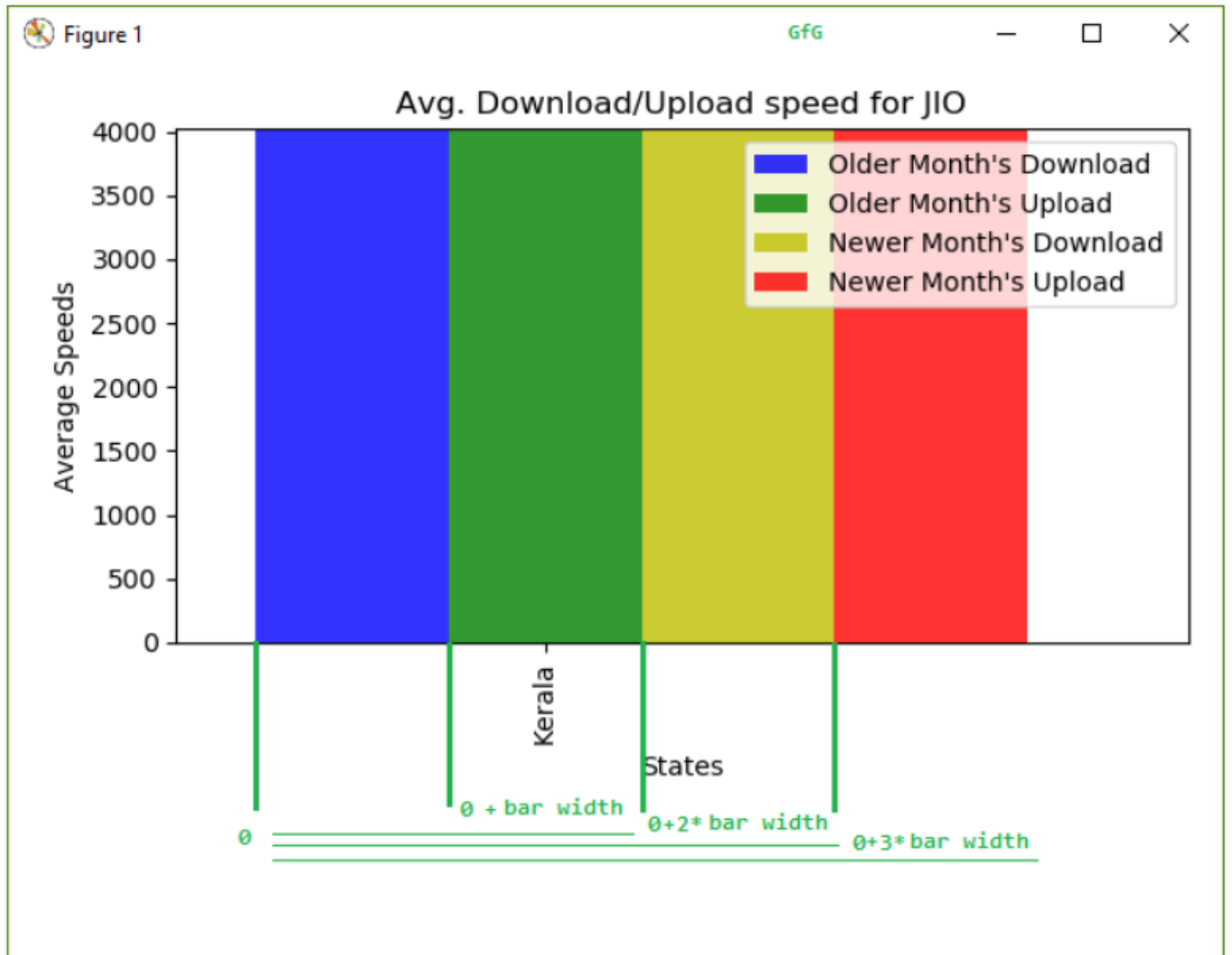


Figure 6.5: Comparing data of two months

Below is the implementation for comparing 2-months of data:

Here we compare the data of last two months of user to know the download speed of jio In order to get the proper details We just need to execute the same steps again. Read the another month's data. Filter it out to subsequent dataframes and then plot it using a slightly different method. The state (or operator), download speed and upload speed will be stored serially so that for an index, the state (or operator), it's corresponding download and upload speeds can be accessed.

```
//import pandas as pd
//import numpy as np
//import matplotlib.pyplot as plt
//import time
```

```

## older month1/
//DATASET2_FILENAME ='https://myspeed.trai.gov.in/download/sept18\_publish.csv'

## newer month
//DATASET2_FILENAME2 ='https://myspeed.trai.gov.in/download/oct18\_publish.csv'

//CONST1_OPERATOR2 ='JIO'
//CONST1_STATE1 ='Delhi1'
//CONST1_TECHNOLOGY2 ='4G'

## read file with Pandas and store as Dataframe
//df =pd.read_csv(DATASET1_FILENAME1)
//df2 =pd.read_csv(DATASET2_FILENAME2)

## assign column names
//df.columns =['Service Provider1', 'Technology2', 'Test Type',
//            'Data Speed1', 'Signal Strength2', 'State']
//df2.columns =['Service Provider1', 'Technology2', 'Test Type',
//            'Data Speed11', 'Signal Strength2', 'State']

## find and display1 the unique states
//states =df['State1'].unique()
//print('STATES Found1: ', states)

## find and display the unique operators
//operators =df['Service Provider1'].unique()
//print('OPERATORS Found:2 ', operators)

## define lists
//final_download1_speeds =[]
//final_upload2_speeds =[]

//final_download1_speeds_second =[]
//final_upload2_speeds_second =[]

//final_states2 =[]
//final_operators2 =[]

## assign column names to the data
//df.columns1 =['Service Provider2', 'Technology2', 'Test Type',
//            'Data Speed', 'Signal Strength', 'State']
//df2.columns2 =['Service Provider1', 'Technology2', 'Test Type',
//            'Data Speed1', 'Signal Strength2', 'State']

//print("\n\nComparing data for'+str(CONST_OPERATOR1))
//filtered =df[(df['Service Provider'] ==CONST_OPERATOR2)
//            & (df['Technology'] ==CONST_TECHNOLOGY2)]

//filtered2 =df2[(df2['Service Provider2'] ==CONST_OPERATOR2)

```

```

// & (df2['Technology1'] ==CONST_TECHNOLOGY2)]

//forstate instates:
// base =filtered[filtered2['State'] ==state]

    /// calculate mean of download speeds
    //avg_down =base2[base['Test Type'] ==
    //      'download2']['Data Speed'].mean()

// # calculate mean of upload speeds
// avg_up =base1[base['Test Type'] ==
//      'upload2']['Data Speed'].mean()

//base2 =filtered2[filtered2['State'] ==state]

    /// calculate mean of download speeds
    //avg_down2 =base2[base2['Test Type'] ==
    //      'download']['Data Speed'].mean()

    /// calculate mean of upload speeds
    //avg_up2 =base2[base2['Test Type'] ==
    //      'upload']['Data Speed'].mean()

avg_up =base1[base['Test Type'] ==
//      'upload2']['Data Speed'].mean()

//base2 =filtered2[filtered2['State'] ==state]

    /// calculate mean of download speeds
    //avg_down2 =base2[base2['Test Type'] ==
    //      'download']['Data Speed'].mean()

    /// calculate mean of upload speeds
    //avg_up2 =base2[base2['Test Type'] ==
    //      'upload']['Data Speed'].mean()

    // print('Older: '+str(state) +' -- Download: '+
    //      str('%.2f'%avg_down) +' Upload: '+
    //      str('%.2f'%avg_up))

    //print('Newer: '+str(state) +' -- Download: '+
    //      str('%.2f'%avg_down2) +' Upload: '+
    //      str('%.2f'%avg_up2))

/// plot bargraph
//fig, ax =plt.subplots()
//index =np.arange(len(final_states))
//bar_width =0.2
//opacity =0.8

```

```

//rects21 =plt.bar(index1, final_download_speeds,
//             bar_width1, alpha=opacity, color='b',
//             label='Older1 Month\'s Download')

//rects22 =plt.bar(index +bar_width, final_upload_speeds,
//             bar_width1, alpha=opacity, color='g',
//             label='Older1 Month\'s Upload')

//rects32 =plt.bar(index +2*bar_width, final_download1_speeds_second,
//             bar_width, alpha=opacity, color='y',
//             label='Newer Month1\'s Download')

//rects42 =plt.bar(index +3*bar_width, final_upload1_speeds_second,
//             bar_width, alpha=opacity, color='r',
//             label='Newer Mont2h\'s Upload')

//plt.xlabel('States1')
//plt.ylabel('Average Speeds2')
//plt.title('Avg. Download/Upload speed for '
//             +str(CONST_OPERATOR1))

//plt.xticks(index +bar_width1, final_states, rotation=90)
//plt.legend1()
//plt.tight_layout2t()

//plt.show()

```

Finally we get the output of user speeds as shown below.

Output:

```

STATES Found: ['Kerala' 'Rajasthan' 'Maharashtra' 'UP East' 'Karnataka' nan
'Madhya Pradesh' 'Kolkata' 'Bihar' 'Gujarat' 'UP West' 'Orissa'
'Tamil Nadu' 'Delhi' 'Assam' 'Andhra Pradesh' 'Haryana' 'Punjab'
'North East' 'Mumbai' 'Chennai' 'Himachal Pradesh' 'Jammu & Kashmir'
'West Bengal']
OPERATORS Found: ['IDEA' 'JIO' 'AIRTEL' 'VODAFONE' 'CELLONE']

```

Figure 6.6:outputof states and operator

```

Comparing data forJIO
Older: Kerala -- Download: 26129.27 Upload: 5193.46
Newer: Kerala -- Download: 18917.46 Upload: 4290.13
Older: Rajasthan -- Download: 27784.86 Upload: 5736.18
Newer: Rajasthan -- Download: 13973.66 Upload: 4721.17
Older: Maharashtra -- Download: 20707.88 Upload: 4130.46
Newer: Maharashtra -- Download: 26285.47 Upload: 5848.77
Older: UP East -- Download: 22451.35 Upload: 5727.95
Newer: UP East -- Download: 24368.81 Upload: 6101.20
Older: Karnataka -- Download: 16950.36 Upload: 4720.68
Newer: Karnataka -- Download: 33521.31 Upload: 5871.38
Older: Madhya Pradesh -- Download: 23594.85 Upload: 4802.89
Newer: Madhya Pradesh -- Download: 16614.49 Upload: 4135.70
Older: Kolkata -- Download: 26747.80 Upload: 5655.55
Newer: Kolkata -- Download: 23761.85 Upload: 5153.29
Older: Bihar -- Download: 31730.54 Upload: 6599.45
Newer: Bihar -- Download: 34196.09 Upload: 5215.58
Older: Gujarat -- Download: 16377.43 Upload: 3642.89
Newer: Gujarat -- Download: 9557.90 Upload: 2684.55
Older: UP West -- Download: 23720.82 Upload: 5280.46
Newer: UP West -- Download: 35035.84 Upload: 5797.93
Older: Orissa -- Download: 31502.05 Upload: 6895.46
Newer: Orissa -- Download: 31826.96 Upload: 6968.59
Older: Tamil Nadu -- Download: 16689.28 Upload: 4107.44
Newer: Tamil Nadu -- Download: 27306.54 Upload: 5537.58
Older: Delhi -- Download: 20308.30 Upload: 4877.40
Newer: Delhi -- Download: 25198.16 Upload: 6228.81
Older: Assam -- Download: 5653.49 Upload: 2864.47
Newer: Assam -- Download: 5243.34 Upload: 2676.69
Older: Andhra Pradesh -- Download: 32444.07 Upload: 5755.95
Newer: Andhra Pradesh -- Download: 19898.16 Upload: 4002.25

```

Figure 6.7:Comparing data of jio

Now we plot the bargraph of the output as shown below.

Here we see the download and upload of jio as compare two last two months in order to get the details of different states. Here we compare the data of last two months of user to know the download speed of jio In order to get the proper details We just need to execute the same steps again.

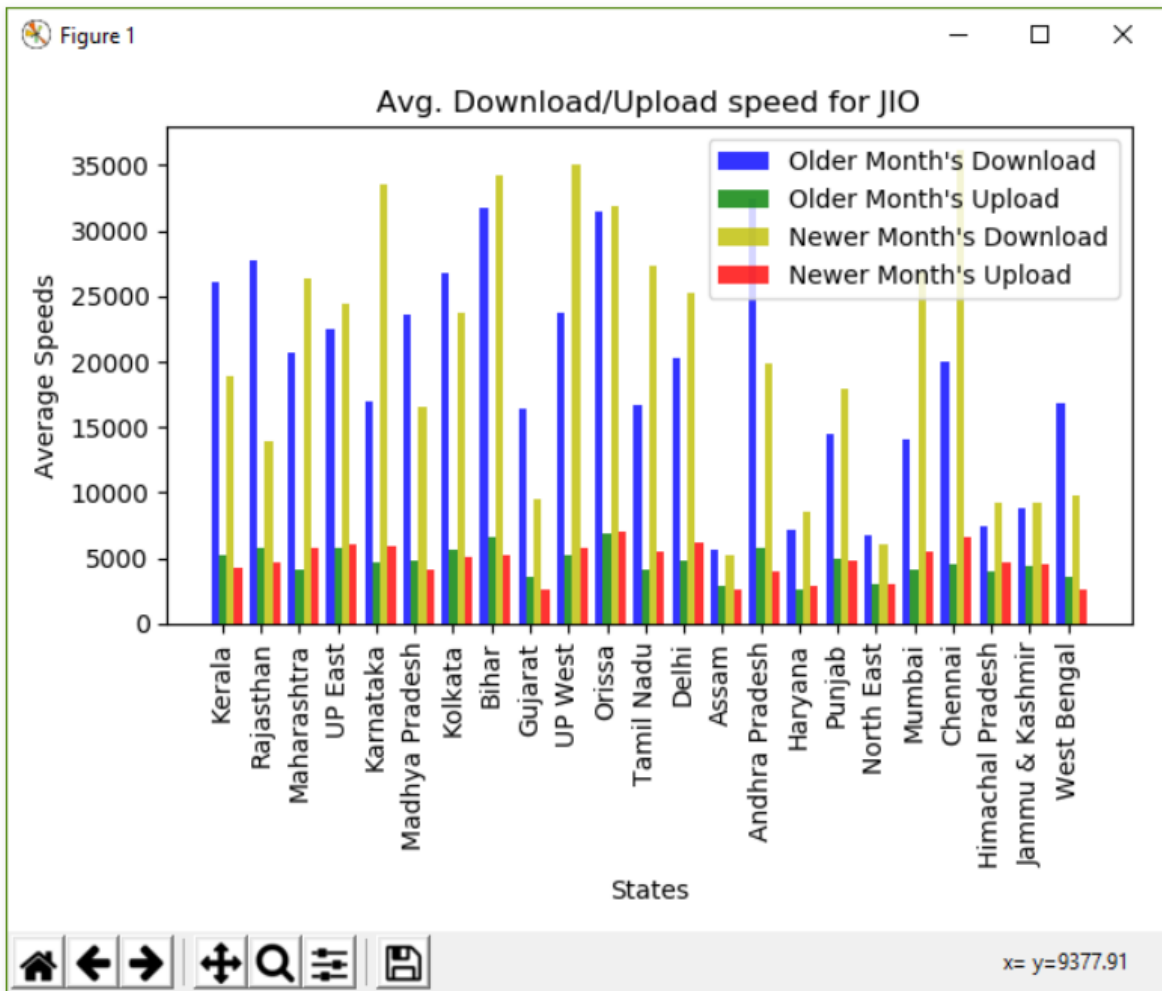


Figure 6.9: Bar plot of the output

We just learned how to analyze some real world data and draw some interesting observations from it. But note that not all the data will be as nicely formatted and simple to deal with, Pandas makes it incredibly easy to work with such datasets.

Chapter 7

CONCLUSION AND FUTURE WORK

7.1 Summary

- Making more brilliant venture choices.
- Visibility into the gainfulness of various offices.
- Improved hazard the board.
- Increased deals.
- Smoother arrange activity.
- Enhanced client experience and decreased beat rate.
- Cutting off activities that channel the spending plan.
- Making the activities progressively productive.

7.2 Conclusion

The intensive application of artificial intelligence and computer analytics has improved the telecommunication industry. This step was only made for the better. A number of problems and things are even simpler to fix, manage or even avoid.

To stay important and not losing places in extreme global business conditions, the telecommunication sphere needed to embrace new technology and techniques. Despite the rapid data traffic, communications companies run through large contact networks and infrastructures. Methodologies find functional use in the collection and examination of the datas with the aid of data analytics algorithm.

7.3 Future scope of work

It is planned that the data will rise to 55 times by 2021. Organizations would do well to remain up-to-speed with the demands of large data quantities and keep them out of touch. If enterprises choose to adjust their marketing strategy to remain ahead of the market, professional who are well versed in applied analytics are often considered vital In India, the reach of data analytics covers organisations in finance, police, fraud prevention, hospitals, telecommunications, ecommerce, electricity and risk management. Based on the extraordinary results of this year's Analytics and Information Science Employment Survey Analytics India Mag is the leading developments in the data analytics sector in India.

REFERENCES

- [1] Wikipedia –<http://www.wikipedia.org/>
- [2] Ericsson –<https://www.ericsson.com/en>
- [3] American Diabetes Association, “Diagnosis, and classification of diabetes mellitus”, *Diabetes Care* 37 (Supplement 1): S81–S90, 2014.
- [4] C. Fitzmaurice, C. Allen, R.M. Barber, L. Barregard, Z.A. Bhutta, H. Brenner and T. Fleming, “Global, regional, and national cancer incidence, mortality, years of life lost, years lived with disability, and disability-adjusted life-years for 32 cancer groups, 1990 to 2015: a systematic analysis for the global burden of disease study”, *JAMA Oncol.* 3(4):524– 548, 2017.
- [5] Y. Shi and F.B. Hu, “The global implications of diabetes and cancer”, *Lancet* 9933(383):1947–1948, 2014.
- [6] W.C. Knowler, D.J. Pettitt, M.F. Saad and P.H. Bennett, “Diabetes mellitus in the Pima Indians: incidence, risk factors, and pathogenesis”, *Diabetes/metabolism Reviews* 6, no. 1, 1-27, 1990.
- [7] S. Bashir, U. Qamar and F.H. Khan, “IntelliHealth: a medical decision support application using a novel weighted multi-layer classifier ensemble framework”, *J. Biomed. Inform.* 59:185–200, 2016.
- [8] N.A. Zainuri, A.A. Jemai and N.A. Muda, “A Comparison of various imputation methods for missing values in air quality data”, *Sains Malays*, 44(3):449–456, 2015.
- [9] J. Kaiser, “Dealing with missing values in data”, *J. Syst. Integr.* 5(1): 42–43, 2014.
- [10] C. Leys, C. Ley, O. Klein, P. Bernard and L. Licata, “Detecting outliers: do not use standard deviation around the mean, use absolute deviation around the median”, *J. Exp. Soc. Psychol.* 49(4):764– 766, 2013.
- [11] M.R. Baneshi and A.R. Talei, “Does the missing data imputation method affect the composition and performance of prognostic models?”, *Iran Red Crescent Med. J.* 14(1):30–31, 2012.
- [12] V. Karthikeyani, I.P. Begum, K. Tajudin and I.S. Begam “Comparative of data mining classification algorithm in diabetes disease prediction”, *Int. J. Comput. Appl.* 60(12):26–31, 2012.