International Journal of Science and Research (IJSR)
ISSN: 2319-7064
ResearchGate Impact Factor (2018): 0.28 | SJIF (2019): 7.583

Python Tools for Big Data Analytics

Lt Col Rahul Dutt Sharma

Faculty of Computer Technology and Studies, Military College of Telecommunication Engineering, Mhow

Abstract: ***Big Data poses an inherent challenge in terms of its interpretation and analytics. Numerous softwares are emerging to refine this data challenge and are bundled with plethora of tools for data correlation and analysis. This paper discusses Big Data, challenges and Use Cases of Big Data to highlight the requirement of Big Data Analytics. It also gives an insight into most popular emerging Python tools for Big Data Analytics along with few of the examples. These libraries provide integrated, intuitive routines for performing common data manipulations and analysis on such data sets. It aims to be the foundation layer for the future of statistical computing in Python.***

Keywords: Big Data, Big Data Analytics, Python, Data Science, Pandas, Numpy, Matplotlib, Scipy

1. Introduction

Big Data is an extensive term for any collection of data sets so large or complex that it becomes difficult to process those using traditional data management techniques such as RDBMS (Relational Database Management Systems). RDBMS has been considered as a one-size- fits-all solution, but the requirements of handling Big Data are varied. Big Data has three V's characteristics, viz. Velocity, Volume and Variety [1].

The data needs to be organized to transform the countless bits and bytes into actionable information. Sheer quantity of data would not be helpful unless we have ways to make sense out of it. Traditionally, programmers wrote code and statisticians did statistics [2]. The programmers coded using a programming language, while statisticians drew interferences using specialized programs such as SPSS (Statistical Package for the Social Sciences) of IBM. Statisticians banked upon the national statistics or market research, whereas programmers managed large amounts of data in databases or log files. But now, Programmers, Statisticians, Scientists, Analysts and infact everyone connected on net has access to a large amount of raw data. With this availability of Big Data from the cloud to virtually each individual changed the entire gambit.

2. Big Data

The term "Big Data" refers to any set of data that is so large or so complex that conventional applications are not adequate to process them. The term also refers to the tools and technologies used to handle "Big Data". Examples of Big Data include the amount of data shared in the internet every day, Amazon website accessed, Twitter feed, Facebook posts and mobile phone location data. In the recent years, data produced by learning environments have also started to get big enough raising the need for Big Data technologies, the tools to handle them and Big Data analytics to analyse that huge amount of data and make valuable decisions for individual or company.

Figure 1: Tools available for data analysis

2.1 Challenges in Handling Big Data

While handling Big Data, various challenges that need to be addressed include [3]:-

a) **Storage**. While the common capacity of hard disks nowadays is in the range of terabytes, the amount of data generated through internet every day is in the order of Exabyte. It is impossible for the traditional RDBMS tools to store and process such Big Data. **Therefore, in order to overcome this challenge, databases should be such that they do not use traditional SQL based queries.** Compression technology is used to compress the data at rest and in memory [4].

b) **Analysis**. As data generated from several types of processes differ in structure and the size of the data is also huge, analysis of the data may consume a lot of time and resources. **To overcome this, scaled out architectures are used to process the data in a distributed manner**. Data are split into smaller pieces and processed in a vast number of computers available throughout the network and the processed data is aggregated.

c) **Reporting**. Generally, the reports are based on statistical data and displaying the same in the form of numbers. Such reports are difficult to be interpreted by human beings if the amount of data involved is large. Hence, the **reports should be represented in an easily comprehendible form by looking** into them. These challenges can be easily overcome using Big Data technologies.

2.2 Types of Big Data Analytics

Big Data Analytics should **not** be treated as a **one- size-fits-**

all blanket technique. Best Data Analysts have the ability to identify the kind of analytics that can be leveraged for gaining benefits for their particular requirement. There are four dominant types of analytics available today [1]:-

a) **Descriptive Analytics**. Data Analytics is used to **describe raw data** and **present it in a form that** is easily **interpretable by human beings**. It **analyses past events** that allows organizations to learn from past behaviours, and help them influence future outcomes. Descriptive statistics are useful in showing things like total stock in inventory or average money spent per customer. **Amazon** has more than 1.5 billion items in its retails stores spread across 200 centres across the world and it uses Big Data to monitor, track and secure them. Amazon stores the product catalogue data in its S3 cloud that can write, read and delete objects up to 5 TB of data each. The catalogue stored in cloud receives more than 50 million updates a week and every 30 minutes all data received is reported back to the different warehouses and the website.

b) **Diagnostic Analytics**. Diagnostic analytics focuses on examining the data/content to find answer to the question **"Why did it happen?"**. It is characterized by techniques like drill-down, data discovery, data mining and correlations. It examines the database deeply to **understand the causes of events and behaviours**. One of the medical diagnostics company analysed database of millions of records to develop first non-intrusive test to predict coronary artery disease. This diagnostic based on the analysis of approximately 100 million samples was conducted to finally identify the 23 primary predictive genes for coronary artery disease.

c) **Predictive Analytics**. Predictive analytics is useful in **"Predicting" the next sequence of events likely to occur.** This provides the organizations with **actionable insights based on data**. Moreover, it also provides estimates of the likelihood of a future outcome though, no statistical algorithm can "predict" the future with 100% guarantee. A very popular application of predictive analytics is to predict the credit score. These scores play an important role for the financial institutions to predict the probability of timely payment by the customers in future.

d) **Prescriptive Analytics**. Prescriptive analytics is a relatively new field and it facilitates users to **"prescribe" options of diverse possible actions** to be implemented and also guide them towards a viable solution. It presents the quantified effect of future decisions in order to advise on all the possible outcomes before those decisions are actually made on ground. Not only does it predicts what will happen, but it also tells the reason of why it will happen and thereby provide recommendations. For example, Netflix uses Big Data Analytics to prescribe favourite song/movie based on customer's interests, behaviour, day and time analysis.

3. Python For Big Data Analytics

3.1 Advantages of Python for Big Data Analytics

Python is the most popular language amongst Data Scientists for Data Analytics not only because of its ease in programming but also due to several other in-built features that Python offers. Few of the most important features are enumerated as under [5]:-

a) **Best Library for Big Data Analytics**. Before Python tools for Big Data, Data Science was something only statisticians, operations researchers, and applied mathematicians could do. Python libraries like Pydoop and Scipy are the best alternatives for data scientists who in the past used more scientific tools like Wolfram Alpha and Mathematics, Matplotlib, R to name a few.

b) **Portable Library and Tools**. Just a couple of lines of code can help the user browse, summarize, and arrange data, just like Excel. That is of more significance if they use scratch pads like Jupyter, as they support creating graphs and tables with one button.

c) **Extensibility**. Python code is extensible over multiple platforms. Many Machine Learning algorithms are written in Python. These include Google TensorFlow, Microsoft Cognitive ToolKit, Scikit-learn and Spark ML (Spark Machine Learning).

d) **Diverse Data Structure Support**. It supports latest data structures like maps and sets in addition to the primitive types like integers and complex numbers. Python tools like Numpy supports matrices, lists, linear algebra and many more libraries required for Data Analytics and Machine Learning Algorithms

e) **Python Command Line Shell**. Apache Spark has a Python shell. That means user can open datasets, do transformations, and run algorithms in one easy command line. Without that user would have to package the program and then submit it to Spark using spark-submit.

f) **Data Transformation Made Easier**. Imagine you can type a few lines or code and have Python read a JSON, CSV, or other file type and then transform that into a Pandas table, which embeds the excel table and flattens it making it easier to read and apply column headings, further making things easier to read. It's as if you took a hard-to-read CSV file and laid it out in familiar spread sheet formation. And then small chunk of this data may be used to rotate rows and columns, apply functions to individual cells and format this data as per user requirement.

3.2 Python Tools for Big Data Analytics

Python has an overwhelming number of packages that can be used in Big Data Analytics and Machine Learning. The Python machine learning ecosystem can be divided into three main types of packages as shown in figure 8.

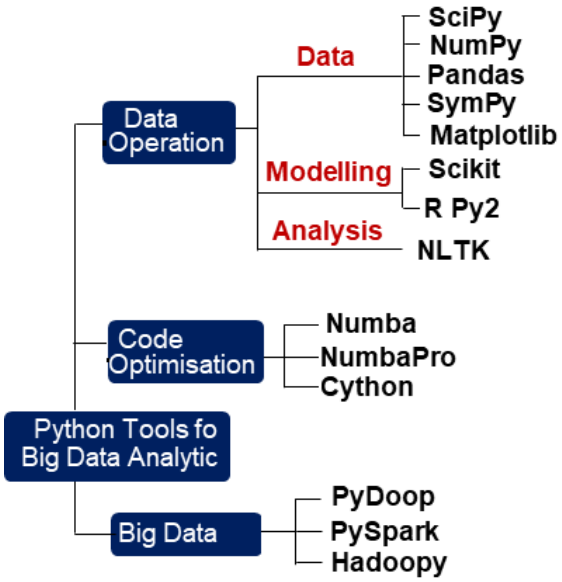

Figure 2: Popular Python Tools for Big Data Analytics

SciPy: SciPy is pronounced as 'Sigh Pie'. It is a Python based open-source software for mathematics, science, and engineering [7]. It provides a vast variety of modules for linear algebra, interpolation & integration, special functions for Fast Fourier Transform, signal processing and image transformation. SciPy is basically a library that integrates basic packages used in scientific computing such as NumPy, Matplotlib, Pandas and SymPy. It is a collection of packages used to solve a large variety of standard problem domains in scientific computing. It can be utilised for standard continuous and discrete probability distributions (density functions, samplers, continuous distribution functions), various statistical tests and more descriptive statistics[7].

a) Discrete Fourier Transform (DFT)

```
In [28]: %matplotlib inline
from matplotlib import pyplot as plt
import numpy as np

#Frequency in terms of Hertz
fre  = 5
#Sample rate
fre_samp = 50
t = np.linspace(0, 2, 2 * fre_samp, endpoint = False )
a = np.sin(fre  * 2 * np.pi * t)
figure, axis = plt.subplots()
axis.plot(t, a)
axis.set_xlabel ('Time (s)')
axis.set_ylabel ('Signal amplitude')
plt.show()
```

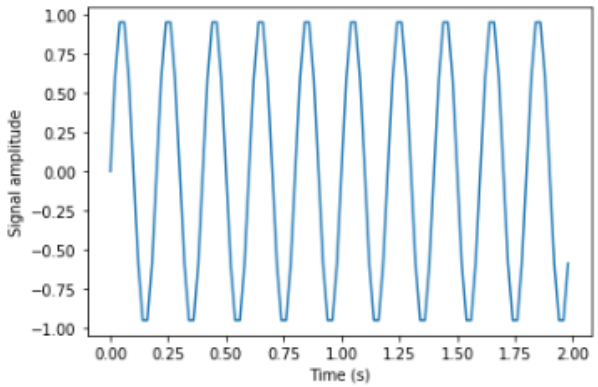

Figure 3: Depiction of Wave using DFT

b) Image Processing

```
In [29]: from scipy import misc
from matplotlib import pyplot as plt
import numpy as np
#get face image of panda from misc package
panda = misc.face()
#plot or show image of face
plt.imshow( panda )
plt.show()
```


```
In [30]: flip_down = np.flipud(misc.face())
plt.imshow(flip_down)
plt.show()
```

Figure 4: Easy Image Processing

c) Interpolation

```
In [2]: import numpy as np
        from scipy import interpolate
        import matplotlib.pyplot as plt
        x=np.linspace(0,4,12)
        y=np.cos(x**2/3+4)
        print (x,y)

[0.         0.36363636 0.72727273 1.09090909 1.45454545 1.81818182
 2.18181818 2.54545455 2.90909091 3.27272727 3.63636364 4.        ] [-0.65364362 -0.61966189 -0.51077021 -0.31047698 -0.0071547
6  0.37976236
  0.76715099  0.99239518  0.85886263  0.27994201 -0.52586509 -0.99582185]

In [41]: plt.plot(x, y,'x')
         plt.show()
```

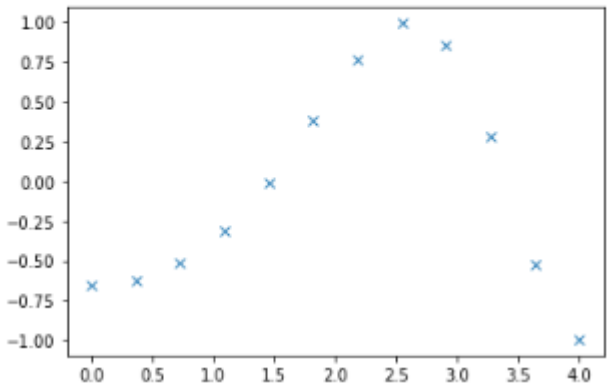

Figure 5: Graphical Interpolation of Arrays

Numpy: NumPy is an abbreviation for Numerical Python. It is a very useful library to store and operate on dense data buffers. It is an open source Python library that is very effictive in mathematical, scientific, engineering, and data science programming. It is extremely efficient to work with the N-dimensional array, matrices, linear algebra, random number, Fourier transform, etc. [5]. NumPy arrays are similar to Python's built-in list data type, but NumPy arrays allow much more efficient storage and data operations as the arrays grow larger in size. NumPy arrays form the core of nearly the entire ecosystem of data science tools in Python [6]. Numpy is very convenient to work with, memory efficient and fast. Various features of Numpy are enumerated as under:-

Mathematical Operations

```
In [56]: a = [[1,2],[3,4]]
         b = [[5,6],[7,8]]

In [57]: np.matmul(a,b)

Out[57]: array([[19, 22],
                [43, 50]])

In [8]:  numpyarray + 10
Out[8]:  array([11, 12, 13, 14, 15, 16, 17])
In [9]:  numpyarray *10
Out[9]:  array([10, 20, 30, 40, 50, 60, 70])
In [10]: numpyarray/2
Out[10]: array([0.5, 1. , 1.5, 2. , 2.5, 3. , 3.5])
```

Figure 6: Mathematical Operations on Numpy array

Reshape Data. Data analytics, at many occasions, need to reshape the data from wide to long. Numpy provides numpy.reshape() and numpy.flatten() to implement it easily.

```
In [25]: e  = np.array([(1,2,3), (4,5,6)])
In [26]: e
Out[26]: array([[1, 2, 3],
                [4, 5, 6]])
In [27]: e.reshape(3,2)
Out[27]: array([[1, 2],
                [3, 4],
                [5, 6]])
```

Figure 7: Reshaping of Numpy Array

Statistical Functions: NumPy provides various statistical functions to determine minimum, maximum, percentile standard deviation and variance from the elements in an array. With just a single line of code all the statistical functions can be performed that are of much importance in Big Data Analytics.

```
In [47]: array1 = np.random.normal(5, 0.5, 10)
In [48]: array1
Out[48]: array([5.18351279,5.00522577,5.92183742,
                4.46941753, 4.40730874,5.16062572,
                5.84246506,4.94775136,4.94911142,4.3573384 ])
In [49]: print(np.min(array1))
         4.357338398182874
In [50]: print(np.max(array1))
         5.9218374209777425
In [51]: print(np.mean(array1))
         5.0244594214906915
In [52]: print(np.median(array1))
         4.977168591194171
In [53]: print(np.std(array1))
         0.5161577321123502
```

Figure 8: Statistical Functions

Pandas: Pandas is a high-performance, very easy-to-use, data-wrangling package. It has a special data type called Data frames, a type of in-memory data table. Pandas provide diverse data structures and functions to aid in working with structured data in a fast, easy, and expressive way. It is one of the important libraries enabling Python to be a powerful and productive Data Analytics Tool. Pandas combine high performance array-computing features of NumPy with the flexible data handling features of spread sheets and RDBMS (such as SQL) [10]. It has a sophisticated indexing functionality to make it easy to reshape, slice and dice, perform aggregations, and also select subsets of data. It contains well defined data structures and manipulation tools to make data analysis fast and easy.

Pandas Data Structures: Pandas has two types of data structures, Series and DataFrames. Series is a one dimensional data structure ("a one dimensional array") to store values and it has a unique index for every value it holds. A DataFrame is similar to a table or a spread sheet and it contains an ordered collection of columns. Each column can have a different value type (numeric, string, boolean, etc). The DataFrame provides an auto row and column index for the values[6].

```
In [28]: from pandas import Series, DataFrame

In [29]: import numpy

In [31]: s1=Series([5,6,3,7,8,-34,76])

In [32]: s1
Out[32]: 0     5
         1     6
         2     3
         3     7
         4     8
         5   -34
         6    76
         dtype: int64

In [34]: data={'student_roll:':[1,2,3,4,5,6],
              'student_name:':['shakti','anurag','vivek','acb','afds','bvc'],
              'studetn_addr:':['street1','s2','s3','s4','s5','s6']}

In [35]: frame=DataFrame(data)

In [36]: frame
```

Out[36]:

	student_roll:	student_name:	studetn_addr:
0	1	shakti	street1
1	2	anurag	s2
2	3	vivek	s3
3	4	acb	s4
4	5	afds	s5
5	6	bvc	s6

Figure 9: Series & Data Frames in pandas

Filters: For effective data analysis, our tools should provide filter facility to sort data with to respect one or more fields. Panda provides this facility very effectively, in this we can take out data equivalent to some field or print data having a field common in all the records. Let's say, we want to see a list of only the users who belong to country "country 7". This filtering of data at run-time is very valuable in Big Data Analytics for sorting the values amongst billions of data sets available.

```
In [18]: article_str[article_str.country == 'country_7']
```

Out[18]:

	my_datetime	event	country	user_id	source	topic
0	2018-01-01 00:01:01	read	country_7	2458151261	SEO	North America
1	2018-01-01 00:03:20	read	country_7	2458151262	SEO	South America
2	2018-01-01 00:04:01	read	country_7	2458151263	AdWords	Africa
3	2018-01-01 00:04:02	read	country_7	2458151264	AdWords	Europe
8	2018-01-01 00:07:21	read	country_7	2458151269	AdWords	North America
11	2018-01-01 00:08:57	read	country_7	2458151272	SEO	Australia
14	2018-01-01 00:11:06	read	country_7	2458151275	Reddit	Africa
15	2018-01-01 00:11:22	read	country_7	2458151276	SEO	North America
21	2018-01-01 00:15:50	read	country_7	2458151282	Reddit	Africa
24	2018-01-01 00:17:58	read	country_7	2458151285	Reddit	Africa
29	2018-01-01 00:23:51	read	country_7	2458151290	Reddit	Asia
31	2018-01-01 00:26:28	read	country_7	2458151292	Reddit	Australia
32	2018-01-01 00:26:48	read	country_7	2458151293	Reddit	South America
33	2018-01-01 00:27:10	read	country_7	2458151294	Reddit	Australia
37	2018-01-01 00:31:24	read	country_7	2458151298	AdWords	Europe

Figure 10: Filtered Data

Data Aggregation, Grouping & Formatting: Pandas can very easily perform various data aggregation functions like min, max, count, sum etc. A Data Analyst requires detailed segmentation to display output or analyse data. For this, grouping of data according to specific field(s) is required. In Fig 11, grouping of data has been done as per animal name and the mean of water requirement per animal is calculated with just a single line of code. Pandas has various formatting methods like merge, sort, reset index, fill etc. Data available to Data Analytics from various sources is generally in smaller tables. Therefore, they need to pull out data from two or more different tables. The solution for that is provided by Formatting functions like merge.

```
In [32]: jungle.groupby('animal').mean()
```

Out[32]:

animal	uniq_id	water_need
elephant	1002.0	550.000000
kangaroo	1021.0	416.666667
lion	1017.5	477.500000
tiger	1006.0	310.000000
zebra	1012.0	184.285714

```
In [33]: jungle.groupby('animal').mean()[['water_need']]
```

Out[33]:

animal	water_need
elephant	550.000000
kangaroo	416.666667
lion	477.500000
tiger	310.000000
zebra	184.285714

Figure 11: Grouping & Mean

PyDoop: Python can be used to write Hadoop MapReduce programs and applications to access HDFS API for Hadoop using PyDoop. PyDoop offer various advantages over Hadoop's in-built solutions for Python programming, i.e., Hadoop Streaming and Jython. One of the biggest advantages of PyDoop is its HDFS API [9] that allows to connect to an HDFS installation, read and write files and get information on files, directories and global file system properties. The MapReduce API of PyDoop is very effective in solving complex problems with minimal programming efforts. Advance MapReduce concepts such as 'Counters' and 'Record Readers' can be easily incorporated in Python using PyDoop.

Numba and NumbaPro: Numba and NumbaPro are just-in-time compilation to speed up applications written directly in

Python and a few annotations. NumbaPro also allows user to use the power of graphics processor unit (GPU). The combination of NumPy array with Numba library provides the best performance for data manipulation and analysis [5].

SymPy: SymPy is an acronym for Symbolic Mathematics in Python. It is very useful to simplify complex mathematical expressions, compute derivatives, integrals and limits, solve equations and matrix operations. SymPy include features such as modules for plotting co-ordinate modes, Geometric entities, 2D and 3D, interactive interface. It is capable of formatting the result of the computations as LaTeX code [2][8].

Matplotlib: Matplotlib is a popular 2D plotting package with some 3D [8] functionality. Matplotlib is a very effective Python library for producing plots and other 2D data visualizations. It is specially effective in creating plots suitable for publication. It integrates well with IPython, thereby providing an interactive environment for plotting and exploring data. The plots are also interactive, the users can zoom in on a section of the plot and pan around the plot using the toolbar in the plot window.

PySpark: PySpark is the combination of Apache Spark and Python. It consists of a wide range of libraries primarily used for Machine Learning and Real-Time Streaming Analytics. In other words, PySpark is a Python API for Spark that lets you harness the simplicity of Python and the power of Apache Spark in order to tame Big Data [6]. It provides an API that can be used to solve parallel data proceeding problems and handle complexities of multiprocessing. PySpark has low latency because of the in-memory processing, its framework is compatible with various languages like Scala, Java, Python and R, which makes it one of the most preferable frameworks for processing huge datasets and provides powerful caching. PySparks extends a PySparkSQL library to apply SQL-like analysis and queries on a huge amount of structured or semi-structured data. It has a GraphFrames is library that provides APIs for performing graph analysis efficiently using the PySpark core and PySparkSQL. It is optimized for fast distributed computing.

4. Conclusion

Big Data Analytics is a task that can be achieved only by obtaining structured data by well-defined software tools and programming languages that offer high computational and scientific features. Python tools provide a solid foundation upon which a very powerful data analysis ecosystem can be established. The features offered by various Python tools are apt for Data Analytics and at the same time they are memory efficient and fast. Tools like NumPy, SciPy and Pandas are very well established in terms of their extensible modules and are widely used by the Data Analytics. Immense number of tools based on Python are still evolving to assist in Data Analytics. Python tools are a compelling choice for Big Data Analytics applications due to the easy to use data structures that cohere with the rest of the scientific Python stack.

References

[1] Davy Cielen, Arno Meysma, Mohd Ali – "Introducing Data Science, Big Data, Machine Learning and more using Python tools".
[2] Abhinav Nagpal, Goldie Gabrani – "Python for Data Analytics, Scientific and Technical Applications".
[3] Katrina Sin and Loganathan Muthu- "Application of Big Data in Education Data Mining and Learning Analytics - A Literature Review".
[4] G V Rossum - "Computer Programming for Everybody-A Scouting Expedition for the Programmers of Tomorrow," CNRI Proposal 90120-1a, Corporation for National Research Initiatives, 1999.
[5] Wes McKinney – "Python for Data Analysis".
[6] Wikipedia – "https://en.wikipedia.org/wiki/SciPy".
[7] Jake VanderPlas – "Python Data Science Handbook".
[8] Vivekananth P, Leo John Baptist A – "An Analysis of Big Data Analytics Techniques" Volume 5, Issue-5, October-2015 International Journal of Engineering and Management Research.
[9] Wes McKinney – "Pandas: A Foundational Python Library for Data Analysis and Statistics".