COGNITENSOR
an AI COMPANY

A Beginner's Guide

to

<u>Data Science</u>

Ashish Airon
Co Founder
https://www.linkedin.com/in/ashishairon/

Content

- Which programming language to learn?
- Essentials for Coding and Data Science
- Essential Libraries
- Statistics
- Visualization Libraries
- Exploratory Data Analysis
- Machine Learning Algorithms
- Deployment of Model
- Deep Learning
- Frameworks of Python
- Databases
- Web Scraping and APIs
- Visualization Tools
- Competitive Data Science
- Essential Newsletters for Data Scientists
- Stages to Solve Analytical Problems
- Intelligence through Analytics
- Resources

Which programming language to learn? (Python vs. R)

Python	R
It is a general-purpose programming language, mainly used for GUI and web applications along with data analysis.	It was designed with features to ease statistical computing and data analysis.
Programs written in Python are faster as compared to R.	Programs written in R are comparatively slower.
It is a highly flexible programming language.	It is a not so flexible programming language.
It is easy to learn.	Beginners without any prior experience in programming language find it slightly harder to learn R.
It is an object-oriented language.	It is a procedural language.
Python doesn't have specialized packages for statistical computing.	R consists of specialized packages for statistical computing.

Interoperability:

A programmer has an option to run R code through Python and vice versa. Although, developers have to use some libraries to integrate one language into another.

To convert Python objects into R objects, the developer can use RPy2, and to call Python script along with procedures of R, the developer can use Python.

Therefore, a data analyst should be flexible enough and work according to the precise needs of the project.

Python

1. Various studies have concluded that Python is the most essential language to be learned by an upcoming data scientist. In 2019, almost 75% of the industry, as well as the professionals supported this statement. For becoming an upcoming data scientist, one should at least learn Python for six months and should start learning other languages(R, Java, etc) only after that.

2. Here we'll be talking about Python and its libraries. So, if you are opting for Python in order to learn data science, then this is the right place for you.

3. Python
   - Working of Python programming language
   - Python Data Structures: Lists, String, Tuple, Dictionary
   - Methods, Exception Handling
   - Concepts of OOPS (object-oriented programming)

Essentials for Coding and Data Science

- To start with coding for data science, some basic knowledge of any programming language is necessary. Here is a helpful link to Python tutorial.

- Once some basic knowledge of Python is acquired, the next steps include practicing at a regular interval. This interval can be anywhere between 1-2 hours a day. It is suggested that one should try challenging coding problems such that on websites like Leet Code and Hackerrank.

- After gaining experience in coding, moving to learn about data science should be the next step. Here is link to edX free courses are real college courses from Harvard, MIT, and more of the world's leading universities.

- If you are an avid reader, this site provides ordered topics for Python, Machine Learning, Artificial Intelligence and Data Science.

Essential Libraries

Although there are many libraries, the following are the most basic and important ones:-

1. **NumPy:** It helps to create arrays, with the help of bindings of C++. Therefore, it is quite fast. There are in-built functions of NumPy as well!

   Why do we need a NumPy array when we have python lists ?
   We can perform operations on all the elements of a NumPy array at once which is not possible with python lists. Some basic operations_are: Reshaping of Arrays, Indexing Techniques, Broadcasting of Arrays, and so on.

- Importing NumPy:
  import numpy as np

- Converting a Python list into NumPy Array :
  height=[170, 176, 189, 160, 165, 150]
  newArray=np.array(height)

- Reshaping a NumPy Array:
  newArray=newArray.reshape(2,3
)

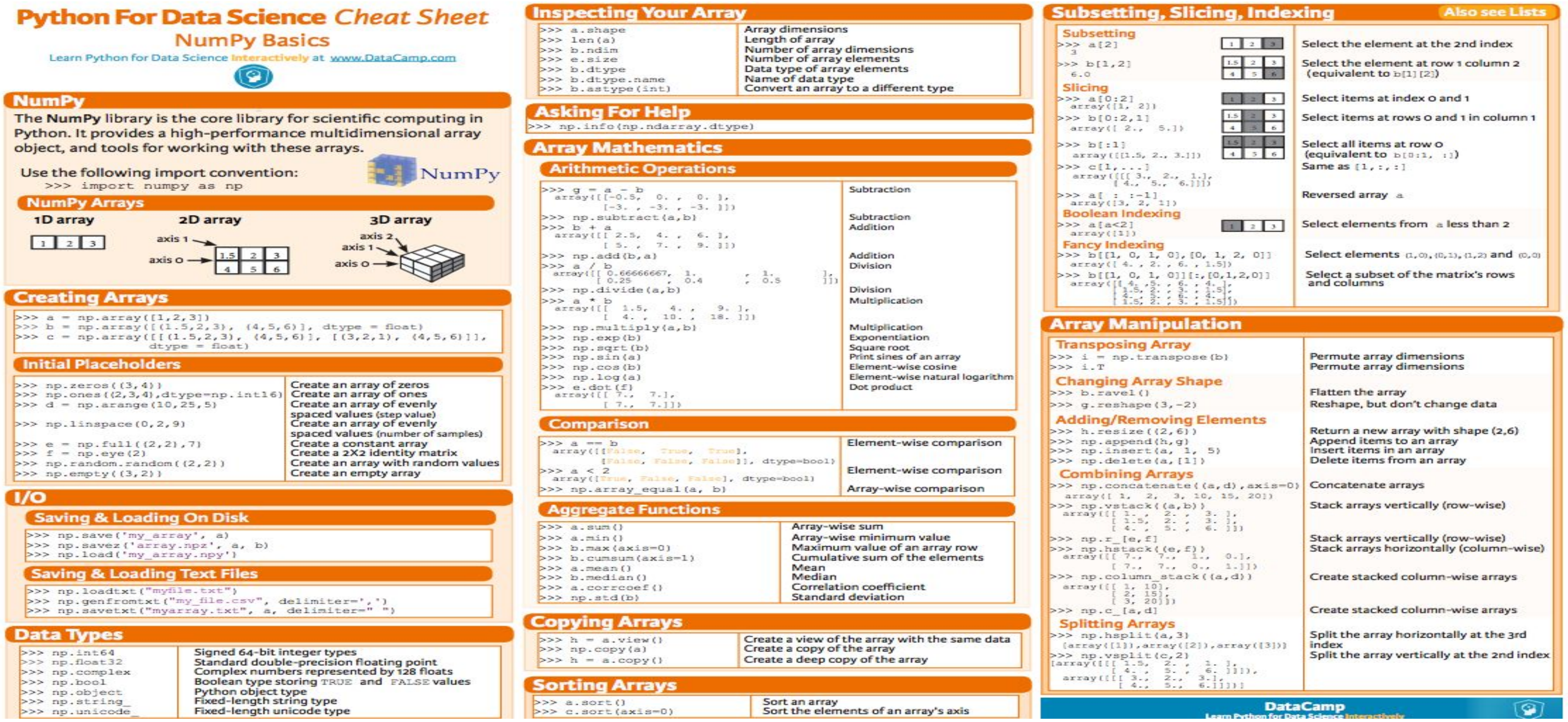
- Dimension of NumPy Array:
  print(newArray.ndim)

- Shape of NumPy Array:
  print(newArray.shape)

Num Py

COGNITENSOR an AI COMPANY

Numpy & Python Cheat Sheet



Python For Data Science *Cheat Sheet*

NumPy Basics

Learn Python for Data Science Interactively at www.DataCamp.com

NumPy

The **NumPy** library is the core library for scientific computing in Python. It provides a high-performance multidimensional array object, and tools for working with these arrays.

Use the following import convention:

```
>>> import numpy as np
```

NumPy

NumPy Arrays

1D array | 2D array | 3D array

| 1 | 2 | 3 |

axis 1, axis 0

| 1.5 | 2 | 3 |
| 4 | 5 | 6 |

axis 2, axis 1, axis 0

Creating Arrays

```
>>> a = np.array([1,2,3])
>>> b = np.array([(1.5,2,3), (4,5,6)], dtype = float)
>>> c = np.array([[(1.5,2,3), (4,5,6)], [(3,2,1), (4,5,6)]],
                 dtype = float)
```

Initial Placeholders

`>>> np.zeros((3,4))`	Create an array of zeros
`>>> np.ones((2,3,4),dtype=np.int16)`	Create an array of ones
`>>> d = np.arange(10,25,5)`	Create an array of evenly spaced values (step value)
`>>> np.linspace(0,2,9)`	Create an array of evenly spaced values (number of samples)
`>>> e = np.full((2,2),7)`	Create a constant array
`>>> f = np.eye(2)`	Create a 2X2 identity matrix
`>>> np.random.random((2,2))`	Create an array with random values
`>>> np.empty((3,2))`	Create an empty array

I/O

Saving & Loading On Disk

```
>>> np.save('my_array', a)
>>> np.savez('array.npz', a, b)
>>> np.load('my_array.npy')
```

Saving & Loading Text Files

```
>>> np.loadtxt("myfile.txt")
>>> np.genfromtxt("my_file.csv", delimiter=',')
>>> np.savetxt("myarray.txt", a, delimiter=" ")
```

Data Types

`>>> np.int64`	Signed 64-bit integer types
`>>> np.float32`	Standard double-precision floating point
`>>> np.complex`	Complex numbers represented by 128 floats
`>>> np.bool`	Boolean type storing `TRUE` and `FALSE` values
`>>> np.object`	Python object type
`>>> np.string_`	Fixed-length string type
`>>> np.unicode_`	Fixed-length unicode type

Inspecting Your Array

`>>> a.shape`	Array dimensions
`>>> len(a)`	Length of array
`>>> b.ndim`	Number of array dimensions
`>>> e.size`	Number of array elements
`>>> b.dtype`	Data type of array elements
`>>> b.dtype.name`	Name of data type
`>>> b.astype(int)`	Convert an array to a different type

Asking For Help

```
>>> np.info(np.ndarray.dtype)
```

Array Mathematics

Arithmetic Operations

`>>> g = a - b`	Subtraction
`array([[-0.5, 0. , 0. ],`	
`[-3. , -3. , -3. ]])`	
`>>> np.subtract(a,b)`	Subtraction
`>>> b + a`	Addition
`array([[ 2.5, 4. , 6. ],`	
`[ 5. , 7. , 9. ]])`	
`>>> np.add(b,a)`	Addition
`>>> a / b`	Division
`array([[ 0.66666667, 1. , 1. ],`	
`[ 0.25 , 0.4 , 0.5 ]])`	
`>>> np.divide(a,b)`	Division
`>>> a * b`	Multiplication
`array([[ 1.5, 4. , 9. ],`	
`[ 4. , 10. , 18. ]])`	
`>>> np.multiply(a,b)`	Multiplication
`>>> np.exp(b)`	Exponentiation
`>>> np.sqrt(b)`	Square root
`>>> np.sin(a)`	Print sines of an array
`>>> np.cos(b)`	Element-wise cosine
`>>> np.log(a)`	Element-wise natural logarithm
`>>> e.dot(f)`	Dot product
`array([[ 7., 7.],`	
`[ 7., 7.]])`	

Comparison

`>>> a == b`	Element-wise comparison
`array([[False, True, True],`	
`[False, False, False]], dtype=bool)`	
`>>> a < 2`	Element-wise comparison
`array([True, False, False], dtype=bool)`	
`>>> np.array_equal(a, b)`	Array-wise comparison

Aggregate Functions

`>>> a.sum()`	Array-wise sum
`>>> a.min()`	Array-wise minimum value
`>>> b.max(axis=0)`	Maximum value of an array row
`>>> b.cumsum(axis=1)`	Cumulative sum of the elements
`>>> a.mean()`	Mean
`>>> b.median()`	Median
`>>> a.corrcoef()`	Correlation coefficient
`>>> np.std(b)`	Standard deviation

Copying Arrays

`>>> h = a.view()`	Create a view of the array with the same data
`>>> np.copy(a)`	Create a copy of the array
`>>> h = a.copy()`	Create a deep copy of the array

Sorting Arrays

`>>> a.sort()`	Sort an array
`>>> c.sort(axis=0)`	Sort the elements of an array's axis

Subsetting, Slicing, Indexing

Also see Lists

Subsetting

`>>> a[2]`	Select the element at the 2nd index
`3`	
`>>> b[1,2]`	Select the element at row 1 column 2 (equivalent to `b[1][2]`)
`6.0`	

Slicing

`>>> a[0:2]`	Select items at index 0 and 1
`array([1, 2])`	
`>>> b[0:2,1]`	Select items at rows 0 and 1 in column 1
`array([ 2., 5.])`	
`>>> b[:1]`	Select all items at row 0 (equivalent to `b[0:1, :]`)
`array([[1.5, 2., 3.]])`	
`>>> c[1,...]`	Same as `[1,:,:]`
`array([[[ 3., 2., 1.],`	
`[ 4., 5., 6.]]])`	
`>>> a[ : :-1]`	Reversed array `a`
`array([3, 2, 1])`	

Boolean Indexing

`>>> a[a<2]`	Select elements from `a` less than 2
`array([1])`	

Fancy Indexing

`>>> b[[1, 0, 1, 0],[0, 1, 2, 0]]`	Select elements `(1,0),(0,1),(1,2)` and `(0,0)`
`array([ 4. , 2. , 6. , 1.5])`	
`>>> b[[1, 0, 1, 0]][:,[0,1,2,0]]`	Select a subset of the matrix's rows and columns
`array([[ 4. ,5. , 6. , 4. ],`	
`[ 1.5, 2. , 3. , 1.5],`	
`[ 4. , 5. , 6. , 4. ],`	
`[ 1.5, 2. , 3. , 1.5]])`	

Array Manipulation

Transposing Array

`>>> i = np.transpose(b)`	Permute array dimensions
`>>> i.T`	Permute array dimensions

Changing Array Shape

`>>> b.ravel()`	Flatten the array
`>>> g.reshape(3,-2)`	Reshape, but don't change data

Adding/Removing Elements

`>>> h.resize((2,6))`	Return a new array with shape (2,6)
`>>> np.append(h,g)`	Append items to an array
`>>> np.insert(a, 1, 5)`	Insert items in an array
`>>> np.delete(a,[1])`	Delete items from an array

Combining Arrays

`>>> np.concatenate((a,d),axis=0)`	Concatenate arrays
`array([ 1, 2, 3, 10, 15, 20])`	
`>>> np.vstack((a,b))`	Stack arrays vertically (row-wise)
`array([[ 1. , 2. , 3. ],`	
`[ 1.5, 2. , 3. ],`	
`[ 4. , 5. , 6. ]])`	
`>>> np.r_[e,f]`	Stack arrays vertically (row-wise)
`>>> np.hstack((e,f))`	Stack arrays horizontally (column-wise)
`array([[ 7., 7., 1., 0.],`	
`[ 7., 7., 0., 1.]])`	
`>>> np.column_stack((a,d))`	Create stacked column-wise arrays
`array([[ 1, 10],`	
`[ 2, 15],`	
`[ 3, 20]])`	
`>>> np.c_[a,d]`	Create stacked column-wise arrays

Splitting Arrays

`>>> np.hsplit(a,3)`	Split the array horizontally at the 3rd index
`[array([1]),array([2]),array([3])]`	
`>>> np.vsplit(c,2)`	Split the array vertically at the 2nd index
`[array([[[ 1.5, 2. , 1. ],`	
`[ 4. , 5. , 6. ]]]),`	
`array([[[ 3., 2., 3.],`	
`[ 4., 5., 6.]]])]`	

DataCamp
Learn Python for Data Science Interactively

For high quality resolution please click here

Essential Libraries (contd.)

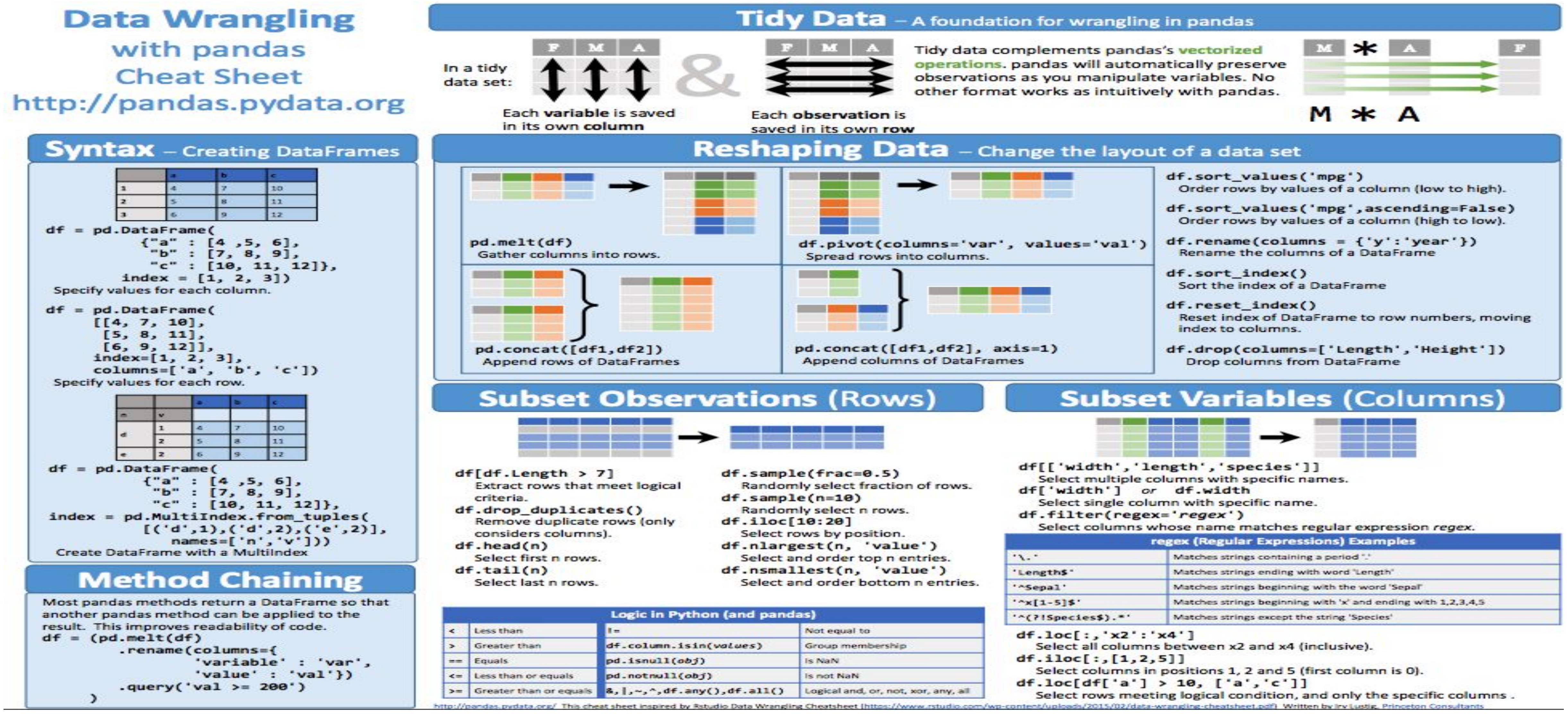
2. **Pandas:** One must gain in-depth knowledge about this library. It is a very important library for data pre-processing. Learn about data frames, and data series and operations present inside them. Gaining insights about Feature Engineering techniques present inside data frames, handling of missing values, scaling values according to the range.

Basic Operations:

- Importing Pandas:
  import pandas as pd
- Reading a CSV file and Creating a dataframe:
  df=pd.read_csv("xyz.csv")
- Print first and last5 rows of Dataframe:
  df.head()
  df.tail
- Basic Information about the Dataframe:
  df.info()
- Statistical Details about Dataframe:
  df.describe()
- Finding Missing Values:
  df.isnull.sum()
- Changing Datatype of a Column:
  df['xyz']=df['xyz'].astype('float64')

COGNITENSOR
an AI COMPANY

Pandas Cheat Sheet



Data Wrangling with pandas Cheat Sheet http://pandas.pydata.org

Tidy Data – A foundation for wrangling in pandas

In a tidy data set: Each **variable** is saved in its own **column** & Each **observation** is saved in its own **row**

Tidy data complements pandas's vectorized operations. pandas will automatically preserve observations as you manipulate variables. No other format works as intuitively with pandas.

M * A

Syntax – Creating DataFrames

	a	b	c
1	4	7	10
2	5	8	11
3	6	9	12

```
df = pd.DataFrame(
          {"a" : [4 ,5, 6],
           "b" : [7, 8, 9],
           "c" : [10, 11, 12]},
        index = [1, 2, 3])
```
Specify values for each column.

```
df = pd.DataFrame(
     [[4, 7, 10],
      [5, 8, 11],
      [6, 9, 12]],
     index=[1, 2, 3],
     columns=['a', 'b', 'c'])
```
Specify values for each row.

		a	b	c
n	v			
d	1	4	7	10
	2	5	8	11
e	2	6	9	12

```
df = pd.DataFrame(
          {"a" : [4 ,5, 6],
           "b" : [7, 8, 9],
           "c" : [10, 11, 12]},
index = pd.MultiIndex.from_tuples(
          [('d',1),('d',2),('e',2)],
             names=['n','v']))
```
Create DataFrame with a MultiIndex

Method Chaining

Most pandas methods return a DataFrame so that another pandas method can be applied to the result. This improves readability of code.

```
df = (pd.melt(df)
        .rename(columns={
                'variable' : 'var',
                'value' : 'val'})
        .query('val >= 200')
     )
```

Reshaping Data – Change the layout of a data set

`pd.melt(df)`
Gather columns into rows.

`df.pivot(columns='var', values='val')`
Spread rows into columns.

`pd.concat([df1,df2])`
Append rows of DataFrames

`pd.concat([df1,df2], axis=1)`
Append columns of DataFrames

`df.sort_values('mpg')`
Order rows by values of a column (low to high).

`df.sort_values('mpg',ascending=False)`
Order rows by values of a column (high to low).

`df.rename(columns = {'y':'year'})`
Rename the columns of a DataFrame

`df.sort_index()`
Sort the index of a DataFrame

`df.reset_index()`
Reset index of DataFrame to row numbers, moving index to columns.

`df.drop(columns=['Length','Height'])`
Drop columns from DataFrame

Subset Observations (Rows)

`df[df.Length > 7]`
Extract rows that meet logical criteria.

`df.drop_duplicates()`
Remove duplicate rows (only considers columns).

`df.head(n)`
Select first n rows.

`df.tail(n)`
Select last n rows.

`df.sample(frac=0.5)`
Randomly select fraction of rows.

`df.sample(n=10)`
Randomly select n rows.

`df.iloc[10:20]`
Select rows by position.

`df.nlargest(n, 'value')`
Select and order top n entries.

`df.nsmallest(n, 'value')`
Select and order bottom n entries.

Logic in Python (and pandas)			
<	Less than	!=	Not equal to
>	Greater than	df.column.isin(*values*)	Group membership
==	Equals	pd.isnull(*obj*)	Is NaN
<=	Less than or equals	pd.notnull(*obj*)	Is not NaN
>=	Greater than or equals	&, |, ~, ^, df.any(), df.all()	Logical and, or, not, xor, any, all

Subset Variables (Columns)

`df[['width','length','species']]`
Select multiple columns with specific names.

`df['width']` *or* `df.width`
Select single column with specific name.

`df.filter(regex='regex')`
Select columns whose name matches regular expression *regex*.

regex (Regular Expressions) Examples	
'\.'	Matches strings containing a period '.'
'Length$'	Matches strings ending with word 'Length'
'^Sepal'	Matches strings beginning with the word 'Sepal'
'^x[1-5]$'	Matches strings beginning with 'x' and ending with 1,2,3,4,5
'^(?!Species$).*'	Matches strings except the string 'Species'

`df.loc[:,'x2':'x4']`
Select all columns between x2 and x4 (inclusive).

`df.iloc[:,[1,2,5]]`
Select columns in positions 1, 2 and 5 (first column is 0).

`df.loc[df['a'] > 10, ['a','c']]`
Select rows meeting logical condition, and only the specific columns .

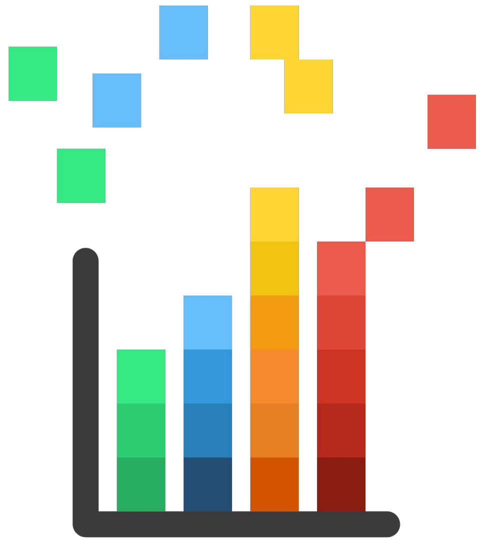
http://pandas.pydata.org/ This cheat sheet inspired by Rstudio Data Wrangling Cheatsheet (https://www.rstudio.com/wp-content/uploads/2015/02/data-wrangling-cheatsheet.pdf) Written by Irv Lustig, Princeton Consultants

For high quality resolution please click Here

Statistics

Statistics is a study collection, analysis, interpretation, presentation, and organization of data. In order to learn data science, one should focus on learning the basics first. But, it also doesn't mean that you have to mug up the formulas! There is no need to do that.

- You should focus on how a particular algorithm or function works!
- Basic Concepts that need to be studied are: Mean, median, mode, percentile, normal and standard normal distributions, standard deviation, correlations, interquartile range, Chebyshev's inequality(probability theory) and central limit theorem.
- (YouTube - Statistics in ML)



Visualization Libraries

- The most commonly used libraries for statistical analysis of data are Matplotlib and Seaborn.

- Matplotlib produces a variety of graphs like histogram(probability density function for feature selection), bar charts, scatterplot, pie charts, and so on. These graphs are research quality graphs as it uses vectors instead of pixels.

- Whereas, Seaborn also provides a wide variety of graphs like pair plot, counterplot, correlation matrix, and so on.

Exploratory Data Analysis (EDA)

It is an essential step before implementing an algorithm on the data. Explore your dataset as much as possible and find the number of categorical, continuous, and discrete variables.

This help uncover the structure of data and will answer questions like:

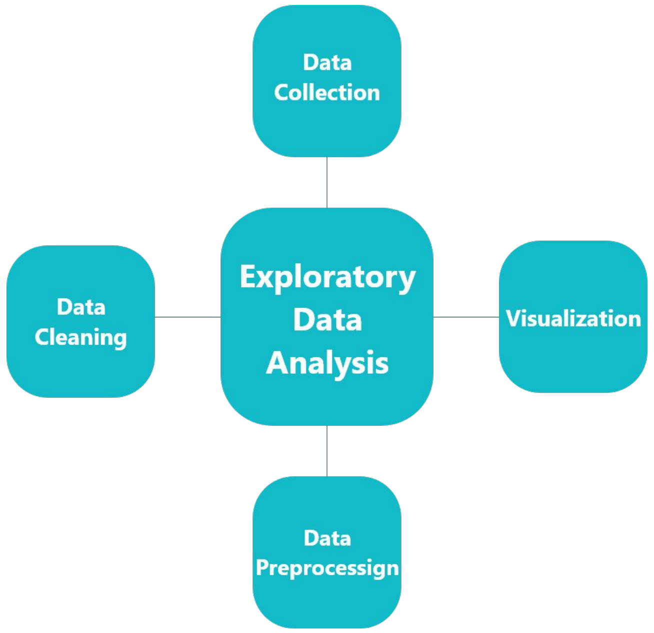
- Which variables suggest interesting relationships?
- Which observations are unusual?

This will give you a better understanding of your dataset and it will help you decide what your dataset needs!

Here is an article for pre-processing using scikit learn library.

Here is a useful link for Data Handling and pre-processing using Pandas library.

After the analysis of data, write down your observations for future references.



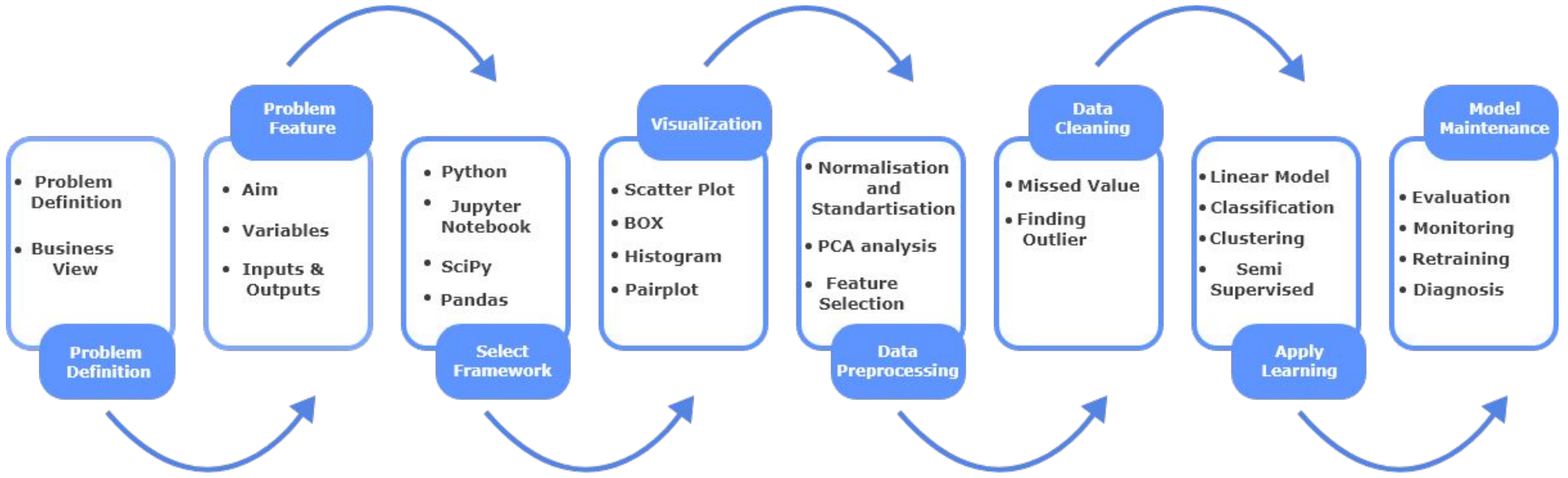
Machine Learning Algorithms

- The main step before implementing the algorithm is understanding the Math behind that particular algorithm(study Statistics well!).
- Learn at least 15-20 commonly used algorithms
- Study about Hyperparameter Tuning (responsible for controlling the overall behavior of the machine learning model)
- Grid search vs. Randomized search
- Regression Models: These models are used to predict continuous values. Some of these models are Linear, Rasso, Ridge, Logistic, Decision Tree Regression and Classifier, Random Forest Regression and Classifier, AdaBoost Regression and Classifier, XGBoost Regression and Classifier.

Machine Learning Algorithms

- 7 steps of Machine Learning:-

1. Data Collection
2. Data Preparation
3. Choose a Model
4. Train the model
5. Evaluate the Model
6. Parameter Tuning
7. Make Predictions



You can find a stepwise implementation of 20+ algorithms on this Python Notebook link, along with some other useful resources.

Deployment of Model

Python is known for its objective of deployment and production. Deployment is necessary as well as challenging. This process can be made easier by encapsulating the machine learning model behind APIs(Application Programming Interface), as it will help in establishing the connection of models with other applications.

Following are some of the platforms used to deploy model:

- **Microsoft Azure**
- **Heroku**
- **PaaS(Platform as service)**
- **Google Cloud**
- **AWS Cloud**

Deep Learning

- Deep Learning vs Machine Learning (Where to use what?)
- Learn about Neural Networks and its working in-depth
- Types of Neural Networks, importance and their pros & cons: ANN, CNN(image processing, image classification, object detection, transfer learning), RNN(time sequences, text-based)
- Feature Engineering (makes the algorithm more efficient)
- Commonly used libraries in deep learning are: Tensorflow, Theano, Keras, PyTorch.

Link for more details How to Improve Performance With Transfer Learning for Deep Learning Neural Networks

COGNITENSOR an AI COMPANY

Deep Learning

- Deep Learning in OpenCV : Deep Learning is the most popular and the fastest growing area in Computer Vision nowadays. Since OpenCV 3.1 there is DNN module in the library that implements forward pass (inferencing) with deep networks, pre-trained using some popular deep learning frameworks, such as Caffe. In OpenCV 3.3 the module has been promoted from opencv_contrib repository to the main repository (https://github.com/opencv/opencv/tree/master/modules/dnn) and has been accelerated significantly. The module has no any extra dependencies, except for libprotobuf, and libprotobuf is now included into OpenCV. The supported frameworks- Caffe ,Tensorflow ,Torch,Darknet. The supported layers: AbsVal,Average Pooling, Batch Normalization etc.

- Transfer learning refers to a technique for predictive modeling on a different but somehow similar problem that can then be reused partly or wholly to accelerate the training and improve the performance of a model on the problem of interest. In deep learning, this means reusing the weights in one or more layers from a pre-trained network model in a new model and either keeping the weights fixed, fine tuning them, or adapting the weights entirely when training the model.

Link for more details How to Improve Performance With Transfer Learning for Deep Learning Neural Networks

Frameworks for Python

To create websites and request APIs, we have the following commonly used frameworks in Python:

- Django (full-stack web framework, used for the development of large and complex web applications).
- Flask (light-weight framework with minimal features, used for the development of simple web applications).
- TensorFlow (has pre-written codes for the most complicated deep learning models, including Recurrent Neural Networks and Convolutional Neural Networks).
- Keras (high-level neural network API , developed to learn and prototype simple concepts and understand models).
- Pytorch (deep learning framework, supports data parallelism and distributed learning model and is ideal for small projects and prototyping).

COGNITENSOR
an AI COMPANY

Databases

The database helps to organize, manage and edit large chunks of data. Some of the databases are:

- MongoDB: NoSql database, processes structured, semi-structured(JSON format) and unstructured data (audio, video files). It represents data in JSON documents.

- MySql: It is based on SQL. It represents data in rows and columns. It is used for structured data only. Less risky than MongoDB.

(SQL for Data Science - audit this course for free)

My SQL	**MongoDB**
Table	**Collection**
Row	**Document**
Column	**Field**
Joins	**Embedded documents, Linking**

Web Scraping and APIs

- One of the most important things in the field of Data Science is the skill of getting the right data for the problem you want to solve. Data Scientists don’t always have a prepared database to work on but rather have to pull data from the right sources. For this purpose, APIs and Web Scraping are used.

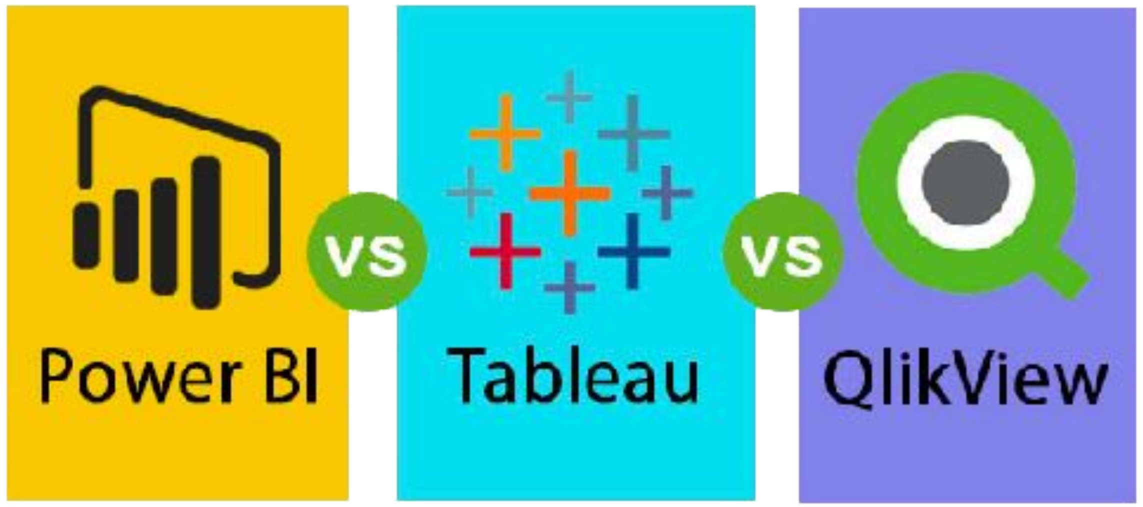
- API (Application Program Interface): An API is a set of methods and tools that allows us to query and retrieve data dynamically. Reddit, Spotify, Twitter, Facebook, and many other companies provide free APIs that enable developers to access the information they store on their servers; others charge for access to their APIs.

- Web Scraping: A lot of data isn't accessible through data sets or APIs but rather exist on the Internet as Web pages. So, through Web Scraping, we can access the data without waiting for the provider to create an API.

Some tools used for scraping are: Beautiful Soup, Selenium, Scrapy, etc.

Visualization Tools

- Power BI: Easy to use the platform, supports R language, and compatible with Microsoft Azure.
- Tableau: Great visualization capabilities, supports both R and Python languages, compatible with Azure, AWS, and so on.
- Qlik Sense: It is a self-service analytics tool that is compatible with both R and Python languages and compatible with the SaaS cloud (Software-as-a-Service). It provides storage capability of about 500GB.



Competitive Data Science

Data Science is all about deriving insights from the data. By developing the competitive aptitude for data science, a person can efficiently extract useful information and present it in a lucid form. For example, consider these questions:-

- What libraries or plots to use for visualization?
- What ML model to use?
- What features are more important than others and how they relate to each other?
- How to deal with unknown values, such as NaN or other filler values(it can be intentional)?
- How do the outliers affect my data analysis?

As you can imagine, the answers to these questions are not straightforward and entirely depends on the data set you are working with. As in any other field, this comes with experience and the most efficient way to gain this experience is by participating in data science competitions. This is similar to competitive coding and after solving lots of questions you will finally understand what data structure and algorithm to use in a given problem statement. You can also view solutions of top performers to understand how they approach a given problem.

- Here is a list of top data science competitions.
- Kaggle
- International Data Analysis Olympiad (IDAHO)
- DrivenData
- DataHack & DSAT

Essential Newsletters for Data Scientists

Newsletters are an excellent form of content curation that can help deliver interesting and insightful information to you with minimal effort on your part. Staying on top of the latest news in the ever-changing realm of data science is highly essential for any aspiring data scientist.

These newsletters can help in learning techniques and technologies, introduce new concepts and learning resources that one wouldn't have known about otherwise and even notify of networking opportunities and public tech talks. Some of the most popular newsletters are as follows:

- Data Science Weekly:The newsletter started in 2013 and has pumped out consistent issues since. It starts off with an Editor Picks section and quickly moves onto listing a bunch of data science articles and videos. Furthermore, it includes a section for job openings, tutorials, and books as well. Link: https://www.datascienceweekly.org/

Essential Newsletters for Data Scientists

- O’Reilly Data Newsletter**:** You have probably heard of O’Reilly Media in one way or another. They also publish ebooks, host conferences, and offer other learning solutions. Their data-focused newsletter delivers 10 links each week that range from news to tutorials to white-papers. Link: https://www.oreilly.com/data/newsletter.html

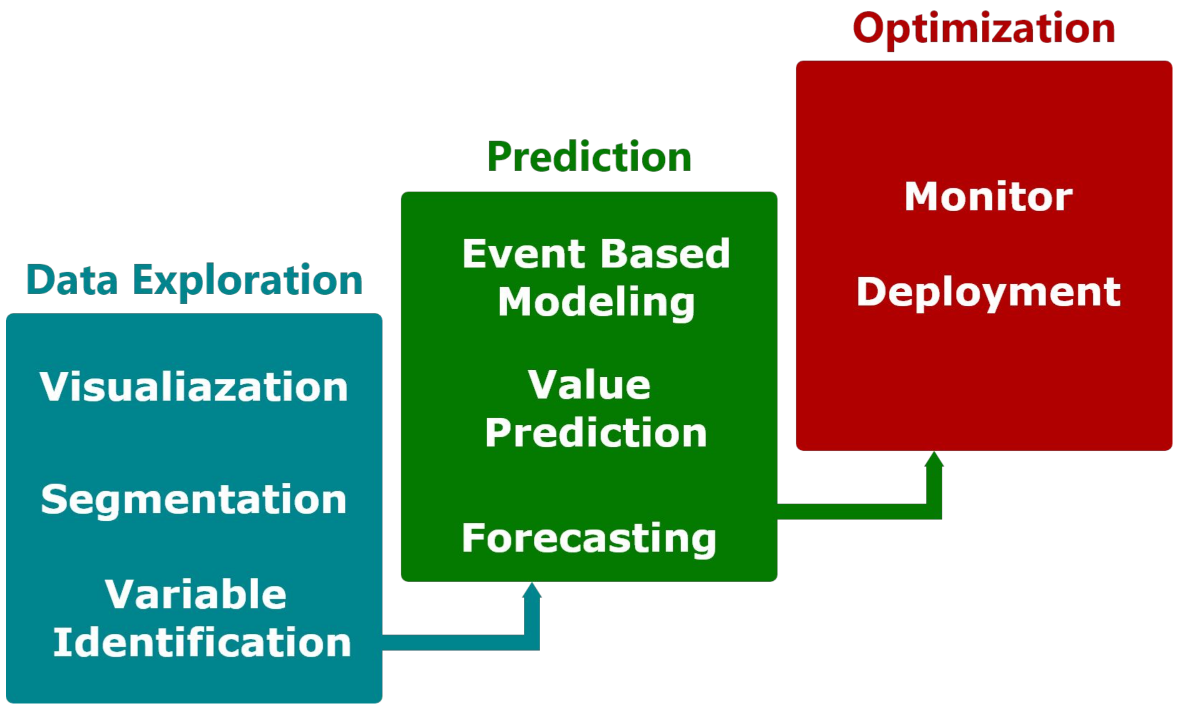
- Data Elixir: Data Elixir takes a similar approach, breaking things down into a wide-ranging collection of weekly news, insights, tools & techniques, resources, and data visualization. The newsletter goes out to over 30,000 subscribers and is delivered every Tuesday. Link: https://dataelixir.com/

- Data Machina: Data Machina is a more technical newsletter that breaks down links by technology, hitting on topics from R to blockchain to algorithms. There’s really a little bit of everything here. Link: https://datamachina.substack.com/

Stages to Solve Analytical Problems:

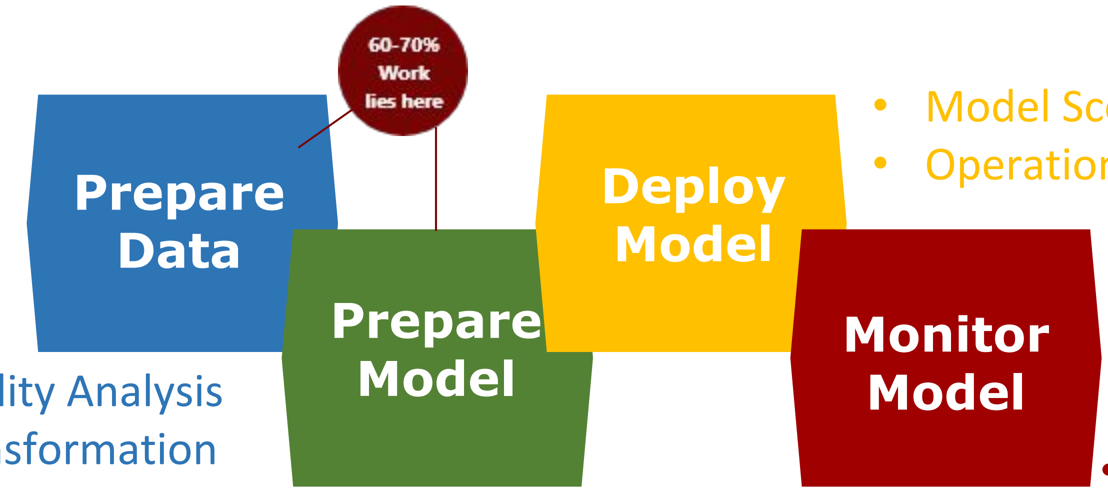
If we are not able to understand how a business makes money, we can never solve their problems using any AI or Analytical techniques.

Pie charts are not best practices to use for visualization. Bar charts are also not recommended in visualization as width of each bar don’t not signify / communicate any message.



Stages to Solve Analytical Problems: (contd.)

- <u>Steps in Data Mining</u>



- Model Scoring
- Operational Deployment

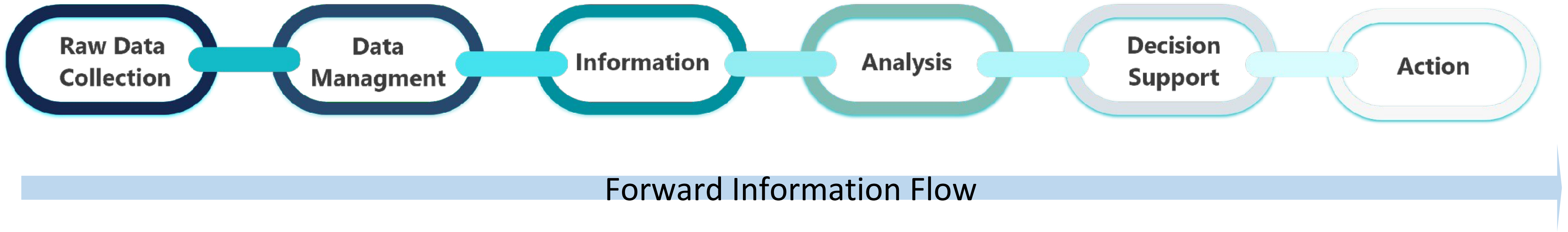
- Data Quality Analysis
- Data Transformation

- Variable Screening
- Unsupervised Learning like customer segmentation
- Supervised Learning like Predictive Segmentation
- SEMMA <u>Process</u>: Sample, Explore Modify(imputing,missing values), Model,Assess

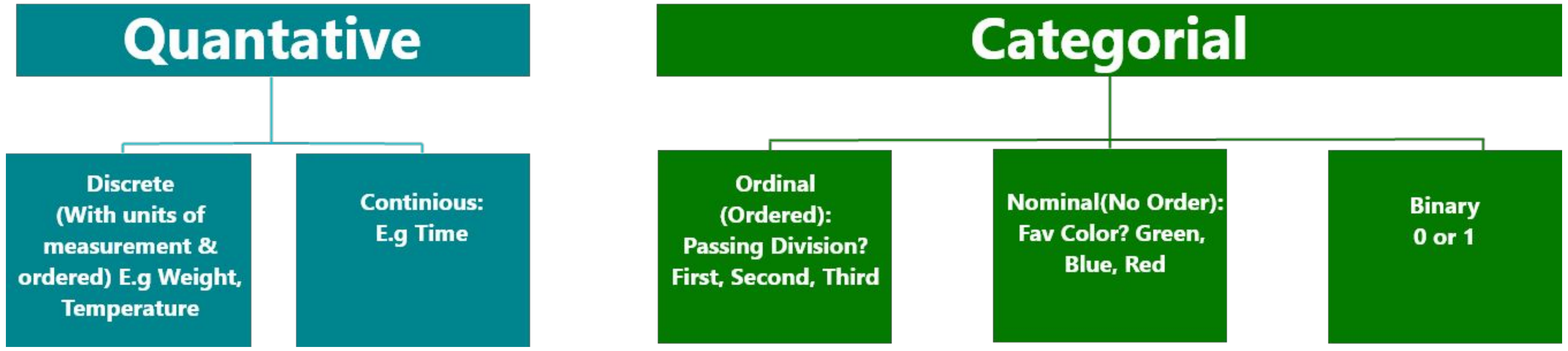
- Model Validation
- Expected life of model
- Benchmarking Model
- Rebuild Model?- Use new data and new variables
- Refresh Model?-Keep current variables and return model on new data

Stages to Solve Analytical Problems: (contd.)

- Information Supply Chain



- Types of variables:


Quantative
Discrete (With units of measurement & ordered) E.g Weight, Temperature
Continious: E.g Time
Categorial
Ordinal (Ordered): Passing Division? First, Second, Third
Nominal(No Order): Fav Color? Green, Blue, Red
Binary 0 or 1

COGNITENSOR
an AI COMPANY

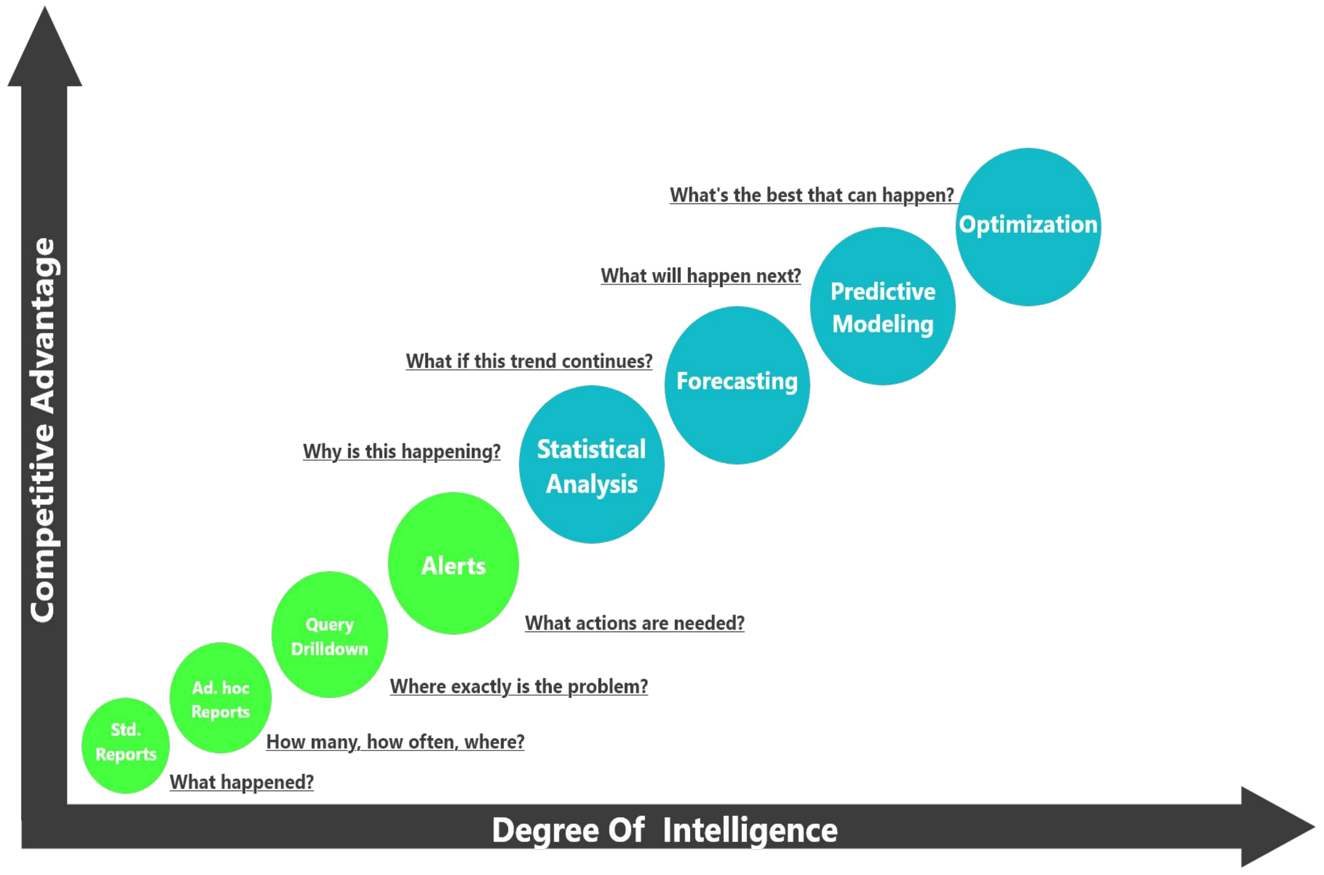
Intelligence through analytics:

The image depicts a linearly increasing graph between Competitive Advantage a firms gains vs The Degree of Intelligence (AI) implemented by them.

There are 8 stages, through each stage as an organization's AI implementation increases, the more competitive advantage it gains over others in the market.

AI perse is not a new term. It dates back to the 1990's, the closer to proper implementation is being done in today's market.

- The Best Artificial Intelligence, Machine Learning and Data Science Resources*



Resources

1. Free Resources

- https://www.coursera.org/learn/machine-learning (Andrew Ng)
- https://www.youtube.com/watch?v=7S865QCGL74&list=PLZoTAELRMXVPBTrWtJkn3wWQxZkmTXGwe

  (Krish Naik)
- https://towardsdatascience.com/ (Articles)
- https://leetcode.com/explore/featured/card/machine-learning-101/ (Leetcode)
- https://www.youtube.com/watch?v=edvg4eHi_Mw (Great Learning)
- https://www.edx.org/course/subject/data-science

2. Paid Resources

https://www.udemy.com/course/machinelearning/ (Udemy)

- https://www.udemy.com/course/data-science-and-machine-learning-with-python-hands-on/?LSNPUBID=SAyYsTvLiGQ&ranEAID=SAyYsTvLiGQ&ranMID=39197&ranSiteID=SAyYsTvLiGQ-XS5RE4YTyCIhy9dT6HGo2g (Udemy)
- https://www.codingelements.com/course/machine-learning-and-deep-learning-online/ (Coding Elements)

3. Practicing your skills:

- https://www.kaggle.com/#

COGNITENSOR
an AI COMPANY

THANK YOU