

[Open in app](#)[Get started](#)

Anushka Bajpai

[Follow](#)

Mar 21 · 5 min read

[Listen](#)[Save](#)

101 DATA SCIENCE with Cheat Sheets (ML, DL, Scraping, Python, R, SQL, Maths & Statistics)

“Perhaps the best test of a man’s intelligence is his capacity for making a summary” — Lytton Strachey

Couldn’t find a better topic to begin my new ‘Medium’ journey with . . .





Open in app

Get started

Photo by [Annie Spratt](#) on [Unsplash](#)

Data Science is an ever-growing field, there are numerous tools & techniques to remember. It is not



[Open in app](#)[Get started](#)

There are thousands of cheat sheets available out there. However, how many of them do you really find useful ?

It's a treasure to have good quality resources compiled at one place. Hence, I decided to go ahead and put in this compilation of Cheat Sheets for a quick yet thorough reference.

For my readers' convenience, I have segregated the best cheat sheets (comprehensive yet concise) separately for each of the below topics . . .

1. **Python** (Basic python as well as ML libraries : Numpy /Pandas/Matplotlib/Seaborn/Spacy)
2. **R**
3. **SQL**
4. **Web scraping Tools** (Beautifulsoup, Scrapy and Selenium)
5. **Data Visualization Tools** (Tableau, Power BI)
6. **Maths and Statistics**
7. **Machine Learning Algorithms** (Supervised and Unsupervised)
8. **Deep Learning and NLP** (Neural Networks, Keras, Tensorflow, Pytorch, NLTK, Spacy)
9. **IDE/OS** (GitHub, Linux, SkLearn, Jupyter Notebook)

PYTHON





Open in app

Get started

Python 3 Cheat Sheet

<p>Base Types</p> <pre>integer, float, boolean, string, bytes int 783 0 -192 0b010 0o642 0xF3 float 9.23 0.0 -1.7e-6 bool True False str "One\nTwo" escaped new line 'I\n' escaped ' bytes b"toto\xef\x7f" hexadecimal octal ↪ immutables</pre>	<p>Container Types</p> <ul style="list-style-type: none"> * ordered sequences, fast index access, repeatable values <table border="0" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 30%;">list [1, 5, 9]</td> <td style="width: 30%;">["x", 11, 8.9]</td> <td style="width: 30%;">{"mot": "}"</td> </tr> <tr> <td>tuple (1, 5, 9)</td> <td>11, "y", 7.4</td> <td>("mot",)</td> </tr> </table> * key containers, no a priori order, fast key access, each key is unique <table border="0" style="width: 100%; border-collapse: collapse;"> <tr> <td>dict {"key": "value"} ↪ expression with only commas → tuple</td> <td>dict (a=3, b=4, k="v") ↪ empty</td> </tr> <tr> <td>collection set {"key1", "key2"} ↪ keys=hashable values (base types, immutables...)</td> <td>set () ↪ empty</td> </tr> </table> 	list [1, 5, 9]	["x", 11, 8.9]	{"mot": "}"	tuple (1, 5, 9)	11, "y", 7.4	("mot",)	dict {"key": "value"} ↪ expression with only commas → tuple	dict (a=3, b=4, k="v") ↪ empty	collection set {"key1", "key2"} ↪ keys=hashable values (base types, immutables...)	set () ↪ empty	<p>Identifiers</p> <ul style="list-style-type: none"> for variables, functions, modules, classes... names a-zA_Z_ followed by a-zA_Z_0_9 diacritics allowed but should be avoided language keywords forbidden lower/UPPER case discrimination • a toto x7 y_max BigOne • byt and doc <p>Variables assignment</p> <ul style="list-style-type: none"> assignment ↪ binding of a name with a value 1) evaluation of right side expression value 2) assignment in order with left side names x=1.2+8+sin(y) a=b=c=0 assignment to same value y, z, x=9. 2, -7.6, 0 multiple assignments a,b,b,a values swap a, *b=seq] unpacking of sequence in item and list *a,b=seq } item and list x+=3 increment ↪ x=x+3 x-=2 decrement ↪ x=x-2 x=None undefined = constant value del x remove name x 																												
list [1, 5, 9]	["x", 11, 8.9]	{"mot": "}"																																						
tuple (1, 5, 9)	11, "y", 7.4	("mot",)																																						
dict {"key": "value"} ↪ expression with only commas → tuple	dict (a=3, b=4, k="v") ↪ empty																																							
collection set {"key1", "key2"} ↪ keys=hashable values (base types, immutables...)	set () ↪ empty																																							
<p>Identifiers</p> <ul style="list-style-type: none"> for variables, functions, modules, classes... names a-zA_Z_ followed by a-zA_Z_0_9 diacritics allowed but should be avoided language keywords forbidden lower/UPPER case discrimination • a toto x7 y_max BigOne • byt and doc <p>Variables assignment</p> <ul style="list-style-type: none"> assignment ↪ binding of a name with a value 1) evaluation of right side expression value 2) assignment in order with left side names x=1.2+8+sin(y) a=b=c=0 assignment to same value y, z, x=9. 2, -7.6, 0 multiple assignments a,b,b,a values swap a, *b=seq] unpacking of sequence in item and list *a,b=seq } item and list x+=3 increment ↪ x=x+3 x-=2 decrement ↪ x=x-2 x=None undefined = constant value del x remove name x 	<p>type (expression)</p> <ul style="list-style-type: none"> can specify integer number base in 2nd parameter int("15") → 15 int("3f", 16) → 63 int(15.56) → 15 float("-11.24e8") → -1124000000.0 round(15.56, 1) → 15.6 rounding to 1 decimal (0 decimal → integer number) bool(x) False for null x, empty container x, None or False x: True for other x str(x) → "... representation string of x for display (cf. formatting on the back) chr(64) → '@' ord('@') → 64 code ↔ char repr(x) → "..." literal representation string of x bytes([72, 9, 64]) → b'H\x09' list("abc") → ['a', 'b', 'c'] dict([(3, "three"), (1, "one")]) → {1: 'one', 3: 'three'} set(["one", "two"]) → {'one', 'two'} separator str and sequence of str → assembled str '.'.join(['toto', '12', 'pswd']) → 'toto:12:pswd' <p>Conversions</p> <ul style="list-style-type: none"> str splitted on whitespaces → list of str "words with spaces".split() → ['words', 'with', 'spaces'] str splitted on separator str → list of str "1, 4, 8, 2".split(",") → ['1', '4', '8', '2'] sequence of one type → list of another type (via list comprehension) [int(x) for x in ('1', '29', '-3')] → [1, 29, -3] 	<p>Identifiers</p> <ul style="list-style-type: none"> for variables, functions, modules, classes... names a-zA_Z_ followed by a-zA_Z_0_9 diacritics allowed but should be avoided language keywords forbidden lower/UPPER case discrimination • a toto x7 y_max BigOne • byt and doc <p>Variables assignment</p> <ul style="list-style-type: none"> assignment ↪ binding of a name with a value 1) evaluation of right side expression value 2) assignment in order with left side names x=1.2+8+sin(y) a=b=c=0 assignment to same value y, z, x=9. 2, -7.6, 0 multiple assignments a,b,b,a values swap a, *b=seq] unpacking of sequence in item and list *a,b=seq } item and list x+=3 increment ↪ x=x+3 x-=2 decrement ↪ x=x-2 x=None undefined = constant value del x remove name x <p>type (expression)</p> <ul style="list-style-type: none"> can specify integer number base in 2nd parameter int("15") → 15 int("3f", 16) → 63 int(15.56) → 15 float("-11.24e8") → -1124000000.0 round(15.56, 1) → 15.6 rounding to 1 decimal (0 decimal → integer number) bool(x) False for null x, empty container x, None or False x: True for other x str(x) → "... representation string of x for display (cf. formatting on the back) chr(64) → '@' ord('@') → 64 code ↔ char repr(x) → "..." literal representation string of x bytes([72, 9, 64]) → b'H\x09' list("abc") → ['a', 'b', 'c'] dict([(3, "three"), (1, "one")]) → {1: 'one', 3: 'three'} set(["one", "two"]) → {'one', 'two'} separator str and sequence of str → assembled str '.'.join(['toto', '12', 'pswd']) → 'toto:12:pswd' <p>Conversions</p> <ul style="list-style-type: none"> str splitted on whitespaces → list of str "words with spaces".split() → ['words', 'with', 'spaces'] str splitted on separator str → list of str "1, 4, 8, 2".split(",") → ['1', '4', '8', '2'] sequence of one type → list of another type (via list comprehension) [int(x) for x in ('1', '29', '-3')] → [1, 29, -3] 																																						
<p>for lists, tuples, strings, bytes...</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 30%; padding: 5px;">negative index</td> <td style="width: 30%; padding: 5px;">-5</td> <td style="width: 30%; padding: 5px;">-4</td> <td style="width: 30%; padding: 5px;">-3</td> <td style="width: 30%; padding: 5px;">-2</td> <td style="width: 30%; padding: 5px;">-1</td> </tr> <tr> <td>positive index</td> <td>0</td> <td>1</td> <td>2</td> <td>3</td> <td>4</td> </tr> <tr> <td>lst=[10, 20, 30, 40, 50]</td> <td>10</td> <td>20</td> <td>30</td> <td>40</td> <td>50</td> </tr> <tr> <td>positive slice</td> <td>0</td> <td>1</td> <td>2</td> <td>3</td> <td>4</td> </tr> <tr> <td>negative slice</td> <td>-5</td> <td>-4</td> <td>-3</td> <td>-2</td> <td>-1</td> </tr> </table> <p>Access to sub-sequences via lst [start slice : end slice : step]</p> <pre>lst[:-1]→[10, 20, 30, 40] lst[:-1]→[50, 40, 30, 20, 10] lst[1:-3]→[20, 30] lst[:3]→[10, 20, 30] lst[1:-1]→[20, 30, 40] lst[::2]→[50, 30, 10] lst[-3:-1]→[30, 40] lst[3:]→[40, 50] lst[::2]→[10, 30, 50] lst[:]→[10, 20, 30, 40, 50] shallow copy of sequence</pre> <p>Missing slice indication → from start / up to end.</p> <p>On mutable sequences (list), remove with del lst[3:5] and modify with assignment lst[1:4]=[15, 25]</p>	negative index	-5	-4	-3	-2	-1	positive index	0	1	2	3	4	lst=[10, 20, 30, 40, 50]	10	20	30	40	50	positive slice	0	1	2	3	4	negative slice	-5	-4	-3	-2	-1	<p>Sequence Containers Indexing</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 30%; padding: 5px;">Items count</td> <td style="width: 30%; padding: 5px;">len(lst)→5</td> <td style="width: 30%; padding: 5px;">Individual access to items via lst [index]</td> </tr> <tr> <td>lst[0]→10</td> <td>= first one</td> <td>lst[1]→20</td> </tr> <tr> <td>lst[-1]→50</td> <td>= last one</td> <td>lst[-2]→40</td> </tr> </table> <p>On mutable sequences (list), remove with del lst[3] and modify with assignment lst[4]=25</p>	Items count	len(lst)→5	Individual access to items via lst [index]	lst[0]→10	= first one	lst[1]→20	lst[-1]→50	= last one	lst[-2]→40
negative index	-5	-4	-3	-2	-1																																			
positive index	0	1	2	3	4																																			
lst=[10, 20, 30, 40, 50]	10	20	30	40	50																																			
positive slice	0	1	2	3	4																																			
negative slice	-5	-4	-3	-2	-1																																			
Items count	len(lst)→5	Individual access to items via lst [index]																																						
lst[0]→10	= first one	lst[1]→20																																						
lst[-1]→50	= last one	lst[-2]→40																																						
<p>Boolean Logic</p> <ul style="list-style-type: none"> Comparisons: < > <= >= == != (boolean results) $\leq \geq \leq = \neq$ a and b logical and both simultaneously true a or b logical or one or other or both pitfall: and and or return value of a or of b (under short-circuit evaluation). → ensure that a and b are booleans. not a logical not True False True and False constants <p>floating numbers... approximated values</p> <p>Operators: + - * / // % **</p> <p>Priority (→) $\times \div \uparrow \uparrow \uparrow \uparrow$</p> <p>Priority (→) $\uparrow \uparrow \uparrow \uparrow$</p> <p>$\oplus$ → matrix \otimes \oplus \otimes</p> <p>(1+5.3)*2→12.6</p> <p>abs(-3.2)→3.2</p> <p>round(3.57, 1)→3.6</p> <p>pow(4, 3)→64.0</p> <p>↑ usual order of operations</p>	<p>Statements Blocks</p> <pre>parent statement: statement block 1... : parent statement: statement block2... : next statement after block 1</pre> <p>Configure editor to insert 4 spaces in place of an indentation tab.</p> <p>Maths</p> <p>angles in radians</p> <pre>from math import sin, pi sin(pi/4)→0.707... cos(2*pi/3)→-0.4999... sqrt(81)→9.0 log(e**2)→2.0 ceil(12.5)→13 floor(12.5)→12</pre> <p>modules math, statistics, random, decimal, fractions, numpy, etc (cf. doc)</p>	<p>Modules/Names Imports</p> <pre>module trucfile truc.py from monmod import nom1, nom2 as fact import monmod → direct access to names, renaming with as import monmod → access via monmod.nom1... → modules and packages searched in python path (cf. sys.path)</pre> <p>Conditional Statement</p> <p>statement block executed only if a condition is true</p> <pre>if logical condition: statements block</pre> <p>Can go with several elif, else... and only one final else. Only the block of first true condition is executed.</p> <p>with a var x:</p> <pre>if bool(x)==True: → if x: if bool(x)==False: → if not x: state="Kid" state="Retired" state="Active"</pre> <p>Exceptions on Errors</p> <p>Signaling an error:</p> <pre>raise ErrClass...</pre> <p>Errors processing:</p> <pre>try: normal processing block</pre> <p>except Exception as e:</p> <pre>error processing block</pre> <p>finally block for final processing in all cases</p>																																						

Source :<https://www.reddit.com/>

Basic Python

1. gto76 — extensive Python cheat sheet offered by GitHub



[Open in app](#)[Get started](#)

4. Cheatography

5. Python crash course

6. DataQuest (basic and intermediate)

7. MementoPython3

Numpy

1. DataCamp

2. Intellipat

3. A Little Bit of Everything

4. Data Quest

5. NumPy for R (and S-plus) Users

6. Ipgp_github

Pandas

1. The Most Comprehensive Cheat Sheet

2. BecomingHuman.AI

3. The Beginner's Cheat Sheet

4. DATACAMP

5. Data Science Central (compact)

6. INTELLIPAT



[Open in app](#)[Get started](#)

4. [Becominghuman.ai](#)

5. [Travis_Blog](#)

Seaborn

1. [Datacamp](#)

2. [Cheatography](#)

3. [MartinNormark](#)

4. [kaggle](#)





Open in app

Get started

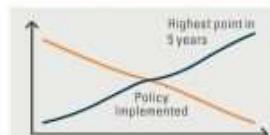
Core Principles of Data Visualization

Audience



Always consider your audience—whether they need a short, written report, a more in-depth paper, or an online exploratory data tool.

Include annotation



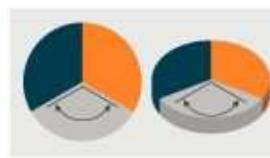
Add explanatory text to help the reader understand how to read or use the visualization (if necessary) and also to guide them through the content.

Use pie charts with care



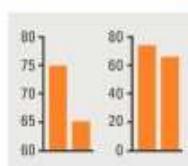
We are not very good at discerning quantities from the slices of the pie chart. Other chart types—for example, bars, stacked bars, treemaps, or slope charts—may be a better choice.

Avoid 3D



Using 3D when you don't have a third variable will usually distort the perception of the data and should thus be avoided.

Start bar and column charts at zero



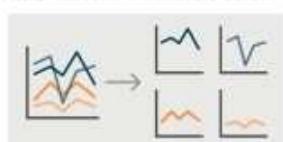
Bar and column charts that do not start at zero overemphasize the differences between the values. For small changes in quantities, consider visualizing the difference or the change in the values.

Make labels easy to read



When applicable, rotate bar and column charts to make the labels horizontal. If possible, make vertical axis labels horizontal, possibly below the title. In general, make labels clear, concise, and easy for your reader to understand.

Try small multiples



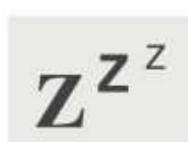
Breaking up a complicated chart into smaller chunks can be an effective way to visualize your data.

Use maps carefully

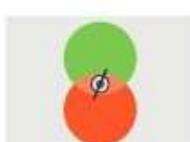


Use maps carefully, always being sure it is the geographic point you are trying to make. Column and bar charts, for example, are often better at enabling comparisons between geographic units.

Color and font considerations



Avoid default colors and fonts—they all look the same and don't stand out.

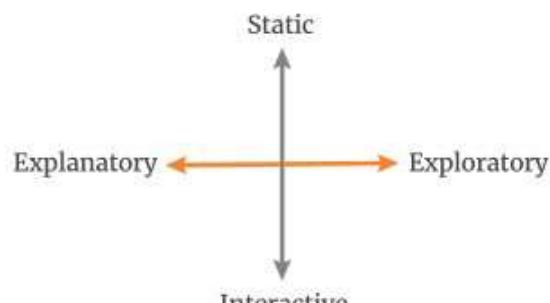


Consider color blindness—about 10% of people (mostly men) have some form of color blindness.



Avoid the rainbow color palette—it doesn't map to our number system and there is no logical ordering.

Visualization Mapping: Form and Function



PolicyViz

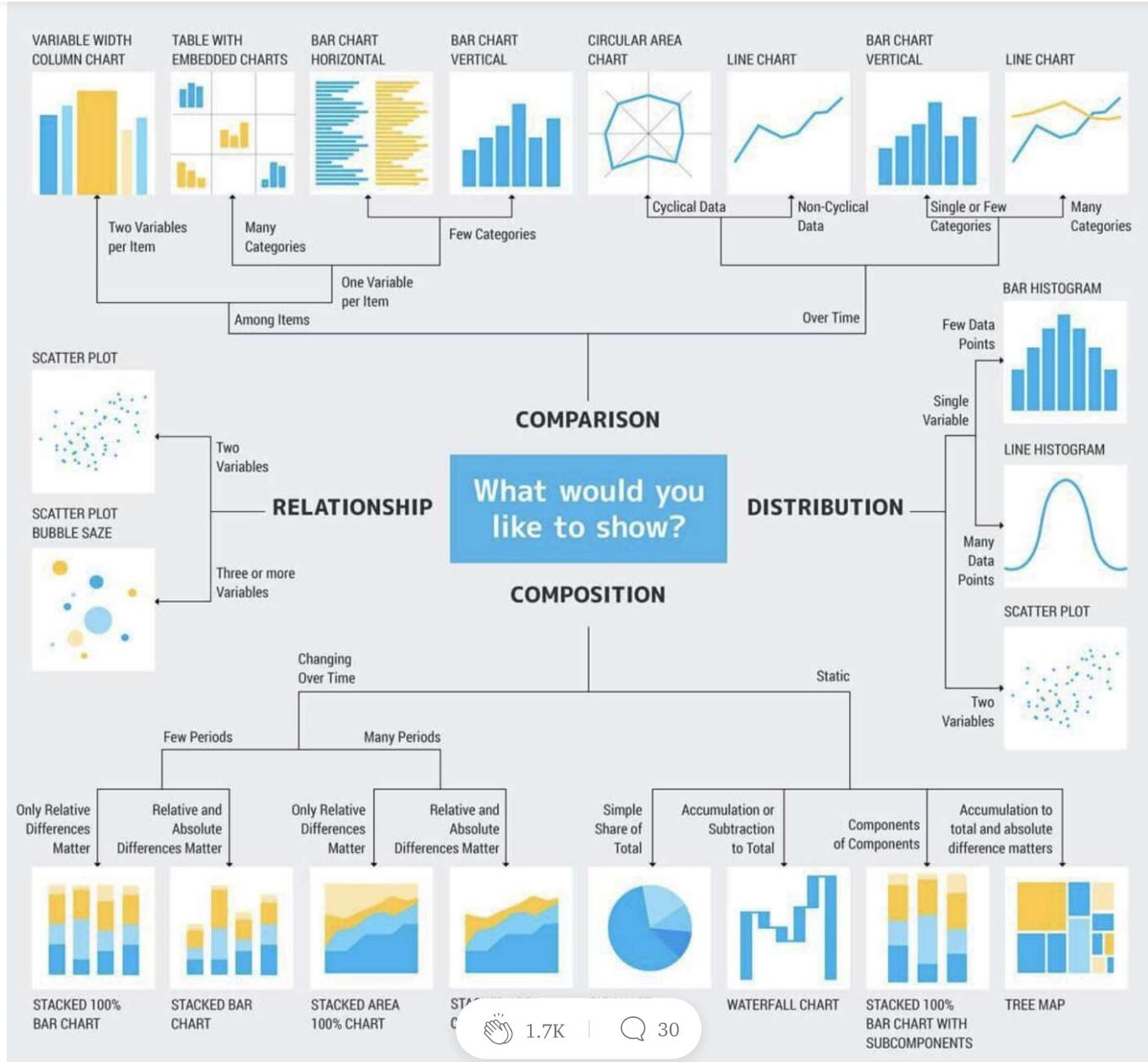
Source : <https://www.kdnuggets.com/2018/08/data-visualization-cheatsheet.html>





Open in app

Get started

Source : <https://www.kaggle.com/getting-started/160583>



Open in app

Get started

Matplotlib is a Python 2D plotting library which produces publication-quality figures in a variety of hardcopy formats and interactive environments across platforms.

Matplotlib Cheat Sheet

BecomingHuman.AI

Source : <https://becominghuman.ai/>

R

1. [github](#)2. [datacamp](#)3. [rstudio](#)4. [intellipat](#)



Open in app

Get started

R Programming Cheat Sheet

JUST THE BASICS

CREATED BY: ARIANNE COLTON AND SEAN CHEN

GENERAL

- R version 3.0 and greater adds support for 64 bit integers
- R is case sensitive
- R index starts from 1

HELP

`help(functionName) OR ?functionName`

Help Home Page	<code>help.start()</code>
Special Character Help	<code>help('?')</code>
Search Help	<code>help.search(..) or ?..</code>
Search Function - with Partial Name	<code>apropos('mea')</code>
See Example(s)	<code>example(topic)</code>

OBJECTS in current environment

Display Object Name	<code>objects() or ls()</code>
Remove Object	<code>rm(object1, object2, ..)</code>

Notes:

- name starting with a period are accessible but invisible, so they will not be found by 'ls'
- To guarantee memory removal, use 'gc', releasing unused memory to the OS. R performs automatic 'gc' periodically

SYMBOL NAME ENVIRONMENT

- If multiple packages use the same function name the function loaded the last will get called.
- To avoid this precede the function with the name of the package e.g. `packageName::functionName(..)`

LIBRARY

Only trust reliable R packages i.e., 'ggplot2' for plotting, 'sp' for dealing spatial data, 'reshape2', 'survival', etc.

Load Package	<code>library(packageName) or require(packageName)</code>
Unload Package	<code>detach(packageName)</code>

Note: `require()` returns the status(True/False)

MANIPULATING STRINGS

```

Putting Together Strings
paste('string1', 'string2', sep = '/')
# separator ('sep') is a space by default
paste(c('1', '2'), collapse = '/')
# returns '1/2'

Split String
stringr::str_split(string = v1,
start = 1, end = 3)
# returns a list

Get SubString
stringr::str_sub(string = v1,
start = 1, end = 3)
isJohnFound <- stringr::str_detect(string = df1[,1],
pattern = ignore.case('John'))
# returns True/False if John was found

Match String
# returns True/False if John was found
df1[isJohnFound, c('col1',
...)]
  
```

DATA TYPES

Check data type: `class(variable)`

FOUR BASIC DATA TYPES

- Numeric** - includes float/double, int, etc.
`is.numeric(variable)`
- Character(string)**
`nchar(variable) # length of a character or numeric`
- Date/POSIXct**
 - Date**: stores just a date. In numeric form, number of days since 1/1/1970 (see below)
`date1 <- as.Date('2012-06-28'), as.numeric(date1)`
 - POSIXct** stores a date and time. In numeric form, number of seconds since 1/1/1970.
`date2 <- as.POSIXct('2012-06-28 18:00')`

Note: Use 'lubridate' and 'chron' packages to work with Dates
- Logical**
 - `(TRUE = 1, FALSE = 0)`
 - Use `==/!=` to test equality and inequality
`as.logical(TRUE) == 1`

Source : <https://intellipaat.com/>

SQL

1. SQL Detailed

2. LearnSQL

3. SQL Tutorial

4. SQL Joins

5. Intellipaat

6. BuggyProgrammer

DATA STRUCTURES

VECTOR

- Group of elements of the SAME type
- R is a vectorized language, operations are applied to each element of the vector automatically
- R has no concept of column vectors or row vectors
- Special vectors: letters and LETTERS, that contain lower-case and upper-case letters

```

Create Vector v1 <- c(1, 2, 3)
Get Length length(v1)
Check If All or Any is True all(v1); any(v1)
Integer Indexing v1[1:3]; v1[c(1, 6)]
Boolean Indexing v1[is.na(v1)] <- 0
Naming c(first = 'a', ..) or names(v1) <- c('first', ..)
  
```

FACTOR

- `as.factor(v1)` gets you the levels which is the number of unique values
- Factors can reduce the size of a variable because they only store unique values, but could be buggy if not used properly

LIST

Store any number of items of ANY type

```

Create List list1 <- list(first = 'a', ...)
Create Empty List vector(mode = 'list', length = 3)
Get Element list1[[1]] or list1[['first']]
Append Using Numeric Index list1[[6]] <- 2
Append Using Name list1[['newElement']] <- 2
  
```

Note: repeatedly appending to list, vector, data.frame etc. is expensive, it is best to create a list of a certain size, then fill it.

DATA FRAME

- Each column is a variable, each row is an observation
- Internally, each column is a vector
- `data.frame` is a data structure that creates a reference to a data.frame, therefore, no copying is performed

```

Create Data Frame df1 <- data.frame(col1 = v1,
                                      col2 = v2,
                                      v3)
Dimension nrow(df1); ncol(df1); dim(df1)
Get/Set Column Names names(df1)
names(df1) <- c(...)
Get/Set Row Names rownames(df1)
rownames(df1) <- c(...)
Preview head(df1, n = 10); tail(...)
Get Data Type class(df1) # is data frame
df1[c('col1') or df1[[1]],]
Index by Column(s) df1[c('col1', 'col2')] or df1[c(1, 3)]
Index by Rows and Columns df1[c(1, 3), 2:3] # returns data from row 1 & 3, columns 2 to 3
  
```

+ Index method: `df1[[1]] or df1[, 'col1'] or df1[, 1]` returns as a vector. To return single column

`data.frame while using single-square brackets, use drop=df1[, 'col1', drop = FALSE]`

DATA.TABLE

What is a data.table

- Extends and enhances the functionality of data.frames
- By default data frame turns character data into factors, while data.table does not

- When you print data frame data, all data prints to the console, with a data.table, it intelligently prints the first and last five rows
- Key Difference:** Data tables are fast because they have an index like a database.

i.e., this search, `df1[,1] > number`, does a sequential scan (vector scan). After you create a key for this, it will be much faster via binary search.

```

Create data.table from data.frame data.table(df1)
Index by Column(s)* df1[, 'col1', with = FALSE] or df1[, list(col1)]
Show info for each data.table in memory (i.e., size...) tables()
Show Keys in data.table key(df1)
Create index for col1 and reorder data according to col1 setkey(df1, col1)
Use Key to Select Data df1[c('col1Value1', 'col1Value2'), ]
Multiple Key Select df1[J('1', c('2', '3'))], ]
Aggregation** df1[, list(col1 = mean(col1), col2Sum = sum(col2)), by = list(col3, col4)]
  
```

* Accessing columns must be done via list of actual names, not as characters. If column names are characters, then "with" argument should be set to FALSE

** Aggregate and dplyr functions will work, but built-in aggregation functionality of data.table is faster

MATRIX

- Similar to data.frame except every element must be the SAME type, most commonly all numerics
- Functions that work with data.frame should work with matrix as well

```

Create Matrix matrix1 <- matrix(1:10, nrow = 5), # fills rows 1 to 5, column 1 with 1:5, and column 2 with 6:10
Matrix matrix1 $t %*% t(matrix2)
Matrix Multiplication # where t() is transpose
  
```

ARRAY

- Multidimensional vector of the SAME type
- `array1 <- array(1:12, dim = c(2, 3, 2))`
- Using arrays is not recommended
- Matrices are restricted to two dimensions while array can have any dimension





Open in app

Get started

Sources

- W3Schools.com
- DataQuest.io

SQL

CHEATSHEET

CONSIDER
SUPPORTING ME

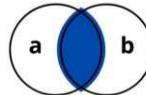
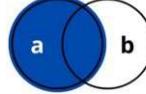
@AbzAaron

**Commands / Clauses**

SELECT	Select data from database
FROM	Specify table we're pulling from
WHERE	Filter query to match a condition
AS	Rename column or table with alias
JOIN	Combine rows from 2 or more tables
AND	Combine query conditions. All must be met
OR	Combine query conditions. One must be met
LIMIT	Limit rows returned. See also FETCH & TOP
IN	Specify multiple values when using WHERE
CASE	Return value on a specified condition
IS NULL	Return only rows with a NULL value
LIKE	Search for patterns in column
COMMIT	Write transaction to database
ROLLBACK	Undo a transaction block

ALTER TABLE	Add/Remove columns from table
UPDATE	Update table data
CREATE	Create TABLE, DATABASE, INDEX or VIEW
DELETE	Delete rows from table
INSERT	Add single row to table
DROP	Delete TABLE, DATABASE, or INDEX

GROUP BY	Group data into logical sets
ORDER BY	Set order of result. Use DESC to reverse order
HAVING	Same as WHERE but filters groups
COUNT	Count number of rows
SUM	Return sum of column
Avg	Return average of column
MIN	Return min value of column
MAX	Return max value of column

Joins**a INNER JOIN b****a LEFT JOIN b****a RIGHT JOIN b****a FULL OUTER JOIN b****Data Definition Language****CREATE**

```
CREATE DATABASE MyDatabase;
CREATE TABLE MyTable (
    id int,
    name varchar(10));
CREATE INDEX IndexName
ON TableName(col1);
```

ALTER

```
ALTER TABLE MyTable
DROP COLUMN col5;
ALTER TABLE MyTable
ADD col5 tinyint;
```

DROP

```
DROP DATABASE MyDatabase;
DROP TABLE MyTable;
```

Order Of Execution

- 1 **FROM**
- 2 **WHERE**
- 3 **GROUP BY**
- 4 **HAVING**
- 5 **SELECT**
- 6 **ORDER BY**
- 7 **LIMIT**

Data Manipulation Language**UPDATE**

```
UPDATE MyTable
SET col1 = 56
WHERE col2 = 'something';
```

INSERT

```
INSERT INTO MyTable (col1, col2)
VALUES ('value1', 'value2');
```

DELETE

```
DELETE FROM MyTable
WHERE col1 = 'something';
```

SELECT

```
SELECT col1, col2
FROM MyTable;
```

Select all columns with filter applied

```
SELECT * FROM tbl
WHERE col > 5;
```

Select first 10 rows for two columns

```
SELECT col1, col2
FROM tbl LIMIT 10;
```

Select all columns with multiple filters

```
SELECT * FROM tbl
WHERE col1 > 5 OR col2 < 2;
```

Select all rows from col1 & col2 ordering by col1

```
SELECT col1, col2
FROM tbl ORDER BY 1;
```

Return count of rows in table

```
SELECT COUNT(*)
FROM tbl;
```

Return sum of col1

```
SELECT SUM(col1)
FROM tbl;
```

Return max value for col1

```
SELECT MAX(col1)
FROM tbl;
```

Compute summary stats by grouping col2

```
SELECT AVG(col1) FROM tbl
GROUP BY col2;
```

Combine data from 2 tables using left join

```
SELECT * FROM tbl1 AS t1 LEFT JOIN
tbl2 AS t2 ON t2.col1 = t1.col1;
```

Aggregate and filter result

```
SELECT col1,
       COUNT(*) AS total
  FROM tbl
 GROUP BY col1
 HAVING COUNT(*) > 10;
```

Implementation of CASE statement

```
SELECT col1,
CASE
    WHEN col1 > 10 THEN 'more than 10'
    WHEN col1 < 10 THEN 'less than 10'
    ELSE '10'
END AS NewColumnName
FROM tbl;
```





Open in app

Get started

SQL Basics Cheat Sheet

AGGREGATION AND GROUPING

GROUP BY groups together rows that have the same values in specified columns. It computes summaries (aggregates) for each unique combination of values.

CITY		
id	name	country_id
1	Paris	1
101	Marseille	1
102	Lyon	1
2	Berlin	2
103	Hamburg	2
104	Munich	2
3	Warsaw	4
105	Cracow	4

→

CITY		
country_id	count	
1	3	
2	3	
3	2	
4	2	

AGGREGATE FUNCTIONS

- avg(expr) – average value for rows within the group
- count(expr) – count of values for rows within the group
- max(expr) – maximum value within the group
- min(expr) – minimum value within the group
- sum(expr) – sum of values within the group

EXAMPLE QUERIES

Find out the number of cities:

```
SELECT COUNT(*)
FROM city;
```

Find out the number of cities with non-null ratings:

```
SELECT COUNT(rating)
FROM city;
```

Find out the number of distinctive country values:

```
SELECT COUNT(DISTINCT country_id)
FROM city;
```

Find out the smallest and the greatest country populations:

```
SELECT MIN(population), MAX(population)
FROM country;
```

Find out the total population of cities in respective countries:

```
SELECT country_id, SUM(population)
FROM city
GROUP BY country_id;
```

Find out the average rating for cities in respective countries if the average is above 3.0:

```
SELECT country_id, AVG(rating)
FROM city
GROUP BY country_id
HAVING AVG(rating) > 3.0;
```

SUBQUERIES

A subquery is a query that is nested inside another query, or inside another subquery. There are different types of subqueries.

SINGLE VALUE

The simplest subquery returns exactly one column and exactly one row. It can be used with comparison operators =, <, >, or >=.

This query finds cities with the same rating as Paris:

```
SELECT name FROM city
WHERE rating = (
    SELECT rating
    FROM city
    WHERE name = 'Paris'
);
```

MULTIPLE VALUES

A subquery can also return multiple columns or multiple rows. Such subqueries can be used with operators IN, EXISTS, ALL, or ANY.

This query finds cities in countries that have a population above 20M:

```
SELECT name
FROM city
WHERE country_id IN (
    SELECT country_id
    FROM country
    WHERE population > 20000000
);
```

CORRELATED

A correlated subquery refers to the tables introduced in the outer query. A correlated subquery depends on the outer query. It cannot be run independently from the outer query.

This query finds cities with a population greater than the average population in the country:

```
SELECT *
FROM city main_city
WHERE population > (
    SELECT AVG(population)
    FROM city average_city
    WHERE average_city.country_id = main_city.country_id
);
```

This query finds countries that have at least one city:

```
SELECT name
FROM country
WHERE EXISTS (
    SELECT *
    FROM city
    WHERE country_id = country.id
);
```

SET OPERATIONS

Set operations are used to combine the results of two or more queries into a single result. The combined queries must return the same number of columns and compatible data types. The names of the corresponding columns can be different.

CYCLING		
id	name	country
1	YK	DE
2	ZG	DE
3	WT	PL
...

SKATING		
id	name	country
1	YK	DE
2	DF	DE
3	AK	PL
...

UNION

UNION combines the results of two result sets and removes duplicates.

UNION ALL doesn't remove duplicate rows.

```
SELECT name
FROM cycling
WHERE country = 'DE'
UNION / UNION ALL
SELECT name
FROM skating
WHERE country = 'DE';
```



INTERSECT

INTERSECT returns only rows that appear in both result sets.

```
SELECT name
FROM cycling
WHERE country = 'DE'
INTERSECT
SELECT name
FROM skating
WHERE country = 'DE';
```



EXCEPT

EXCEPT returns only the rows that appear in the first result set but do not appear in the second result set.

```
SELECT name
FROM cycling
WHERE country = 'DE'
EXCEPT / MINUS
SELECT name
FROM skating
WHERE country = 'DE';
```



Try out the interactive [SQL Basics](#) course at [LearnSQL.com](#), and check out our other SQL courses.

LearnSQL.com is owned by Vertabelo SA
vertabelo.com | CC BY-NC-ND Vertabelo SA

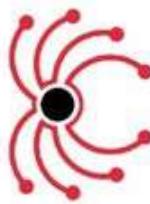
Source : <https://learnsql.com/>

Web Scraping Tools



[Open in app](#)[Get started](#)

Web Scraping



Web Scraping

Crawl arbitrary websites, extract structured data from them and export it to formats such as Excel, CSV or JSON.

[Source](#)

1. [Data Camp \(Python webscraping\)](#)
2. [Beautifulsoup -Tutorialspoint](#)
3. [Web Scraping with Scrapy and MongoDB](#)
4. [Beautiful Soup — Akul's Blog](#)
5. [Beautiful Soup — Cheatography](#)
6. [Selenium — Intellipaat](#)
7. [Selenium — Automatetheplanet](#)

Some blogs and GitHub repositories with more details : —

[Medium Blog by Frank](#)

[Blog by Hartley Brody](#)



[Open in app](#)[Get started](#)

Use web scraping to become more competitive



[Source](#)

Here is am article on Legal issues associated with Web scraping.

Data Visualization Tools

Tableau

1. [Intellipaat](#)
2. [Tableau community](#)
3. [howto](#)
4. [Datacamp](#)

Maths and Statistics

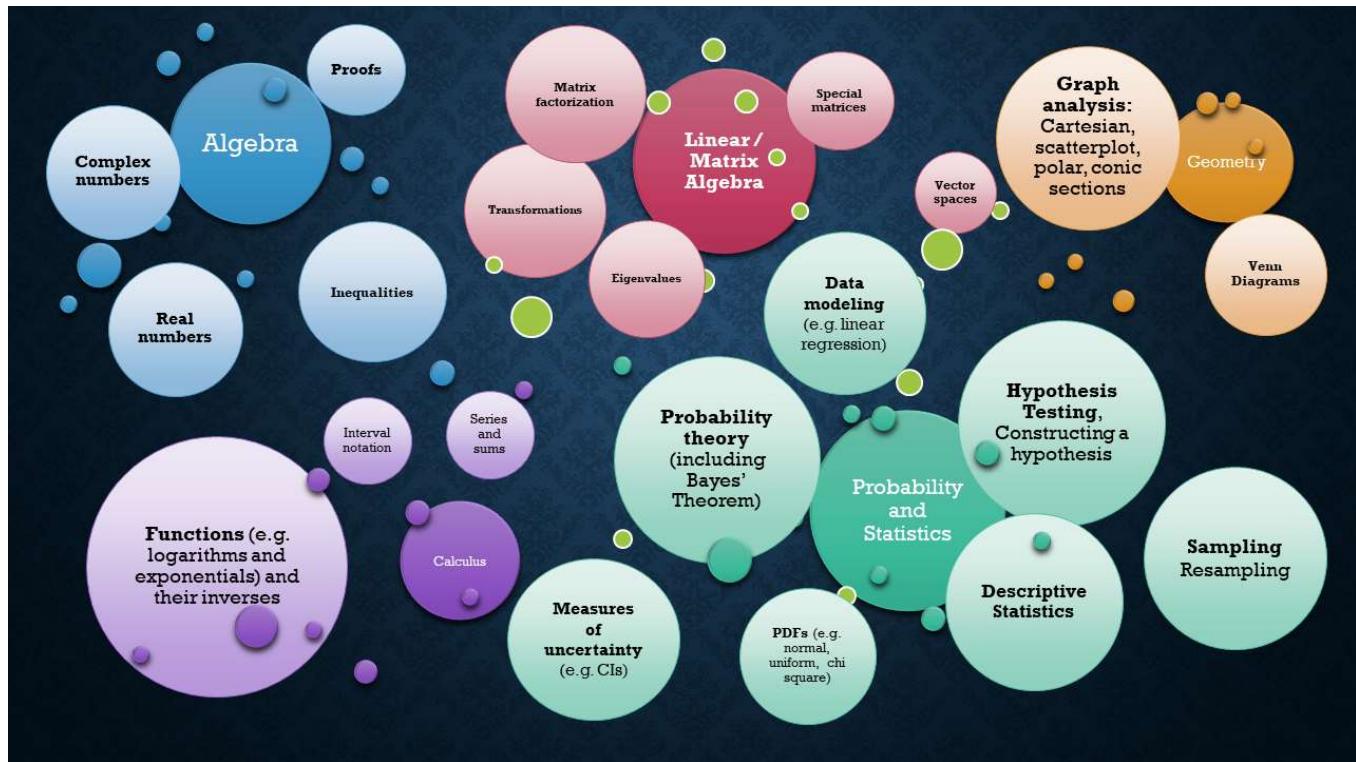
1. [Probability](#)
2. [Statistics](#)



Open in app

Get started

6. Stats for interviews



Machine Learning Algorithms





Open in app

Get started

IDATHA

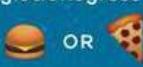
MACHINE LEARNING IN EMOJI



- Supervised
- Unsupervised
- Reinforcement

- **Supervised:** Humans build models based on input/output.
- **Unsupervised:** Human input, machine output, human utilizes if satisfactory.
- **Reinforcement:** Human input, machine output, human reward/punish, cycle continues.

BASIC REGRESSION

- **Linear** `linear_model.LinearRegression()`
Lots of numerical data 
- **Logistic** `linear_model.LogisticRegression()`
Target variable is categorical 

CLASSIFICATION

- **Neural net** `neural_network.MLPClassifier()`
Complex relationships, prone to overfitting.
Basically magic. 
- **K-NN** `neighbors.KNeighborsClassifier()`
Group membership based on proximity 
- **Decision tree** `tree.DecisionTreeClassifier()`
if/then/else Non - contiguous data
Can also be regression 
- **Random forest** `ensemble.RandomForestClassifier()`
Find best split randomly, can also be regression 
- **SVM** `svm.SVC()` `sv.LinearSVC()`
Maximum margin classifier. Fundamental Data Science algorithm 
- **Naive Bayes** 
Updating knowledge step by step with new info
`GaussianNB()` `MultinomialNB()` `BernoulliNB()`

CLUSTER ANALYSIS

- **K-MEANS** `cluster.KMeans()`
Similar datum into groups based on centroids 
- **Anomaly detection** `covariance.EllipticalEnvelope()`
Finding outliers through grouping 

FEATURE REDUCTION

- **T-Distrib. stochastic** `manifold.TSNE()`
Visualize high dimensional data.
Convert similarity to joint probabilities 
- **Principle component analysis** `decomposition.PCA()`
Distill feature space into components that describe greatest variance 
- **Canonical correlation analysis** `decomposition.CCA()`
Making sense of cross-correlation matrices 
- **Linear discriminant analysis** `lda.LDA()`
Linear combination of features that separate classes 

OTHER IMPORTANT CONCEPTS

- BIAS VARIANCE TRADEOFF**
- UNDERFITTING / OVERFITTING**
- INERTIA**
- ACCURACY FUNCTION** $(TP + TN) / (P+N)$
- Precision Function** $TP / (TP + FP)$
- Specificity Function** $TN / (FP + TN)$
- Sensitivity Function** $TP / (TP + FN)$

Source : <https://twitter.com/IDATHAuy/status/1035531415893540864>

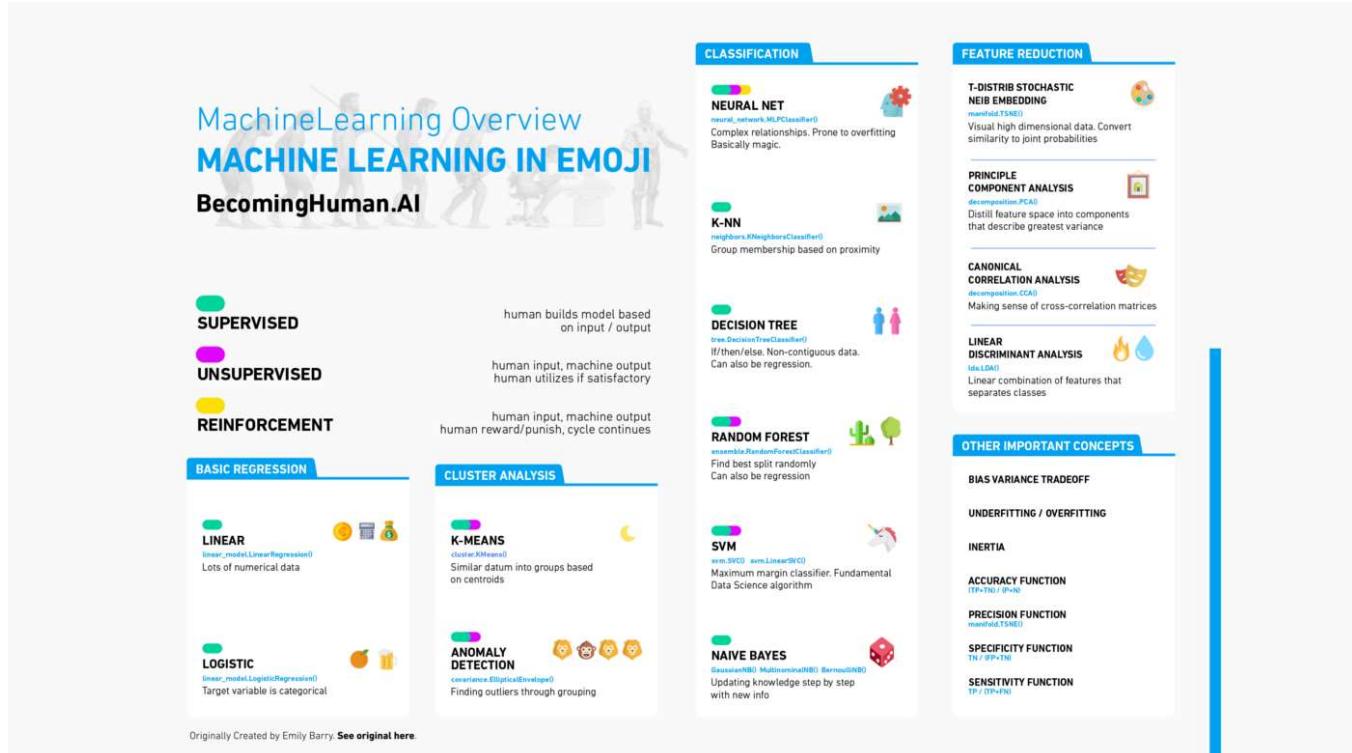
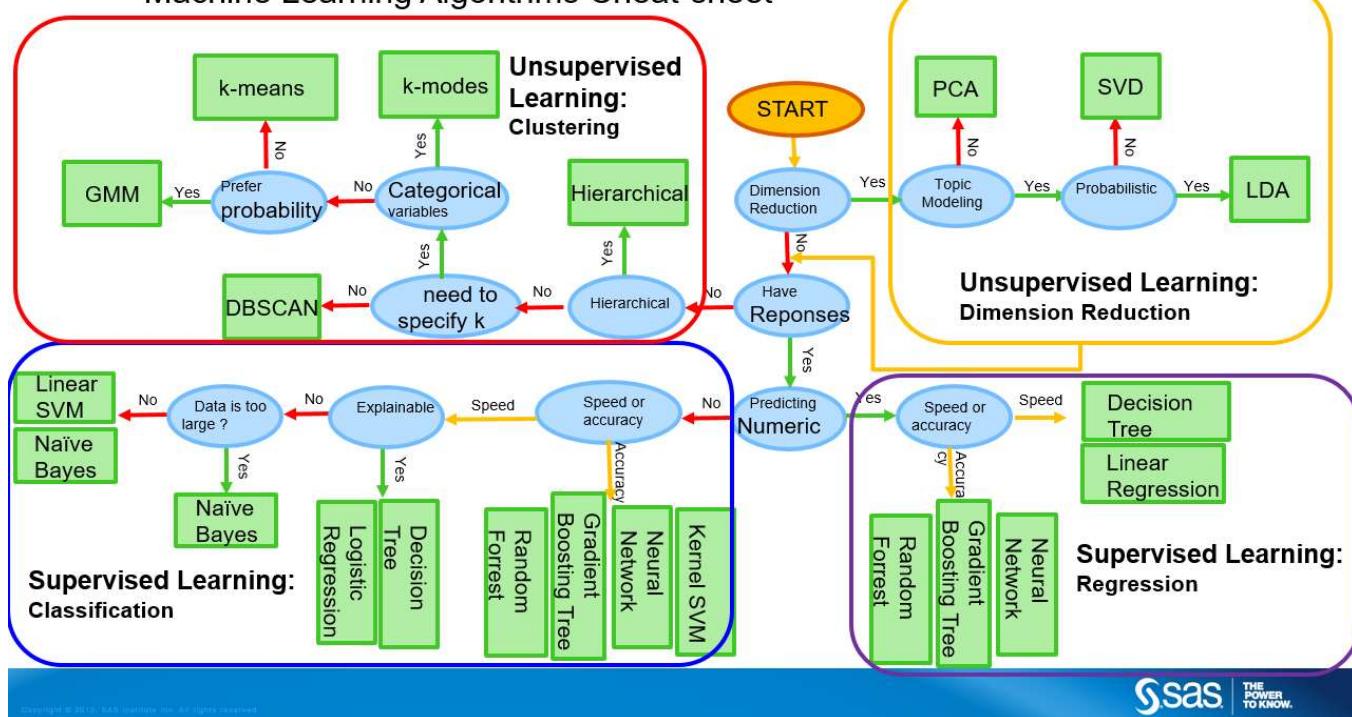
Machine Learning Algorithms

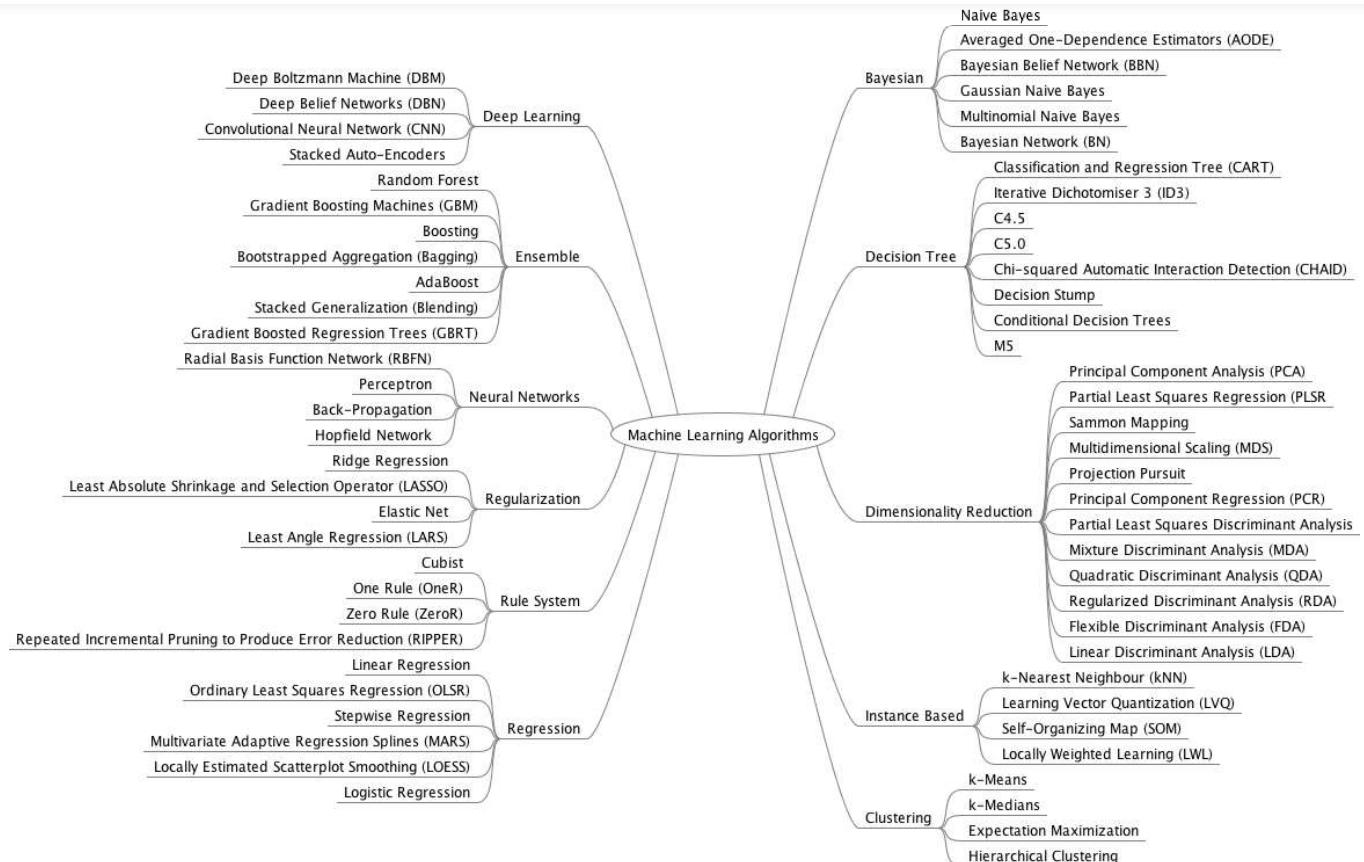




Open in app

Get started

Sas_mlR_bloggersSource : <https://becominghuman.ai/>**Machine Learning Algorithms Cheat-sheet**Source : <https://coursebapu.com/>

[Open in app](#)[Get started](#)

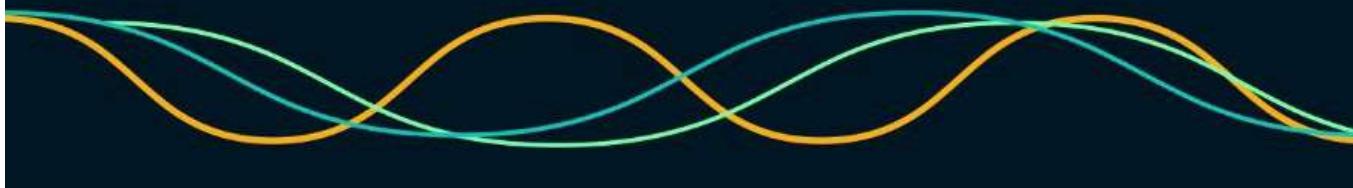


Open in app

Get started



TOP PREDICTION ALGORITHMS



Type	Name	Description	Advantages	Disadvantages
------	------	-------------	------------	---------------

Linear	Linear regression	The “best fit” line through all data points. Predictions are numerical.	Easy to understand -- you clearly see what the biggest drivers of the model are.	<ul style="list-style-type: none"> ✗ Sometimes too simple to capture complex relationships between variables. ✗ Tendency for the model to “overfit”.
	Logistic regression	The adaptation of linear regression to problems of classification (e.g., yes/no questions, groups, etc.)	Also easy to understand.	<ul style="list-style-type: none"> ✗ Sometimes too simple to capture complex relationships between variables. ✗ Tendency for the model to “overfit”.

Tree-based	Decision tree	A graph that uses a branching method to match all possible outcomes of a decision.	Easy to understand and implement.	<ul style="list-style-type: none"> ✗ Not often used on its own for prediction because it's also often too simple and not powerful enough for complex data.
	Random Forest	Takes the average of many decision trees, each of which is made with a sample of the data. Each tree is weaker than a full decision tree, but by combining them we get better overall performance .	A sort of “wisdom of the crowd”. Tends to result in very high quality models. Fast to train.	<ul style="list-style-type: none"> ✗ Can be slow to output predictions relative to other algorithms. ✗ Not easy to understand predictions.
	Gradient Boosting	Uses even weaker decision trees, that are increasingly focused on “hard” examples .	High-performing.	<ul style="list-style-type: none"> ✗ A small change in the feature set or training set can create radical changes in the model. ✗ Not easy to understand predictions.

Neural networks	Neural networks	Mimics the behavior of the brain. Neural networks are interconnected neurons that pass messages to each other. Deep learning uses several layers of neural networks put one after the other .	Can handle extremely complex tasks - no other algorithm comes close in image recognition.	<ul style="list-style-type: none"> ✗ Very, very slow to train, because they have so many layers. Require a lot of power. ✗ Almost impossible to understand predictions.
-----------------	-----------------	--	---	---



©2017 Dataiku, Inc. | www.dataiku.com | contact@dataiku.com | @dataiku





Open in app

Get started

the world of machine learning algorithms – a summary

regression

Ordinary Least-Squares Regression (OLSR)
 Linear Regression
 Logistic Regression
 Stepwise Regression
 Multivariate Adaptive Regression Splines (MARS)
 Locally Estimated Scatterplot Smoothing (LOESS)
 Jackknife Regression

regularization

Ridge Regression
 Least Absolute Shrinkage and Selection Operator (LASSO)
 Elastic Net
 Least-Angle Regression (LARS)

instance based

also called **case-based, memory-based**

k-Nearest Neighbour (kNN)
 Learning Vector Quantization (LVQ)
 Self-Organizing Map (SOM)
 Locally Weighted Learning (LWL)

dimensionality reduction

Principal Component Analysis (PCA)
 Principal Component Regression (PCR)
 Partial Least Squares Regression (PLSR)
 Sammon Mapping
 Multidimensional Scaling (MDS)
 Projection Pursuit
 Discriminant Analysis (LDA, MDA, QDA, FDA)

deep learning

Deep Boltzmann Machine (DBM)
 Deep Belief Networks (DBN)
 Convolutional Neural Network (CNN)
 Stacked Auto-Encoders

associated rule

Apriori
 Eclat
 FP-Growth

ensemble

Logit Boost (Boosting)
 Bootstrapped Aggregation (Bagging)
 AdaBoost
 Stacked Generalization (blending)
 Gradient Boosting Machines (GBM)
 Gradient Boosted Regression Trees (GBRT)
 Random Forest

think big data

bayesian

Naïve Bayes
 Gaussian Naïve Bayes
 Multinomial Naïve Bayes
 Averaged One-Dependence Estimators (AODE)
 Bayesian Belief Network (BBN)
 Bayesian Network (BN)
 Hidden Markov Models
 Conditional random fields (CRFs)

decision tree

Classification and Regression Tree (CART)
 Iterative Dichotomiser 3 (ID3)
 C4.5 and C5.0 (different versions of a powerful approach)
 Chi-squared Automatic Interaction Detection (CHAID)
 Decision Stump
 M5
 Random Forests
 Conditional Decision Trees

clustering

Single-linkage clustering
 k-Means
 k-Medians
 Expectation Maximisation (EM)
 Hierarchical Clustering
 Fuzzy clustering
 DBSCAN
 OPTICS algorithm
 Non Negative Matrix Factorization
 Latent Dirichlet allocation (LDA)

neural networks

Self Organizing Map
 Perceptron
 Back-Propagation
 Hopfield Network
 Radial Basis Function Network (RBFN)
 Backpropagation
 Autoencoders
 Hopfield networks
 Boltzmann machines
 Restricted Boltzmann Machines
 Spiking Neural Networks
 Learning Vector quantization (LVQ)

...and others

Support Vector Machines (SVM)
 Evolutionary Algorithms
 Inductive Logic Programming (ILP)
 Reinforcement Learning (Q-Learning, Temporal Difference, State-Action-Reward-State-Action (SARSA))
 ANOVA
 Information Fuzzy Network (IFN)
 Page Rank
 Conditional Random Fields (CRF)





Open in app

Get started

2. Stanford(deep learning)

3. Stanford(neural nets)

4. Datacamp (Keras)

5. Aimov institute

INTRO TO DEEP LEARNING

SUPERVISED LEARNING

INPUT: X	OUTPUT: Y	NN TYPE
HOME FEATURES	PRICE	STANDARD NN
AD+USER INFO	WILL CLICK ON AD (0/1)	
IMAGE	OBJECT (1...1000)	CONV. NN (CNN)
AUDIO ENGLISH	TEXT TRANSCRIPT CHINESE	RECURRENT NN (RNN)
IMAGE/RADAR	POS OF OTHER CARS	CUSTOM/HYBRID

NNs CAN DEAL WITH BOTH STRUCTURED & UNSTRUCTURED DATA

STRUCTURED: "THE QUICK BROWN FOX"
UNSTRUCTURED: "HUMANS ARE GOOD AT THIS"

WHY NOW?

LOTS OF DATA
HARDWARE
OPTIMIZED ALGO'S
LARGE NN
MED NN
SMALL NN
CLASSIC ML

AMT. OF DATA TABLED

ONE OF THE BIG BREAKTHROUGHS HAS BEEN MOVING FROM SIGMOID TO RELU FOR FASTER GRADIENT DESCENT

IDEA → EXPERIMENT → CODE

SIGMOID:
RELU:

FASTER COMPUTATION IS IMPORTANT TO SPEED UP THE ITERATIVE PROCESS

© Tessellated

Source : <https://sketchnotearmy.com/>

Tensorflow

1. BecomingHuman.AI

2. Altoros

3. Github

4. Stanford.edu





Open in app

Get started



About

TensorFlow

TensorFlow™ is an open source software library for numerical computation using data flow graphs. TensorFlow was originally developed for the purposes of conducting machine learning and deep neural networks research, but the system is general enough to be applicable in a wide variety of other domains as well.

Skflow

Scikit Flow provides a set of high level model classes that you can use to easily integrate with your existing Scikit-learn pipeline code. Scikit Flow is a simplified interface for TensorFlow, to get people started on predictive analytics and data mining. Scikit Flow has been merged into TensorFlow since version 0.8 and now called TensorFlow Learn.

Keras

Keras is a minimalist, highly modular neural networks library, written in Python and capable of running on top of either TensorFlow or Theano

Installation

How to install new package in Python:

```
pip install <package-name>
```

Example: pip install requests

How to install tensorflow?

```
device = cpu/gpu
```

```
python_version = cp27/cp34
```

```
sudo pip install
```

```
https://storage.googleapis.com/tensorflow/linux/$device/tensorflow-0.8.0-$python_version-none-linux_x86_64.whl
```

How to install Skflow

```
pip install sklearn
```

How to install Keras

```
pip install keras
```

update ~/.keras/keras.json - replace "theano" by "tensorflow"

Helpers

Python helper

Important functions

`type(object)`

Get object type

`help(object)`

Get help for object (list of available methods, attributes, signatures and so on)

`dir(object)`

Get list of object attributes (fields, functions)

`str(object)`

Transform an object to string

`object?`

Shows documentations about the object

`globals()`

Return the dictionary containing the current scope's global variables.

`locals()`

Update and return a dictionary containing the current scope's local variables.

`id(object)`

Return the identity of an object. This is guaranteed to be unique among simultaneously existing objects.

```
import __builtin__
dir(__builtin__)
```

Other built-in functions

TensorFlow

Main classes

```
tf.Graph()
tf.Operation()
tf.Tensor()
tf.Session()
```

Some useful functions

```
tf.get_default_session()
tf.get_default_graph()
tf.reset_default_graph()
ops.reset_default_graph()
tf.device("/cpu:0")
tf.name_scope(value)
tf.convert_to_tensor(value)
```

TensorFlow Optimizers

```
GradientDescentOptimizer
AdadeltaOptimizer
AdagradOptimizer
```

MomentumOptimizer

AdamOptimizer

FtrlOptimizer

RMSPropOptimizer

Reduction

reduce_sum

reduce_prod

reduce_min

reduce_max

reduce_mean

reduce_all

reduce_any

accumulate_n

Activation functions

`tf.nn?`

relu

relu6

elu

softplus

softsign

dropout

bias_add

sigmoid

tanh

sigmoid_cross_entropy_with_logits

softmax

log_softmax

softmax_cross_entropy_with_logits

sparse_softmax_cross_entropy_with_logits

weighted_cross_entropy_with_logits

etc.

Skflow

Main classes

```
TensorFlowClassifier
TensorFlowRegressor
```

TensorFlowDNNClassifier

TensorFlowDNNRegressor

TensorFlowLinearClassifier

TensorFlowLinearRegressor

TensorFlowRNNClassifier

TensorFlowRNNRegressor

TensorFlowEstimator

Each classifier and regressor have following fields

`n_classes=0` (Regressor), `n_classes` are expected to be input (Classifiers)

`batch_size=32,`

`steps=200, // except`

`TensorFlowRNNClassifier - there is 50`

`optimizer='Adagrad', learning_rate=0.1,`





Open in app

Get started

1. BuggyProgrammer

2. DataCamp

3. Quick tutorial

Keras Cheat Sheet

BecomingHuman.AI

DataCamp

Keras is a powerful and easy-to-use deep learning library for Theano and TensorFlow that provides a high-level neural networks API to develop and evaluate deep learning models.

A Basic Example

```
>>> import numpy as np
>>> from keras.models import Sequential
>>> from keras.layers import Dense
>>> data = np.random.random(1000,10)
>>> labels = np.random.randint(0,2,1000)
>>> model = Sequential()
>>> model.add(Dense(32, input_dim=10))
>>> model.add(Dense(16, activation='sigmoid'))
>>> model.compile(optimizer='rmsprop',
    loss='binary_crossentropy',
    metrics=['accuracy'])
>>> accuracy = model.fit(data, labels,
    epochs=10)
```

Keras Data Sets

Your data needs to be stored as Numpy arrays or as a list of Numpy arrays. Ideally, you split the data in training and test sets, for which you can also resort to the `train_test_split` module of `sklearn.cross_validation`.

```
>>> from keras.datasets import boston_housing
>>> (x_train, y_train), (x_test, y_test) = boston_housing.load_data()
>>> x_train.shape, y_train.shape, x_test.shape, y_test.shape
(<1000, 13), (1000, 1), (<1000, 13), (1000, 1)
>>> x_train[0]
array([ 5.0, 17.0,  1.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  2.0])
>>> y_train[0]
array([ 24.0])
>>> model = Sequential()
>>> model.add(Dense(64, activation='relu', input_dim=13))
>>> model.add(Dense(32, activation='relu'))
>>> model.add(Dense(1, activation='linear'))
>>> model.compile(optimizer='rmsprop',
    loss='mse')
>>> model.fit(x_train, y_train, epochs=10, batch_size=32)
>>> predictions = model.predict(x_test)
```

Other

```
>>> from tensorflow import session
>>> data = np.load('fashion_mnist_train_images.npy')
>>> data = data.reshape((60000, 28, 28, 1))
>>> data = data.astype('float32') / 255
>>> data.shape
(60000, 28, 28, 1)
>>> labels = np.load('fashion_mnist_train_labels.npy')
>>> labels.shape
(60000, 1)
>>> model = Sequential()
>>> model.add(Flatten(input_shape=(28, 28, 1)))
>>> model.add(Dense(128, activation='relu'))
>>> model.add(Dropout(0.2))
>>> model.add(Dense(10, activation='softmax'))
```

Model Architecture

Sequential Model

```
>>> from keras.models import Sequential
>>> model = Sequential()
>>> model.add(Dense(32, input_dim=10))
>>> model.add(Dense(16, activation='relu'))
>>> model.add(Dense(8, activation='relu'))
>>> model.add(Dense(1, activation='sigmoid'))
```

Multilayer Perceptron (MLP)

```
>>> from keras.layers import Dense
>>> model = Sequential()
>>> model.add(Dense(32, input_dim=10))
>>> model.add(Dense(16, activation='relu'))
>>> model.add(Dense(8, activation='relu'))
>>> model.add(Dense(1, activation='sigmoid'))
```

Binary Classification

```
>>> from keras.layers import Dense
>>> model = Sequential()
>>> model.add(Dense(32, input_dim=10))
>>> model.add(Dense(16, activation='relu'))
>>> model.add(Dense(8, activation='relu'))
>>> model.add(Dense(1, activation='sigmoid'))
```

Multi-Class Classification

```
>>> from keras.layers import Dense
>>> model = Sequential()
>>> model.add(Dense(32, input_dim=10))
>>> model.add(Dense(16, activation='relu'))
>>> model.add(Dense(8, activation='relu'))
>>> model.add(Dense(4, activation='softmax'))
```

Regression

```
>>> from keras.layers import Dense
>>> model = Sequential()
>>> model.add(Dense(32, activation='relu', input_dim=10))
>>> model.add(Dense(16, activation='relu'))
>>> model.add(Dense(1, activation='linear'))
```

Convolutional Neural Network (CNN)

```
>>> from keras.layers import Conv2D, MaxPooling2D, Flatten
>>> model = Sequential()
>>> model.add(Conv2D(32, 3, padding='same', input_shape=(100, 100, 3)))
>>> model.add(Activation('relu'))
>>> model.add(MaxPooling2D())
>>> model.add(Flatten())
>>> model.add(Dense(256))
>>> model.add(Activation('relu'))
>>> model.add(Dense(10))
>>> model.add(Activation('softmax'))
```

MLP: Binary Classification

```
>>> model.compile(optimizer='adam',
    loss='binary_crossentropy',
    metrics=['accuracy'])
```

MLP: Multi-Class Classification

```
>>> model.compile(optimizer='adam',
    loss='categorical_crossentropy',
    metrics=['accuracy'])
```

MLP: Regression

```
>>> model.compile(optimizer='rmsprop',
    loss='mse',
    metrics=['accuracy'])
```

Early Stopping

```
>>> from keras.callbacks import EarlyStopping
>>> opt = RMSprop(lr=0.001, decay=1e-4)
>>> model.compile(loss='categorical_crossentropy',
    optimizer=opt,
    metrics=['accuracy'])
>>> early_stopping = EarlyStopping(patience=2)
```

Model Fine-tuning

Model Training

```
>>> model.fit(X_train, y_train, batch_size=32, epochs=15, validation_data=(X_val, y_val))
```

Inspect Model

```
>>> model.output_shape
>>> model.summary()
>>> model.get_config()
>>> model.get_weights()
```

Prediction

```
>>> model.predict(X_val, batch_size=32)
>>> model.predict_classes(X_val, batch_size=32)
```

Evaluate Your Model's Performance

```
>>> score = model.evaluate(X_val, y_val, batch_size=32)
```

Compile Model

Preprocessing

Sequence Padding

One-Hot Encoding

Train and Test Sets

Standardization/Normalization

Neural Networks Basic Cheat Sheet

BecomingHuman.AI

Index

- Backfed Input Cell
- Input Cell
- Noisy Input Cell
- Hidden Cell
- Probabilistic Hidden Cell
- Spiking Hidden Cell
- Output Cell
- Match Input Output Cell
- Recurrent Cell
- Memory Cell
- Different Memory Cell
- Kernel
- Convolutional or Pool

Perceptron (P) **Feed Forward (FF)** **Radial Basis Network (RBF)** **Deep Feed Forward (DFF)** **Recurrent Neural Network (RNN)** **Long / Short Term Memory (LSTM)** **Gated Recurrent Unit (GRU)**

Auto Encoder (AE) **Variational AE (VAE)** **Sparse AE (SAE)** **Denoising AE (DAE)** **Markov Chain (MC)** **Hopfield Network (HN)** **Boltzmann Machine (BM)** **Restricted BM (RBm)**

Deep Believe Network (DBN) **Deep Convolutional Network (DCN)** **Deep Network (DN)** **Deep Convolutional Inverse Graphics Network (DCIGN)**

Generative Adversarial Network (GAN) **Liquid State Machine (LSM)** **Extreme Learning Machine (ELM)** **Echo Network Machine (ENM)** **Kohonen Network (KN)**

Deep Residual Network (DRN) **Support Vector Machine (SVM)** **Neural Turing Machine (NTM)**

Original Copyright by AsimovInstitute.org See original here

Source: <https://becominghuman.ai/>



Open in app

Get started

2. Computingeverywhere

3. nltk.org

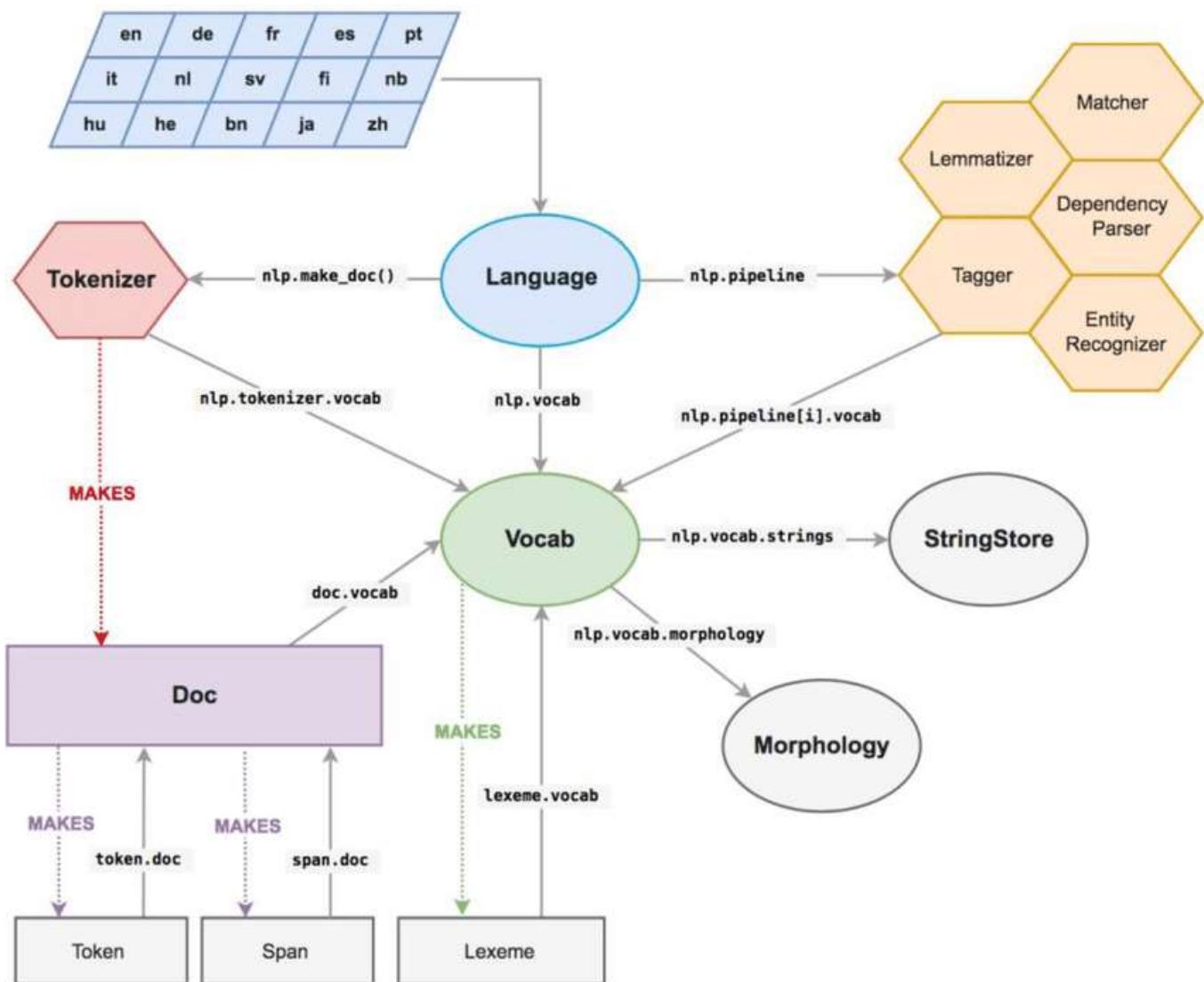
Spacy

1. Spacy.io

2. Datacamp

3. kaggle

4. Cheatography

Spacy Architecture (Source : <https://www.researchgate.net>)

IDE and OS

Tutorials





Open in app

Get started

3. Guru99

Jupyter Notebook

1. Datacamp

2. Cheatography

3. Edureka

4. Anaconda

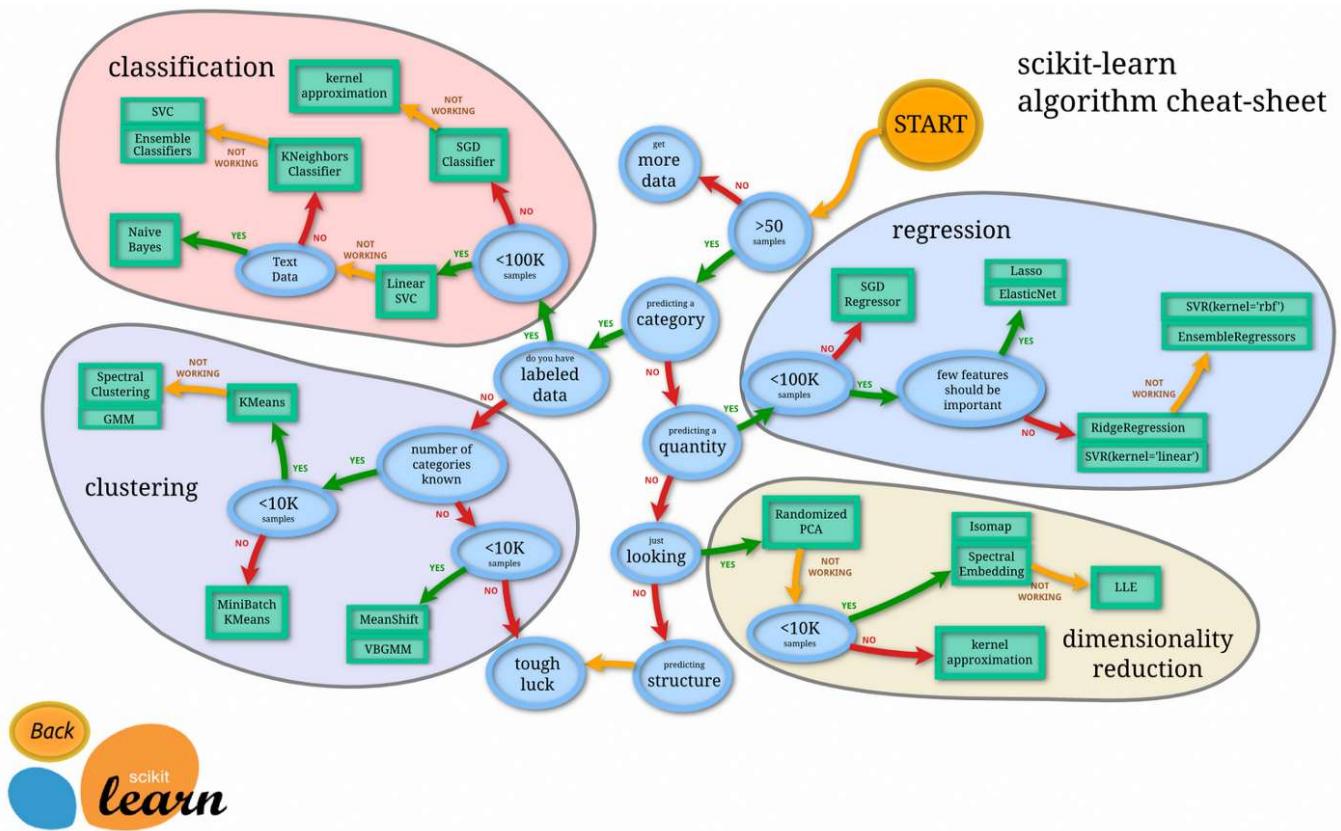
ScikitLearn

1. DataCamp

2. Edureka.co

3. Intellipaat

4. Cheatography

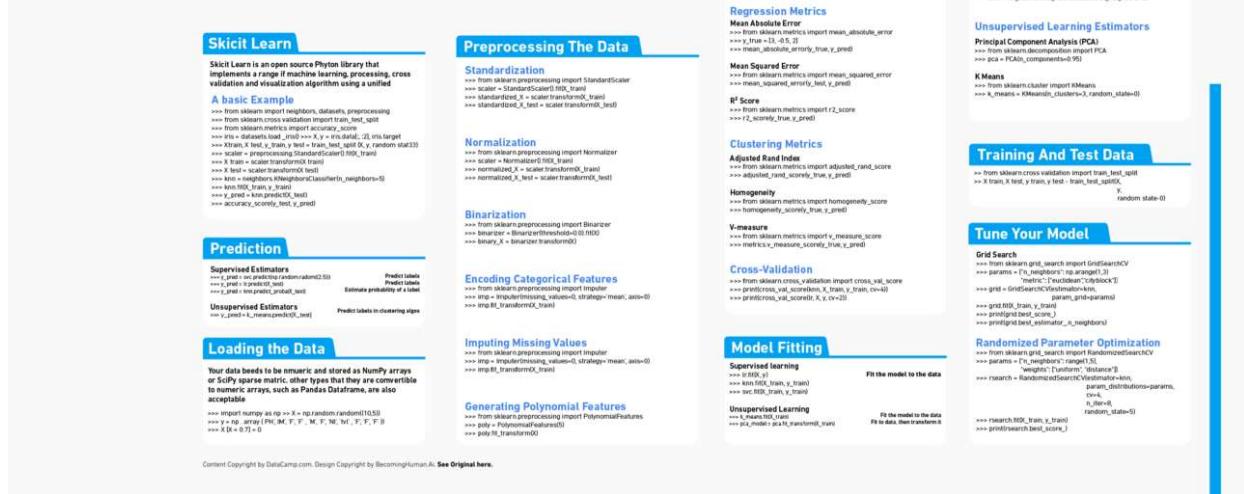




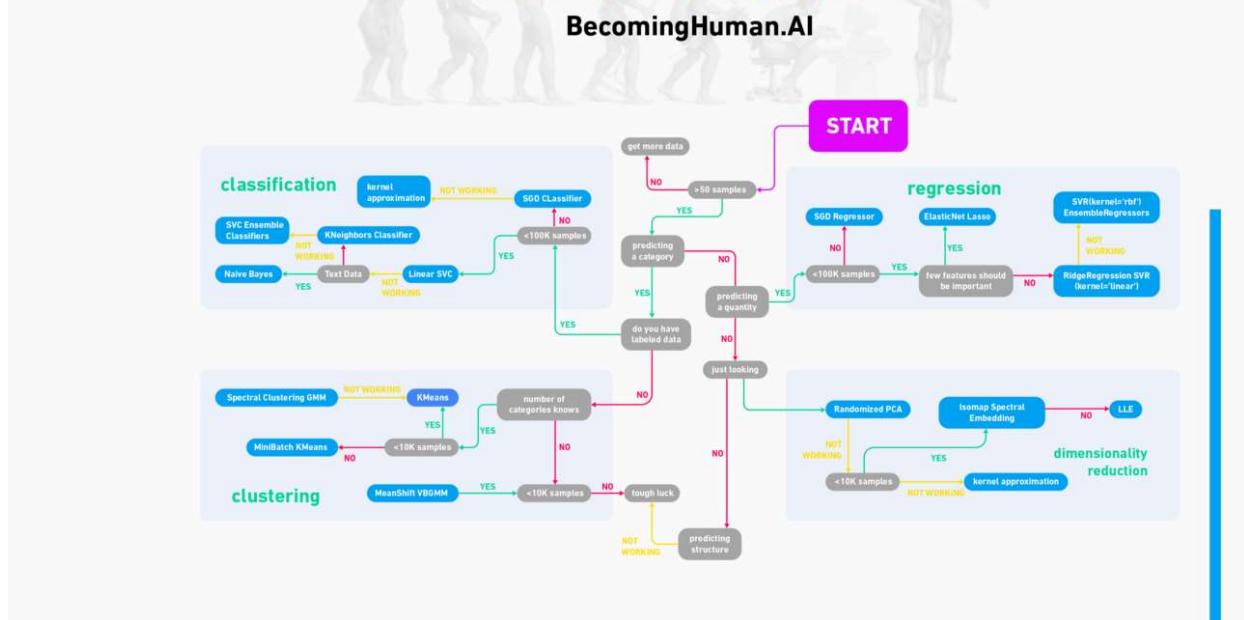
Open in app

Get started

Cheat-Sheet Skicit learn
Phyton For Data Science
BecomingHuman.AI DataCamp



Skicit-learn Algorithm



Source : <https://becominghuman.ai/>

GitHub

- ## 1. BitBucket

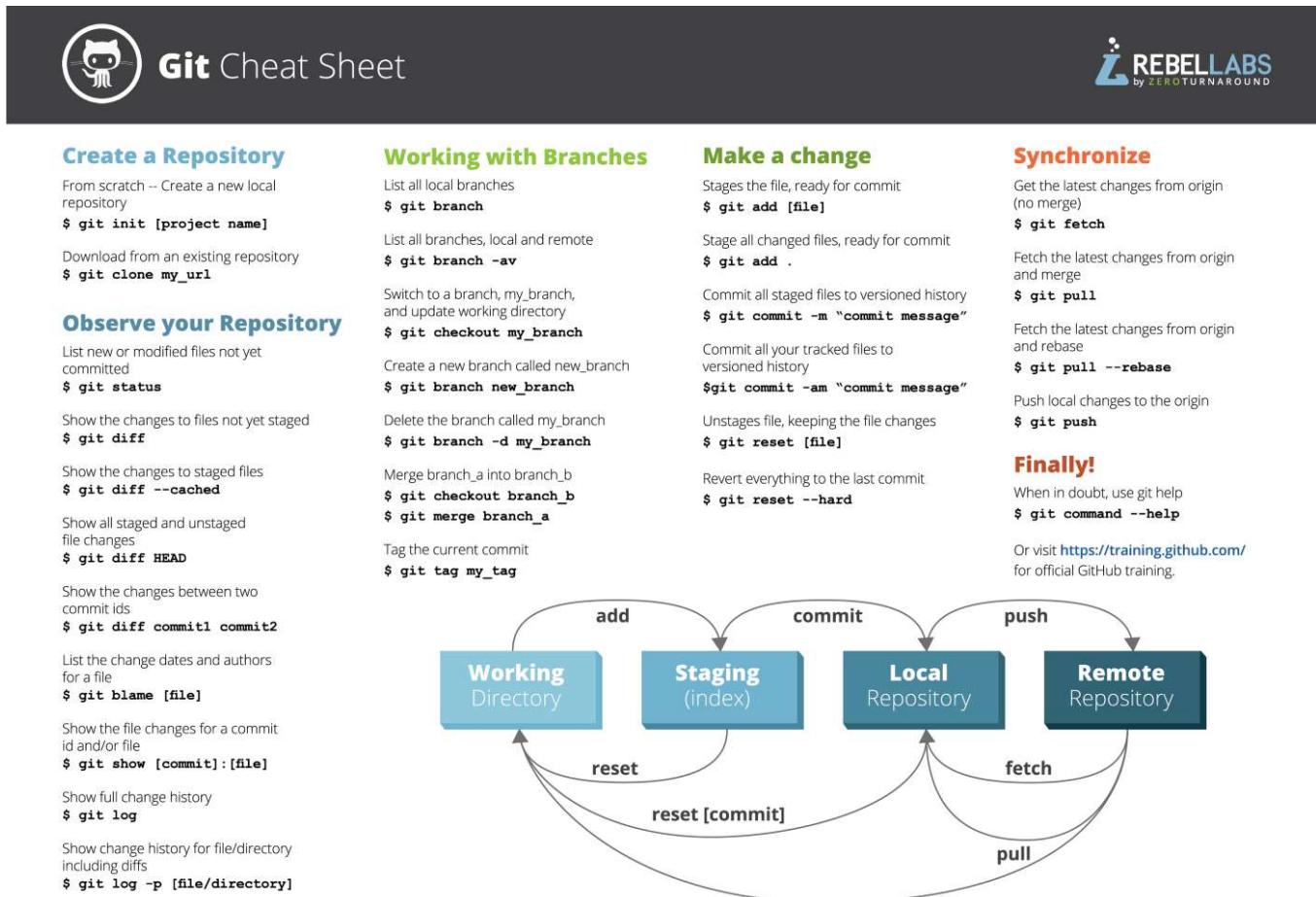
2. FreeCodeCamp



[Open in app](#)[Get started](#)

5. Git Training

6. Intellipat



Source : REBELLABS

Pytorch

1. Pytorch.org

2. Pytorch.org (for beginners)

3. KDNuggets

4. Cheatography

5. Github

6. ProgrammerSought

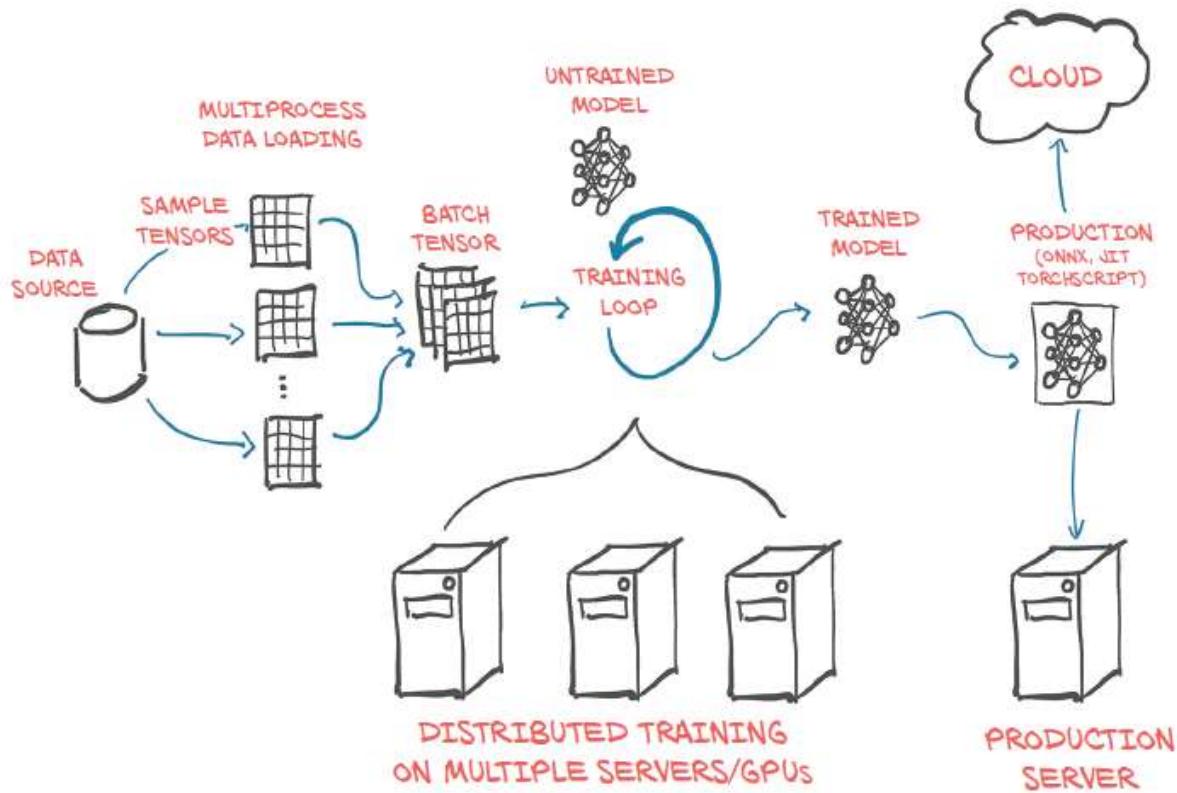
7. TechRepublic





Open in app

Get started



Pytorch Deep Learning Implementation (example)

Final Thoughts

So that was my effort in segregating and collecting the best cheat sheets on various Data Science facets. I hope it helps. In case I missed any or you guys have any suggestion or query, please do add your comments below.

I will be more than happy to add them to the list above!

Happy Learning!!

Cheers :)



[Open in app](#)[Get started](#)

Photo by [Katrina Wright](#) on [Unsplash](#)



[About](#) [Help](#) [Terms](#) [Privacy](#)



[Open in app](#)[Get started](#)[Download on the
App Store](#)[GET IT ON
Google Play](#)