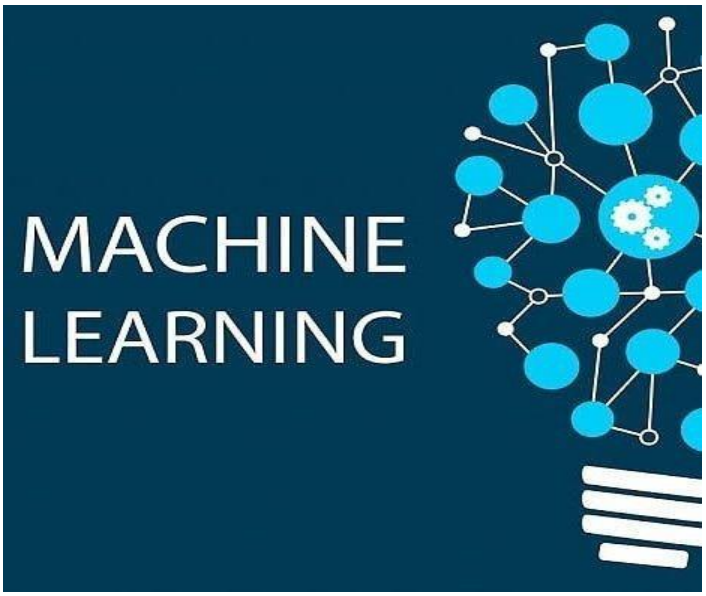


Python, Math, Machine learning

Learn.machinelearning

ABSTRACT

Learn Machine learning from scratch with Python, Linear algebra, Probability & statistics, Machine learning and Deep learning.

K UDAY KIRAN REDDY

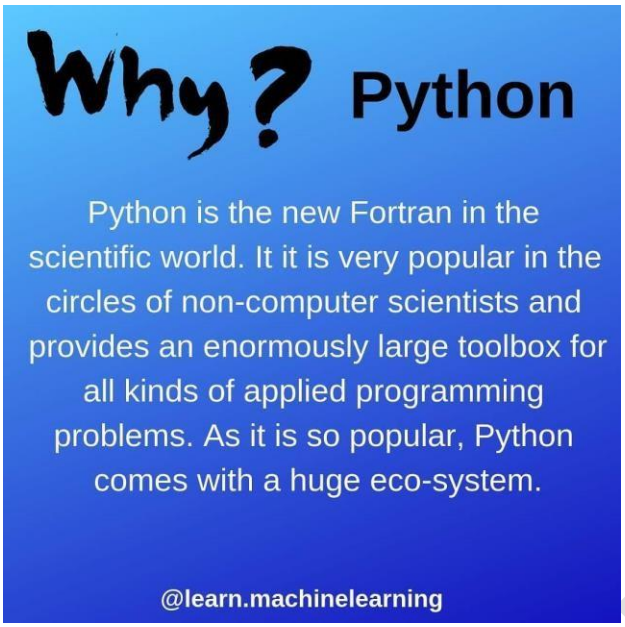
Contents

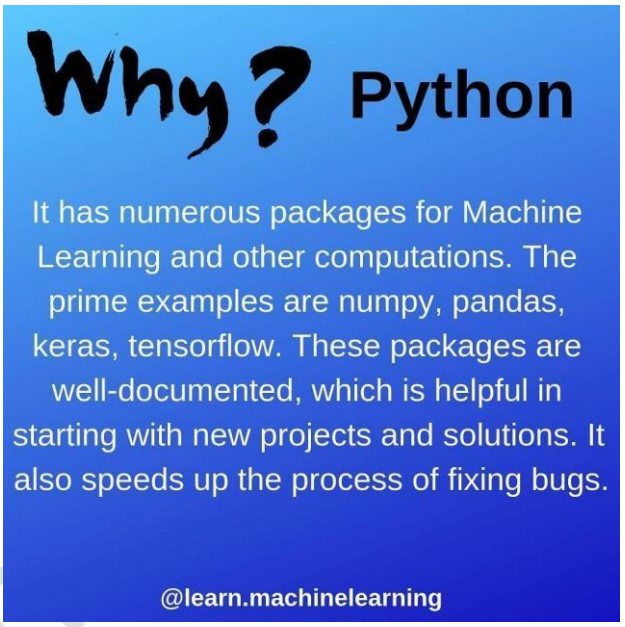
20. Missing Values
21. Outliers
22. Train, Test and validation dataset
23. Curse of Dimensionality
24. Linear regression
25. Logistic Regression
26. Polynomial regression
27. Naïve Bayes
28. K nearest Neighbors
29. Support Vector machines
30. Decision trees
31. Ensemble Methods(Random forest, gradient boosing, staccking, AdaBoost, Cascading)

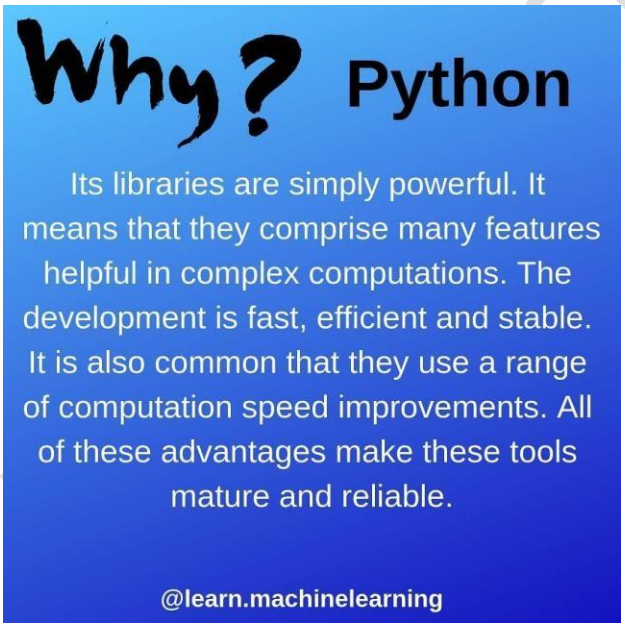
Future Work (Book Update on Regular Intervals)

1. Code for all the concepts
2. Interview Questions
3. Unsupervised Algorithms
4. Few Less known algorithms
5. And more

WHY PYTHON?


Why? Python
Python is the new Fortran in the scientific world. It it is very popular in the circles of non-computer scientists and provides an enormously large toolbox for all kinds of applied programming problems. As it is so popular, Python comes with a huge eco-system.
@learn.machinelearning


Why? Python
It has numerous packages for Machine Learning and other computations. The prime examples are numpy, pandas, keras, tensorflow. These packages are well-documented, which is helpful in starting with new projects and solutions. It also speeds up the process of fixing bugs.
@learn.machinelearning


Why? Python
Its libraries are simply powerful. It means that they comprise many features helpful in complex computations. The development is fast, efficient and stable. It is also common that they use a range of computation speed improvements. All of these advantages make these tools mature and reliable.
@learn.machinelearning


Why? Python
Another important advantage of using Python libraries is a considerable support from the community as the developers can easily find tutorials and tips valuable in a development process. A stable community makes the start threshold lower - it is easier to use new technologies from scratch.
@learn.machinelearning

WHY PYTHON?

Blogs to read:

1. Why I Think Python is Perfect for Machine Learning and Artificial Intelligence
2. Why Use Python for AI and Machine Learning?
3. Why Python is the most popular language used for Machine Learning

Videos to Watch:

1. Which Programming Language for AI? | Machine Learning
2. Best Programming Languages for Machine Learning

Important topics to learn in python

Are the most important topics in python?

- Identifiers
- Keywords
- comments
- Indentation
- Variables
- Data Types
- Operators
- Decision making

@learn.machinelearning

Are the most important topics in python?

- Loops
- Break statments
- List
- Tuple
- Dictionary
- Strings
- Functions
- Exception Handling

@learn.machinelearning

Python Concepts

Identifiers

An identifiers is a name given to program elements such as variables , array, class and functions etc. An identifier is a sequence of letters, digits, and underscores, the first character of which can not be a digit. Following rules must be kept in mind while naming an identifier.

1) The first character must be an alphabet (uppercase or lowercase) or can be an underscore.
2)An identifier can not start with a digit.
3) All succeeding characters must be alphabets or digits.
4) No special characters like !, @, #, $, % or punctuation symbols is allowed except the underscore("_")
5) No two successive underscores are allowed.
6) Keywords can not be used as identifiers.

@learn.machinelearning

Keywords

Python keywords are set words that cannot be used as an identifier in program. These words are known as "Reserved Words" or "Keywords". Python Keywords are standard identifiers and their meaning and purpose is predefined by the compiler.

list of available keywords

1) and	2) assert	3) in	4)del
5) else	6) raise	7) from	8) if
9) continue	10) not	11) pass	12) finally
13) while	14) yield	15) is	16) as
17) break	18) return	19) elif	20) except
21) def	22) gloabal	23) import	24) for
25) or	26) print	27) lambda	28) with
29) class	30) try	31) exec	

@learn.machinelearning

Comments

A hash sign (#) that is not inside a string literal begins a comment. All characters after the # and up to the end of the physical line are part of the comment and the Python interpreter ignores them. For commenting multiple lines we use. start with ''' and end with '''.

Examples

```
# This is a comment.

name = "machine learning" # This is again comment

"""
This is a
multilline comment
"""
```

@learn.machinelearning

Indentation

In Python, a block of code is defined using line indentation, any block of code is started with indentation and ends with the first un-indented line. The amount of spaces is an indentation must be consistent throughout that same block.The use of indentation makes the code look clean and consistent. we can use 4 spaces or a tab as indentation.

Example

valid block of code

```
if True:
    print("machine learning")
    print("ml from scratch")
```

Invalid block of code

```
if True:
      print("machine learning")
    print("ml from scratch")
```

@learn.machinelearning

variables

Variables is an identifier used to refer memory location in computer memory that holds a value for that variable, this value can be changed during the execution of the program. Whenever you create a variable in Python programming, this means you are allocating some space in the memory for that variable. The size of memory block allocated and type of the value it holds is completely dependent upon the type of variable. Naming a variable in Python Programming is one of the very important task.

Rules for naming a variable

1) Variable name can consist of letter, alphabets and start with underscore character.
2) First character of variable name cannot be digit.
3) Keywords are not allowed to use as a variable name.
4) Blank spaces are not allowed in variable name.
5) Python is case sensitive i.e. UPPER and lower case are significant.
6) Special characters like are not allowed except the underscore.

@learn.machinelearning

Assigning Values to Variables

In Python it not mandatory too declare a variables beforehand to use it in program. As you assign a value to a variable, the variable is declared automatically. The assignment operator (=) is used to assign values to a variable, any type of value can be assigned to any valid variable.

The operand in the left side of the assignment operator (=) indicates the name of the variable and the operand in the right side of the assignment operator (=) indicates the value stored in that variable.

Example

INPUT

```
number = 100     # An integer
float  = 1000.0  # A float
string  = "ML"  # A string
print(number)
print(float)
print(string)
```

OUTPUT

```
100
1000.0
ML
```

Multiple Assignment

INPUT

```
number,float,string = 1,2.0,"ML"
print(number)
print(float)
print(string)
```

OUTPUT

```
100
1000.0
ML
```

@learn.machinelearning

Python Concepts

Data Types

The data stored in memory can be of many types. For example, a person's age is stored as a numeric value and his or her address is stored as alphanumeric characters. Python has various standard data types that are used to define the operations possible on them and the storage method for each of them.

Python has five standard data types –

1) Numbers
2) String
3) List
4) Tuple
5) Dictionary

@learn.machinelearning

Data Type - NUMBER

Number data types store numeric values. Number objects are created when you assign a value to them.

four different numerical types –

1) int (signed integers)
2) long (long integers)
3) float (floating point real values)
4) complex (complex numbers)

Examples

1) int	a = 12
2) long	l = 123L
3) float	f = 12.3
4) complex	c = 12.3j

@learn.machinelearning

Data Type - String

Strings in Python are identified as a contiguous set of characters represented in the quotation marks. Python allows for either pairs of single or double quotes. Subsets of strings can be taken using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the string and working their way from -1 at the end.
The plus (+) sign is the string concatenation operator and the asterisk (*) is the repetition operator.

Example

INPUT	OUTPUT
string = 'machine'	
print(string)	machine
print(string[0])	m
print(string[2:5])	chi
print(string[2:])	chine
print(string * 2)	machinemachine
print(string + " learning"	machine learnig

@learn.machinelearning

Identity Operators

Identity operators compare the memory locations of two objects.

1) "is" x is y, here is results in 1 if id(x) equals id(y).
2) "is not" x is not y, here is not results in 1 if id(x) is not equal to id(y).

Logical Operators

There are following logical operators supported by Python language. Assume variable a holds 10 and variable b holds 20 then.

1) and Logical AND	(a and b) is true.
2) or Logical OR	(a or b) is true.
3) not Logical NOT	Not(a and b) is false.

Membership Operators

Python's membership operators test for membership in a sequence, such as strings, lists, or tuples.

1) "in" x in y, here in results in a 1 if x is a member of sequence y.
2) "not in" x not in y, here not in results in a 1 if x is not a member of sequence y.

Bitwise Operators

Bitwise operator works on bits and performs bit by bit operation. Assume if a = 60; and b = 13.
Binary format of a = 0011 1100 b = 0000 1101

1) & Binary AND (c = a & b)(means 0000 1100) c = 12
2) | Binary OR (c = a | b)(means 0011 1101) c = 61
3) ^ Binary XOR (c = a ^ b)(means 0011 0001) c = 49
4) ~ Binary Ones Complement (c = ~a)(means 1100 0011) c = -61
5) << Binary Left Shift (c = a << 2) (means 1111 0000) c = 240
6) >> Binary Right Shift (c = a >> 2)(means 0000 1111) c = 15

Python Concepts

Assignment Operators

Assignment operators are used to assign value to a variable, you can assign a variable value or the result of an arithmetical expression

1) = Assignment c = a+b assigns value of a + b into c
2) += Add and assign c += a is equivalent to c = c + a
3) -= subtract and assign c -= a is equivalent to c=c - a
4) *= Multiply and assign c *= a is equivalent to c = c * a
5) /= Divide and assign c /= a is equivalent to c = c / a
6) %= Modulus and assign c %= a is equivalent to c = c % a
7) **= Exponent and assign c **= a is equivalent to c = c ** a
8) //= Floor Division and assign c //= a is equivalent to c = c // a

Arithmetic Operators

Arithmetic Operators are used to perform arithmetic operations like addition, subtraction, multiplication, division, %modulus, exponent, etc.

For example assume a=5 and b=10

1)	+	Addition	a + b = 15
2)	-	Subtraction	a - b = -5
3)	*	Multiplication	a * b = 50
4)	/	Division	b / a = 2.0
5)	%	Modulus	b % a = 0(remainder)
6)	**	Exponent	b ** a = 100000
7)	//	Floor division	b // a = 2

Operators

In Python programming, an operator is a special symbol that is used to carry out some specific operation on its operand. Python operators can be used with many types of variables or constants, but some of the operators are restricted to work on specific data types. Most operators are binary, meaning they take two operands, but **a few** are unary and only take one operand.

Type of operators

1) Arithmetic Operators
2) Assignment Operators
3) Comparison (Relational) Operators
4) Logical Operators
5) Identity Operators
6) Bitwise Operators
7) Membership Operators

Data Type - Dictionary

Python's dictionaries are kind of hash table type. They work like associative arrays or hashes found in Perl and consist of key-value pairs. A dictionary key can be almost any Python type, but are usually numbers or strings. Values, on the other hand, can be any arbitrary Python object.
Dictionaries are enclosed by curly braces ({ }) and values can be assigned and accessed using squa**re brace**s ([]).

EXAMPLES

INPUT

```
dict1 = { 1 : "machine" , "learning" : 2 }
print( dict1 )
dict1 [ 3 ] = "scratch"
print( dict2 )
```

OUTPUT

```
{1 : "machine" , "learning" : 2 }
{1 : "machine" , "learning" : 2 , 3 : :scratch"}
```

Data Type - Tuples

Tuple is an ordered sequence of comma-separated values (items or elements). Tuples are same as list, but the only difference is that tuples are immutable. Once a tuple is created their elements and size can not be changed.A Tuple is defined within parentheses () where items are separated by **comma**s.

EXAMPLES

```
list1 = ( 1,2,3,4,5 )

list2 = ( 1.2,34.5,654.5 )

list3 = ( "machine","learning" )

list4 = ( 1,"machine learning", 34.43 )
```

Data Type - Lists

A list is an ordered series of comma-separated values, each of the value as termed as Item. A list variable is defined by having values separated by commas and enclosed within square brackets ([]). A list can hold values of different data types.

EXAMPLES

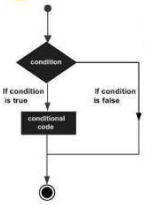
```
list1 = [1,2,3,4,5]

list2 = [1.2,34.5,654.5]

list3 = ["machine","learning"]

list4 = [1,"machine learning", 34.43]
```

Python Concepts

Operators Precedence

1) ** (Exponentiation)
2) ~ + - (Complement, unary plus and minus)
3) * / % // (Multiply, divide, modulo and floor division)
4) + - (Addition and subtraction)
5) >> << (Right and left bitwise shift)
6) & (Bitwise 'AND')
7) ^ | (Bitwise exclusive `OR' and regular `OR')
8) <= < > >= (Comparison operators)
9) <> == != (Equality operators)
10) = %= /= //= -= += *= **= (Assignment operators)
11) is is not (Identity operators)
12) in not in (Membership operators)
13) not or and (Logical operators)

Decision Making

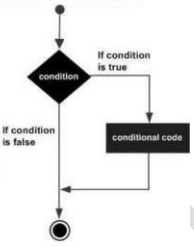


Decision making is anticipation of conditions occurring while execution of the program and specifying actions taken according to the conditions. Decision structures evaluate multiple expressions which produce TRUE or FALSE as outcome. You need to determine which action to take and which statements to execute if outcome is TRUE or FALSE otherwise.

1) **if statements -** An if statement consists of a boolean expression followed by one or more statements.
2) **if...else statements -** An if statement can be followed by an optional else statement, which executes when the boolean expression is FALSE.
3) **nested if statements -** You can use one if or else if statement inside another if or else ifstatement(s).

IF...ELIF...ELSE Statements

If the boolean expression evaluates to TRUE, then the block of statement(s) inside the if statement is executed. If boolean expression evaluates to FALSE, then the first set of code after the end of the if statement(s) is executed.



INPUT

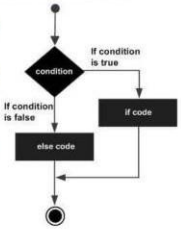
```
var1 = 100
if var1:
   print("Machine learning")
   print var1
var2 = 0
if var2:
   print(" No value")
   print(var2)
print("Happy machine learning")
```

OUTPUT

```
Machine learning
100
Happy machine learning
```

IF...ELIF...ELSE Statements

An else statement can be combined with an if statement. An else statement contains the block of code that executes if the conditional expression in the if statement resolves to 0 or a FALSE value.The else statement is an optional statement and there could be at most only one else statement following if.



INPUT

```
var1 = 100
if var1==100:
   print("Machine learning")
   print(var1)
else:
   print("Deep learning")
   print(var1)
```

OUTPUT

```
Machine learning
100
```

nested IF statements

There may be a situation when you want to check for another condition after a condition resolves to true. In such a situation, you can use the nested ifconstruct.In a nested if construct, you can have an if...elif...else construct inside another if...elif...else construct.

Syntax

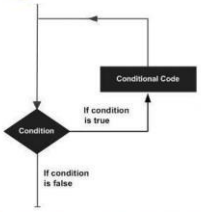
INPUT

```
var = 100
if var < 200:
   print("Machine learning")
   if var == 150:
      print("Which is 150")
   elif var == 100:
      print("Which is 100")
else:
   print("Not True")
```

OUTPUT

```
Machine learning
which is 100
```

Loops



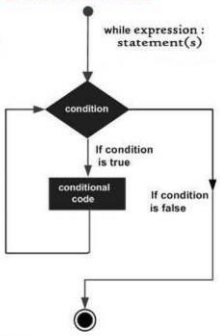
In general, statements are executed sequentially: The first statement in a function is executed first, followed by the second, and so on. There may be a situation when you need to execute a block of code several number of times.

1) while loop - Repeats a statement or group of statements while a given condition is TRUE. It tests the condition before executing the loop body.
2) for loop - Executes a sequence of statements multiple times and abbreviates the code that manages the loop variable.
3) nested loops - You can use one or more loop inside any another while, for or do..while loop.

Python Concepts

while Loop Statements

A while loop statement in Python programming language repeatedly executes a target statement as long as a given condition is true.



INPUT

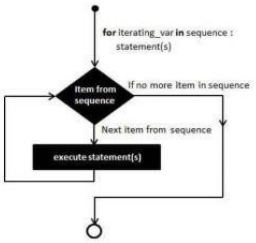
```
count = 0
while (count < 3):
  print('machine learning ',count)
  count = count + 1
print("happy machine learning")
```

OUTPUT

```
machine learning 0
machine learning 1
machine learning 2
happy machine learning
```

for Loop Statements

If a sequence contains an expression list, it is evaluated first. Then, the first item in the sequence is assigned to the iterating variable iterating_var. Next, the statements block is executed. Each item in the list is assigned to iterating_var, and the statement(s) block is executed until the entire sequence is exhausted.



INPUT

```
for count in range(0,3):
  print('machine learning ',count)
print("happy machine learning")
```

OUTPUT

```
machine learning 0
machine learning 1
machine learning 2
happy machine learning
```

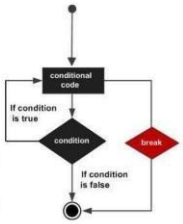
Loop Control Statements

Loop control statements change execution from its normal sequence. When execution leaves a scope, all automatic objects that were created in that scope are destroyed.

1) break statement - Terminates the loop statement and transfers execution to the statement immediately following the loop.
2) continue statement - Causes the loop to skip the remainder of its body and immediately retest its condition prior to reiterating.
3) pass statement - The pass statement in Python is used when a statement is required syntactically but you do not want any command or code to execute.

break statement

It terminates the current loop and resumes execution at the next statement, just like the traditional break statement in C.The most common use for break is when some external condition is triggered requiring a hasty exit from a loop. The break statement can be used in both while and for loops.



INPUT

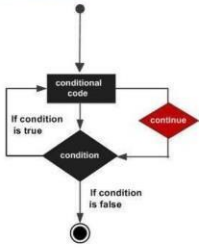
```
for letter in 'Python':
  if letter == 'h':
    break
  print('Current Letter :', letter)

print("Good bye!")
```

OUTPUT

```
Current Letter : P
Current Letter : y
Current Letter : t
Good bye!
```

continue statement

It returns the control to the beginning of the while loop.. The continuestatement rejects all the remaining statements in the current iteration of the loop and moves the control back to the top of the loop. The continue statement can be used in both while and for loops.



INPUT

```
for letter in 'Python':
  if letter == 'h':
    continue
  print('Current Letter :', letter)
print("Good bye!")
```

OUTPUT

```
Current Letter : P
Current Letter : y
Current Letter : t
Current Letter : o
Current Letter : n
Good bye!
```

pass statement

It is used when a statement is required syntactically but you do not want any command or code to execute.
The pass statement is a null operation; nothing happens when it executes. The pass is also useful in places where your code will eventually go, but has not been written yet

INPUT

```
for letter in 'Python':
  if letter == 'h':
    pass
    print('This is pass block')
  print('Current Letter :', letter)
print("Good bye!")
```

OUTPUT

```
Current Letter : P
Current Letter : y
Current Letter : t
This is pass block
Current Letter : h
Current Letter : o
Current Letter : n
Good bye!
```

Python Concepts

LIST

The list is a most versatile datatype available in Python which can be written as a list of comma-separated values (items) between square brackets. Important thing about a list is that items in a list need not be of the same type.

Examples

Creating List

```
list1 = ['machine', 'leaning', 1996, 20.30]
list2 = [1, 2, 3, 4, 5 ]
list3 = ["a", "b", "c", "d"]
```

Accessing Values

INPUT

```
print("list1[0]: ", list1[0])
print("list2[1:5]: ", list2[1:5])
```

OUTPUT

```
list1[0]:  Machine
list2[1:5]:  [2, 3, 4, 5]
```

Updating

INPUT

```
list1[2] = 2019
print("New value available at index 2 : ")
print(list[2])
```

OUTPUT

```
New value available at index 2 :
2019
```

LIST

Delete List Elements

INPUT

```
del list1[2];
print("After deleting value at index 2 : ")
print(list1)
```

OUTPUT

```
After deleting value at index 2 :
['machine', 'learning', 20.30]
```

Basic List Operations

1) Length	len([1, 2, 3])	3
2) Concatenation	[1, 2, 3] + [4, 5, 6]	[1, 2, 3, 4, 5, 6]
3) Repetition	['Hi!'] * 4	['Hi!', 'Hi!', 'Hi!', 'Hi!']
4) Membership	3 in [1, 2, 3]	True
5) Iteration	for x in [1, 2, 3]: print x,	1 2 3

LIST

Indexing, Slicing, and Matrixes

```
L = ["machine", "learning", "from", "scratch"]
```

L[2]	from	Offsets start at zero
L[-2]	from	Negative: count from the right
L[1:]	["learning","from","scratch"]	Slicing fetches sections

Built-in List Functions & Methods

function	input	function applied	result
1) cmp()	l1, l2 = [123, 'abc'], [456, 'abc']	cmp(l1,l2)	-1
2) len()	l1 = [123, 'abc', 12.23]	len(l1)	3
3) max()	l1 = [1,2,3,4,5,6]	max(l1)	6
4) min()	l1 = [1,2,3,4,6,7]	min(l1)	1
5) list()	tuple = (1,2,3,4)	list(tuple)	[1,2,3,4]

LIST

Built-in List Functions & Methods

```
l, l1  = [1,3,3,7,9] , [8,6]
```

function	function applied	input	result
1) append()	l.append(4)	print(l)	[1,3,3,7,9,4]
2) count()	l.count(3)	print(l.count(3)	2
3) extend()	l.extend(l1)	print(l.extend(l1)	[1,3,3,7,8,8,6]
4) index()	l.index(7)	print(l.index(7)	3
5) insert()	l.insert(1,10)	print(l.insert(1,10)	[1,10,3,3,7,9
6) pop()	l.pop()	print(l.pop())	9
7) remove()	l.remove(9)	print(l.remove(9))	[1,3,3,7]
8) reverse()	l.reverse()	print(l.reverse())	[9,7,3,3,1]
9) sort()	l.sort()	print(l.sort())	[1,3,3,7,9]

Tuple

A tuple is a sequence of immutable Python objects. Tuples are sequences, just like lists. The differences between tuples and lists are, the tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square brackets.

Examples

Creating tuple

```
tuple1 = ('machine', 'leaning', 1997, 20.30)
tuple2 = (1, 2, 3, 4, 5 )
tupl3  = ("a", "b", "c", "d")
```

Accessing Values

INPUT

```
print("tuple1[0]: ", tuple1[0])
print("tuple2[1:5]: ", tuple2[1:5])
```

OUTPUT

```
tuple1[0]:  Machine
tuple2[1:5]:  [2, 3, 4, 5]
```

Tuple

Basic Tuples Operations

1) Length	len((1, 2, 3))	3
2) Concatenation	(1, 2, 3) + (4, 5, 6)	(1, 2, 3, 4, 5, 6)
3) Repetition	('Hi!',) * 4	('Hi!', 'Hi!', 'Hi!', 'Hi!')
4) Membership	3 in (1, 2, 3)	True
5) Iteration	for x in (1, 2, 3): print x,	1 2 3

Indexing, Slicing, and Matrixes

```
L = ('spam', 'Spam', 'SPAM!')
```

1) L[2] 'SPAM!'	Offsets	start at zero
2) L[-2] 'Spam'	Negative:	count from the right
3) L[1:]	['Spam', 'SPAM!']	Slicing fetches sections

Python Concepts

Dictionary

Each key is separated from its value by a colon (:), the items are separated by commas, and the whole thing is enclosed in curly braces. An empty dictionary without any items is written with just two curly braces, like this: {}. Keys are unique within a dictionary while values may not be. The values of a dictionary can be of any type, but the keys must be of an immutable data type such as strings, numbers, or tuples.

Accessing Values

INPUT

```
dict = {'Name': 'machine learning', 'Age': 22}
print("dict['Name']: ", dict['Name'])
print("dict['Age']: ", dict['Age'])
```

OUTPUT

```
dict['Name']:  machine learning
dict['Age']:  22
```

Dictionary

Updating Dictionary

INPUT

```
dict = {'Name': 'machine learning', 'Age': 1}
dict['Age'] = 8; # update existing entry
dict['area'] = "Hyderabad"; # Add new entry
print("dict['Age']: ", dict['Age'])
print("dict['area']: ", dict['area'])
```

OUTPUT

```
dict['age']:  8
dict['area']:  Hyderabad
```

Dictionary

Delete Dictionary Elements

```
dict = {'Name': 'machine learning', 'Age': 1}
del dict['Name']; # remove entry with key 'Name'
dict.clear();     # remove all entries in dict
del dict ;        # delete entire dictionary
```

Built-in Dictionary Functions & Methods

1) cmp()	cmp(dict1,dict2)	returns values 0,-1,1
2) len()	len(dict1)	return length of dict
3) str()	str(dict1)	returns string representation
4) type()	type(dict1)	returns type of variable

Dictionary

1) **clear()** - **dict.clear()** - removes all items from the dict
2) **copy()** - **dict.copy()** - returns a shallow copy of the dict
3) **fromkeys()** - **dict.fromkeys(seq[, value])** - creates a new dictionary with keys from seq and values set to value.
4) **get()** - **dict.get(key, default = None)** - returns a value for the given key. If key is not available then returns default value None.
5) **has_key()** - **dict.has_key(key)** - returns true if a given key is available in the dictionary, otherwise it returns a false.
6) **items()** - **dict.items()** - returns a list of dict's (key, value) tuple pairs
7) **keys()** - **dict.keys()** - returns a list of all the available keys in the dictionary.
8) **setdefault()** - **dict.setdefault(key, default=None)** - is similar to get(), but will set dict[key]=default if key is not already in dict.
9) **update()** - **dict.update(dict2)** - adds dictionary dict2's key-values pairs in to dict. This function does not return anything.
10) **values()** - **dict.values()** - returns a list of all the values available in a given dictionary.

Python Concepts

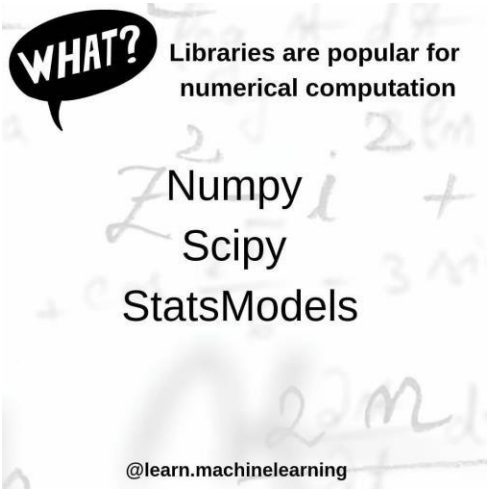
Blogs to read:

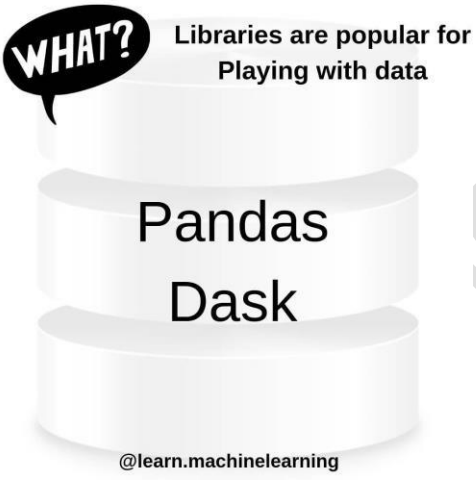
1. https://www.learnpython.org/
2. https://www.codecademy.com/learn/learn-python
3. https://www.python.org/
4. https://www.tutorialspoint.com/python/
5. https://www.sololearn.com/Course/Python/

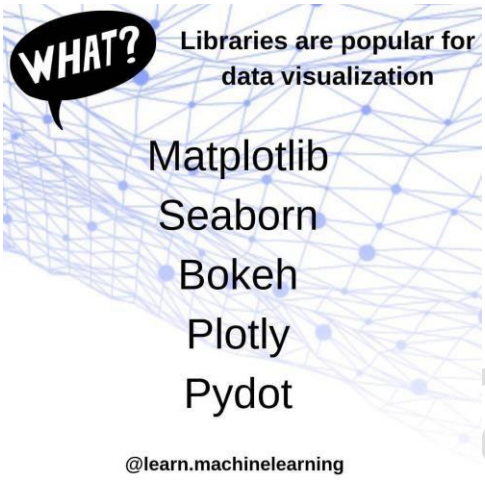
Videos to watch:

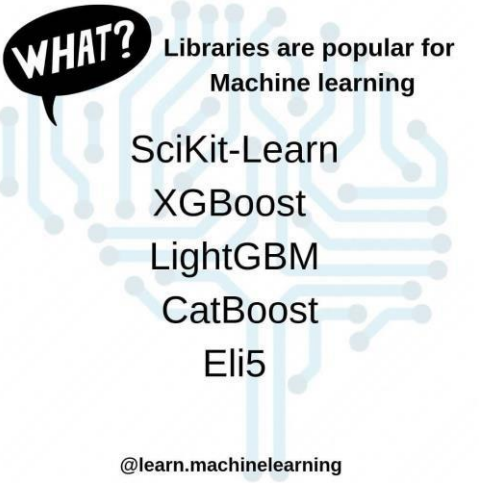
1. Treehouse
2. LearnCode.academy
3. Derek Banas
4. thenewboston

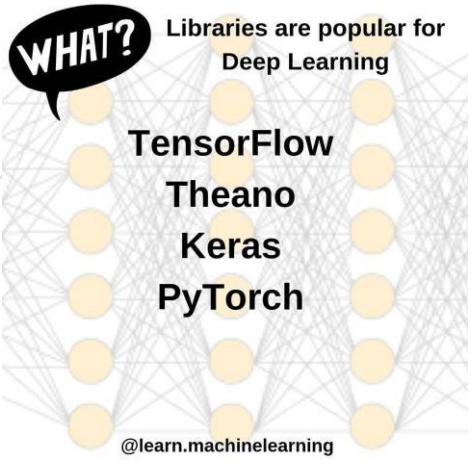
Most popular Libraries

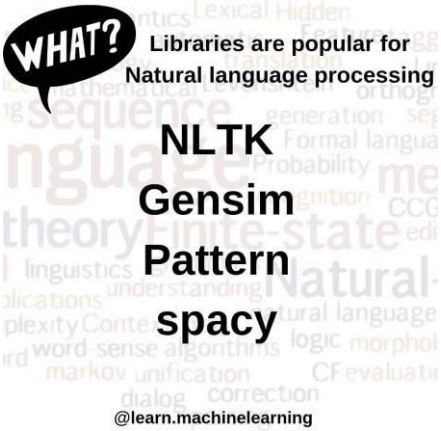












Most popular Libraries

Blogs to read:

1. Top 10 Python Libraries You Must Know In 2019
2. 17 Best Python Libraries
3. Python libraries and packages for Data Scientists
4. Top 20 Python libraries for Data Science

Videos to Watch:

1. No videos

NUMPY

Blogs to read:

1. http://cs231n.github.io/python-numpy-tutorial/
2. Numpy Tutorial Part 1
3. Python NumPy Tutorial
4. Datacamp

Videos to watch:

1. Python NumPy Tutorial
2. Complete Python NumPy Tutorial
3. NumPy Python Tutorial 2018 - Part 1
4. NumPy Python Tutorial 2018 Part 2

PANDAS

Blogs to read:

1. Pandas Tutorial: DataFrames in Python
2. Python Pandas Tutorial
3. Pandas Basics
4. Python Pandas Tutorial: Dataframe, Date Range, Slice

Videos to watch:

1. Pandas Tutorial (Data Analysis In Python)
2. Data analysis in Python with pandas
3. Python Pandas Tutorial - Data Analysis in Python

MATPLOTLIB

Blogs to read:

1. Matplotlib Tutorial: Python Plotting
2. Matplotlib Tutorial: Learn basics
3. Matplotlib Tutorial – Python Matplotlib Library
4. Matplotlib Crash Course Python Tutorial

Videos to watch:

1. Matplotlib Tutorials
2. Matplotlib tutorial1
3. Matplotlib Plotting Tutorials
4. Data Visualization with Matplotlib for beginners

SCIKIT-LEARN

Blogs to read:

1. Python Machine Learning: Scikit-Learn Tutorial
2. Scikit-Learn Tutorial: Machine Learning in Python Examples
3. The Data Scientist's Toolbox Tutorial
4. scikit-learn Documentation

Videos to watch:

1. Machine learning in Python with scikit-learn
2. Scikit-learn Machine Learning with Python and SKlearn
3. Machine Learning with Scikit-Learn
4. scikit-learn tutorials

TENSORFLOW

Blogs to read:

1. Tensorflow Documentation
2. TensorFlow Tutorial: Deep Learning for Beginner's
3. TensorFlow Tutorial For Beginners
4. MIT Deep Learning Basics: Introduction and Overview with TensorFlow

Videos to watch:

1. Basics of TensorFlow - TF Workshop - Session 1
2. TensorFlow Tutorials
3. Tensorflow/TF Tutorial by Sentdex
4. Deep Learning with Tensorflow

Free Courses:

1. Intro to TensorFlow
2. Intro to TensorFlow for Deep Learning
3. Deep Learning with Tensorflow

KERAS

Blogs to read:

1. Keras Tutorial for Beginners with Python: Deep Learning EXAMPLE
2. Keras Documentation
3. Keras Tutorial: The Ultimate Beginner's Guide
4. Keras Tutorial: Deep Learning in Python
5. Keras Tutorial: Develop Your First Neural Network in Python Step-By-Step

Videos to watch:

1. Keras - Python Deep Learning Neural Network API
2. Deep Learning with Keras - Python
3. Keras Tutorials

Free courses:

1. Deep Learning Fundamentals with Keras
2. Advanced Deep Learning with Keras in Python

PYTORCH

Blogs to read:

1. Pytorch Documentation
2. An Introduction to PyTorch
3. A PyTorch tutorial – deep learning in Python

Videos to watch:

1. Neural Network Programming - Deep Learning with PyTorch
2. PyTorch Tutorial: Zero to GANs

Free Courses:

1. Intro to Deep Learning with PyTorch
2. Deep Learning with Python and PyTorch

NLTK

Blogs to read:

1. NLTK Documentation
2. NLTK Python Tutorial (Natural Language Toolkit)
3. NLTK (Natural Language Toolkit) Tutorial in Python

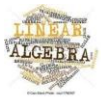
Videos to watch:

1. NLTK with Python 3 for Natural Language Processing
2. NLTK Text Processing Tutorial Series

Free courses:

1. Natural Language Processing Fundamentals in Python
2. Natural Language Processing (NLP)

WHY Linear Algebra???



Why Linear Algebra?

- Linear Algebra is a branch of mathematics that lets you concisely describe coordinates and interactions of planes in higher dimensions and perform operations on them.
- Broadly speaking, in linear algebra data is represented in the form of linear equations. These linear equations are in turn represented in the form of matrices and vectors.
- As a field, it's useful to you because you can describe (and even execute with the right libraries) complex operations used in machine learning using the notation and formalisms from linear algebra.

Why Linear Algebra?

- Linear algebra finds widespread application because it generally parallelizes extremely well. Further to that most linear algebra operations can be implemented without messaging passing which makes them amenable to MapReduce implementations.
- Machine Learning deals with the handling of enormous data sets. And an effective way to represent this data is in the form of 2D arrays or rectangular blocks in which each row represents a sample or a complete record and a column represents a feature or an attribute. It is natural to think of the array as a matrix and each column (attribute) as a vector.

Why Linear Algebra?

- If you want to get into the theory of it all, you need to know linear algebra. If you want to read white papers and consider cutting edge new algorithms and systems, you need to know a lot of math.
- The concepts of Linear Algebra are crucial for understanding the theory behind Machine Learning, especially for Deep Learning. They give you better intuition for how algorithms really work under the hood, which enables you to make better decisions. So if you really want to be a professional in this field, you cannot escape mastering some of its concepts.

Linear Algebra Syllubus

Syllabus

- Scalar
- Vector
- Matrix
- Tensor
- Matrix operations
- Dot product
- Angle between 2 vectors
- Unit vector

- Projections
- Equation of a line, plane, hyperplane, circle, shpere, hypersphere, ellipse, ellipsoid, hyperellipsoid
- Distance between points
- PCA (pricipal component analysis)

- Linear Least Squares
- SVD (singular value decomposition)
- Eigen decomposition
- Eigen values & eigen vectors
- Matrix factorization
- Matrix Decomposition
- LU Decomposition

- Matrix Vector Multiplication
- Matrix Matrix Multiplication
- Matrix Multiplication Properties
- Orthogonalization
- Orthonormalization
- QR Decomposition

Linear Algebra

Scalar

Scalar is a single number. Usually, we write scalars in italic and lowercase, such as "x". Scalar could be any type of number, for example

1. number
2. Rational number
3. Irrational number

Code in numpy

```
import numpy as np
x = np.array(1.666666)
```

@learn.machinelearning

vector

Vector A vector is a tuple of one or more values called scalars(array of numbers.). Typically, we denote vectors with lowercase letter, and put elements of a vector as a column enclosed in square brackets

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

Code in numpy

```
import numpy as np
x = np.array([1,2,3,4,5,6,7,8,9])
```

@learn.machinelearning

Matrix

Matrix is a 2-D array of numbers, which usually denoted with bold and uppercase variable names, such as:

$$\boldsymbol{X} = \begin{bmatrix} X_{1,1} & X_{1,2} & \dots & X_{1,n} \\ X_{2,1} & X_{2,2} & \dots & X_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ X_{m,1} & X_{m,2} & \dots & X_{m,n} \end{bmatrix}$$

Code in numpy

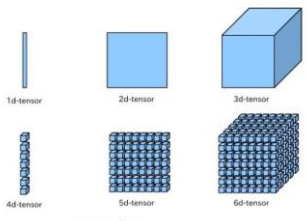
```
import numpy as np
x = np.array([[1,2,3,4],[5,6,7,8],
[9,10,11,12],[13,14,15,16]])
```

@learn.machinelearning

Tensor

Tensor is a k-D array of numbers. The concept of tensor is a bit tricky. You can consider tensor as a container of numbers, and the container could be in any dimension. For example, scalars, vectors, and matrices, are considered as the simplest tensors:

1. Scalar is a 0-dimensional tensor
2. Vector is a 1-dimensional tensor
3. Matrix is a 2-dimensional tensor



source medium.com

@learn.machinelearning

Tensor

commonly, we store time series data in 3-D tensor, image data in 4-D tensor, video data in 5-D tensor, etc.

Code in numpy

```
import numpy as np

x = np.array([[[1, 2, 3],[4, 5, 6],[7, 8, 9]],
 [[10, 20, 30],[40, 50, 60],[70, 80, 90]],
 [[100, 200, 300],[400, 500, 600],[700, 800, 900]]])
```

@learn.machinelearning

Linear Algebra

Matrix operations

Scalar+Matrix: add the scalar to each element in the matrix

$$\mathbf{C} = \mathbf{A} + b, \text{ where } C_{i,j} = A_{i,j} + b$$

$$\mathbf{C} = \mathbf{A} + b = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} + 3.14 = \begin{bmatrix} 4.14 & 5.14 \\ 6.14 & 7.14 \end{bmatrix}$$

@learn.machinelearning

Matrix operations

Vector + Matrix : add the vector to each row in the matrix

$$\mathbf{C} = \mathbf{A} + \boldsymbol{v}, \text{ where } C_{i,j} = A_{i,j} + v_j$$

$$\mathbf{C} = \mathbf{A} + \boldsymbol{v} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} + \begin{bmatrix} 7 \\ 8 \\ 9 \end{bmatrix} = \begin{bmatrix} 8 & 10 & 12 \\ 11 & 13 & 15 \end{bmatrix}$$

@learn.machinelearning

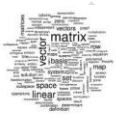
Matrix operations

Matrix + Matrix: add their corresponding elements as long as the matrices have the same shape

$$\mathbf{C} = \mathbf{A} + \mathbf{B}, \text{ where } C_{i,j} = A_{i,j} + B_{i,j}$$

$$\mathbf{C} = \mathbf{A} + \mathbf{B} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} + \begin{bmatrix} 7 & 8 & 9 \\ 10 & 11 & 12 \end{bmatrix} = \begin{bmatrix} 8 & 10 & 12 \\ 14 & 16 & 18 \end{bmatrix}$$

@learn.machinelearning



Matrix operations

Scalar • Matrix: each element of the matrix multiply the scalar

$$\mathbf{C} = \mathbf{A} \cdot b, \text{ where } C_{i,j} = A_{i,j} \cdot b$$

$$\mathbf{C} = \mathbf{A} \cdot b = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \cdot 5 = \begin{bmatrix} 5 & 10 \\ 15 & 20 \end{bmatrix}$$

@learn.machinelearning

Matrix operations

Matrix • Vector: dot product of each row in the matrix and the vector

$$\mathbf{A} \cdot \boldsymbol{v} = \boldsymbol{b}, \text{ where } A_{i,:} \cdot \mathbf{v} = b_i$$

$$\mathbf{A} \cdot \boldsymbol{v} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \cdot \begin{bmatrix} 7 \\ 8 \\ 9 \end{bmatrix} = \begin{bmatrix} 1 \cdot 7 + 2 \cdot 8 + 3 \cdot 9 \\ 4 \cdot 7 + 5 \cdot 8 + 6 \cdot 9 \end{bmatrix} = \begin{bmatrix} 50 \\ 122 \end{bmatrix}$$

@learn.machinelearning

Matrix operations

Matrix-Matrix Multiplicatio

This is a more complex operation than matrix addition because it does not simply involve multiplying the matrices element-wise. Instead a more complex procedure is utilised, for each element, involving an entire row of one matrix and an entire column of the other.

AB≠BA

A **B** **A * B**

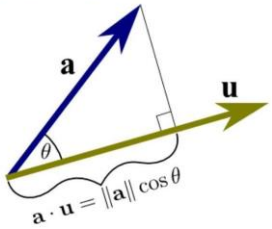
$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \begin{pmatrix} 6 & 3 \\ 5 & 2 \\ 4 & 1 \end{pmatrix} = \begin{pmatrix} 1\cdot6+2\cdot5+3\cdot4 & 1\cdot3+2\cdot2+3\cdot1 \\ 4\cdot6+5\cdot5+6\cdot4 & 4\cdot3+5\cdot2+6\cdot1 \end{pmatrix}$$

@learn.machinelearning

Linear Algebra

Projections

One important use of dot products is in projections. The scalar projection of a onto u is the length of the segment AB shown in the figure below. The vector projection of a onto u is the vector with this length that begins at the origin point in the same direction (or opposite direction if the scalar projection is negative) as a.



@learn.machinelearning

Matrix Factorization

- Matrix decompositions are also called as Matrix Factorization it is a method that reduces a matrix into sub-matrices.
- It is same as writing factors to a numbers example we can write 15 as 5X3.
- It is easy to perform complex matrix operations on the decomposed matrix rather than on the original matrix itself.
- Two simple and widely used matrix decomposition methods are the LU matrix decomposition and the QR matrix decomposition

Applications

- Background Removal
- Topic Modeling
- Recommendations Systems
- Eigen Faces
- PCA (Principle component analysis
- Non-negative matrix factorization
- SVD (singular value decomposition)

If you know any matrix factorization application please mention them in the comments.

@learn.machinelearning

LU decomposition

- The LU decomposition is for square matrices and decomposes a matrix into L and U components.

$$S = L \cdot U$$

- Where S is the square matrix that we wish to decompose, L is the lower triangle matrix and U is the upper triangle matrix.
- The factors L and U are triangular matrices. The factorization that comes from elimination is S = LU.
- LU decomposition is found using an iterative process and can fail for those matrices that cannot be decomposed.
- simplify the solving of systems of linear equations, such as finding the coefficients in linear regression, as well as in calculating the determinant and inverse of a matrix.

@learn.machinelearning

LU decomposition

- Example (code)

```
import  numpy as np
from scipy.linalg import lu

INPUT
A = array([[1, 2, 3], [4, 5, 6],
[7, 8, 9]])
print(A)
# LU decomposition
L, U = lu(A)
print(L)
print(U)
```

OUTPUT

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]

[[ 1.         0.         0.        ]
 [ 0.14285714  1.         0.        ]
 [ 0.57142857  0.5        1.        ]]

[[ 7.00000000e+00  8.00000000e+00  9.00000000e+00]
 [ 0.00000000e+00  8.57142857e-01  1.71428571e+00]
 [ 0.00000000e+00  0.00000000e+00 -1.58603289e-16]]
```

@learn.machinelearning

QR decomposition

- The QR decomposition is for m x n matrices (not limited to square matrices) and decomposes a matrix into Q and R components.

$$S = Q \cdot R$$

- Where S is the matrix that we wish to decompose, Q a matrix with the size m x m, and R is an upper triangular matrix with the size m x n.
- QR decomposition is found using an iterative process and can fail for those matrices that cannot be decomposed.
- simplify the solving of systems of linear equations, such as finding the coefficients in linear regression, as well as in calculating the determinant and inverse of a matrix.

@learn.machinelearning

QR decomposition

Example (Code)

```
import numpy as np
from numpy.linalg import qr

INPUT
A = np.array([[1, 2], [3, 4], [5, 6]])
print(A)
# QR decomposition
Q, R = qr(A, 'complete')
print(Q)
print(R)
```

OUTPUT

```
[[1 2]
 [3 4]
 [5 6]]

[[-0.16903085  0.89708523  0.40824829]
 [-0.50709255  0.27602622 -0.81649658]
 [-0.84515425 -0.34503278  0.40824829]]

[[-5.91607978 -7.43735744]
 [ 0.          0.82807867]
 [ 0.          0.        ]]
```

@learn.machinelearning

Linear Algebra

Eigenvectors

Eigenvectors are unit vectors, which means that their length or magnitude is equal to 1.0. They are often referred as right vectors, which simply means a column vector (as opposed to a row vector or a left vector). A right-vector is a vector as we understand them.

@learn.machinelearning

Eigenvalues

- Eigenvalues are coefficients applied to eigenvectors that give the vectors their length or magnitude. For example, a negative eigenvalue may reverse the direction of the eigenvector as part of scaling it.
- A matrix that has only positive eigenvalues is referred to as a positive definite matrix, whereas if the eigenvalues are all negative, it is referred to as a negative definite matrix.
- Decomposing a matrix in terms of its eigenvalues and its eigenvectors gives valuable insights into the properties of the matrix.

@learn.machinelearning

Eigendecomposition

- Eigendecomposition of a matrix is a type of decomposition that involves decomposing a square matrix into a set of eigenvectors and eigenvalues.
- A . v = lambda . v
- where A is the parent square matrix that we are decomposing, v is the eigenvector of the matrix, and lambda is the lowercase Greek letter and represents the eigenvalue scalar.
- Eigendecomposition can also be used to calculate the principal components of a matrix in the Principal Component Analysis method or PCA that can be used to reduce the dimensionality of data in machine learning.

@learn.machinelearning

Eigendecomposition

Example (code)

INPUT

```
# eigendecomposition
import numpy as np
from numpy.linalg import eig
# define matrix
A = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
print(A)
# calculate eigendecomposition
values, vectors = eig(A)
print(values)
print(vectors)
```

OUTPUT

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]

[ 1.61168440e+01 -1.11684397e+00 -9.75918483e-16]

[[-0.23197069 -0.78583024  0.40824829]
 [-0.52532209 -0.08675134 -0.81649658]
 [-0.8186735   0.61232756  0.40824829]]
```

@learn.machinelearning

Linear Algebra

Blogs to read:

1. Math only Math
2. Math - Love
3. Math Stack Exchange
4. Interactive Mathematics
5. Math with bad drawings

Videos to Watch:

1. MIT linear algebra course
2. Khan academy linear algebra

Free courses:

1. EDX- linear algebra
2. Coursera Linear algebra

Why Probability and Statistics??

why probability and statistics important???

There are two types of methods in statistics.
1. inferential statistical methods
2. descriptive statistical methods

we can use descriptive statistical methods to transform raw observations into information that you can understand and share. we can use inferential statistical methods to reason from small samples of data to whole domains.

@learn.machinelearning

why probability and statistics important???

statisticians refer to machine learning as "applied statistics" or "statistical learning" rather than the computer-science-centric name

Statistics is a subfield of mathematics. It refers to a collection of methods for working with data and using data to answer questions.

@learn.machinelearning

why probability and statistics important???

Descriptive statistics refer to methods for summarizing raw observations into information that we can understand and share.

Inferential statistics is a fancy name for methods that aid in quantifying properties of the domain or population from a smaller set of obtained observations called a sample.

@learn.machinelearning

why probability and statistics important???

Probability is the chance that something will happen—how likely it is that some event will happen.
Probability of an event happening P(E) = Number of ways it can happen n(E)/ Total number of outcomes n(T).
Probability is the measure of the likelihood that an event will occur. Probability is quantified as a number between 0 and 1, where 0 indicates impossibility and 1 indicates certainty.

@learn.machinelearning

why probability and statistics important???

Data science often uses statistical inferences to predict or analyze trends from data, while statistical inferences uses probability distributions of data. Hence knowing probability and its applications are important to work effectively on data science problems.

@learn.machinelearning

Syllabus

syllabus

1. Probability density function (PDF)
2. Cumulative distribution function (CDF)
3. Mean, Median, Mode
4. Variance, Std-dev
5. Percentiles, Quantiles
6. Inter quantile range (IQR)

syllabus

7. Median absolute deviation
8. Outliers
9. Univariate, Bivariate & Multivariate analysis
10. Symmetric distribution
11. Skewness
12. kurtosis

syllabus

13. Standard normal variate
14. Standardization
15. Normalization
16. Kernel density estimation (KDE)
17. Sampling distribution
18. Central limit theorem

syllabus

19. Q-Q plot
20. Bernoulli distribution
21. Binomial distribution
22. Lognormal distribution
23. power law distribution
24. Co-variance
25. Pearson correlation coefficient

syllabus

26. Spearman Rank correlation coefficient
27. Correlation Vs causation
28. Confidence intervals
29. Bootstraping
30. resampling
31. T-Distribution

syllabus

32. Hypothesis testing
33. Permutation test
34. K-S test
35. Analysis of variance (ANOVA)
36. Conditional probability
37. Basics of probability

Probability and Statistics `

Mean

The mean is the average of the numbers. It is easy to calculate: add up all the numbers, then divide by how many numbers there are. In other words it is the sum divided by the count.

Example : What is the Mean of these numbers? 6, 11, 7

Add the numbers: 6 + 11 + 7 = 24
Divide by how many numbers (there are 3 numbers): 24 / 3 = 8
The Mean is 8

Median

The Median is the "middle" of a sorted list of numbers. To find the Median, place the numbers in value orderand find the middle.

$$MED + IAN \over 2$$

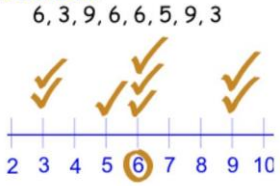
Example: find the Median of 12, 3 and 5
Put them in order:
3, 5, 12
The middle is 5, so the median is 5.

Mode

The mode is simply the number which appears most often.We can have more than one mode. Having two modes is called "bimodal". Having more than two modes is called "multimodal"

Example : 6, 3, 9, 6, 6, 5, 9, 3.

In {6, 3, 9, 6, 6, 5, 9, 3} the Mode is 6, as it occurs most often.



variance

The Variance is defined as: The average of the squared differences from the Mean. To calculate the variance follow these steps:

Work out the Mean (the simple average of the numbers)
Then for each number: subtract the Mean and square the result (the squared difference).
Then work out the average of those squared differences.

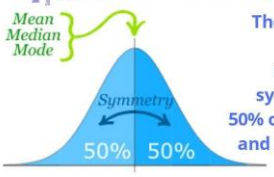
Standard Deviation

The Standard Deviation is a measure of how spread out numbers are.
Its symbol is σ (the greek letter sigma)
The formula is easy: it is the square root of the Variance.

$$\sigma = \sqrt{\frac{\sum (x - \bar{x})^2}{n}}$$

σ = lower case sigma
$\sum$ = capital sigma
$\bar{x}$ = x bar

Normal distribution



The Normal Distribution has

mean = median = mode
symmetry about the center
50% of values less than the mean and 50% greater than the mean

Data can be "distributed" (spread out) in different ways.But there are many cases where the data tends to be around a central value with no bias left or right, and it gets close to a "Normal Distribution" like this.

It is often called a "Bell Curve" because it looks like a bell. Many things closely follow a Normal Distribution: heights of people,size of things produced by machines errors in measurements, blood pressure, marks on a test

Probability and Statistics `

continuous random variable

continuous random variable is a random variable where the data can take infinitely many values. For example, a random variable measuring the time taken for something to be done is continuous since there are an infinite number of possible times that can be taken.

Probability density function

Probability density function (PDF) is a statistical expression that defines a probability distribution for a continuous random variableas opposed to a discrete random variable. When the PDF is graphically portrayed, the area under the curve will indicate the interval in which the variable will fall. The total area in this interval of the graph equals the probability of a continuous random variable occurring.

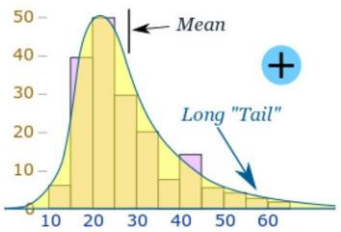
Probability density function

Probability density function (PDF) is a statistical expression that defines a probability distribution for a continuous random variableas opposed to a discrete random variable. When the PDF is graphically portrayed, the area under the curve will indicate the interval in which the variable will fall. The total area in this interval of the graph equals the probability of a continuous random variable occurring.

@learn.machinelearning t.me/machinelearningfromscratch

Skewed Data

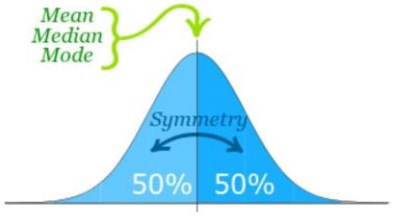
Positive Skew



- positive skew is when the long tail is on the positive side of the peak, and some people say it is "skewed to the right".
- The mean is on the right of the peak value.

Skewed Data

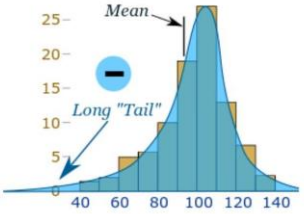
Normal Distribution



- A Normal Distribution is not skewed.
- It is perfectly symmetrical.
- And the Mean is exactly at the peak.

Skewed Data

Negative Skew

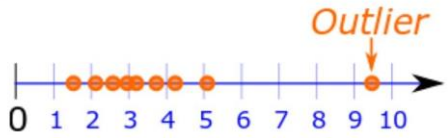


- Why is it called negative skew? Because the long "tail" is on the negative side of the peak.
- People sometimes say it is "skewed to the left" (the long tail is on the left hand side)
- The mean is also on the left of the peak.

Probability and Statistics `

Outlier

A value that "lies outside" (is much smaller or larger than) most of the other values in a set of data.



For example in the scores 1,2,3,4,1,2,3,5,3,4,2 both 9 and 10 are "outliers".

Median absolute deviation

median absolute deviation(MAD) is a robust measure of how spread out a set of data is. The variance and standard deviation are also measures of spread, but they are more affected by extremely high or extremely low values and non normality.

Step 1: Calculate the Median.
Step 2: Calculate how far away each data point is from the Median using positive distances. These are called absolute deviations.
Step 3: find the median of the absolute differences.

Quartiles

Quartiles are the values that divide a list of numbers into quarters:
1. Put the list of numbers in order
2. Then cut the list into four equal parts
3. The Quartiles are at the "cuts"

Example: 5, 7, 4, 4, 6, 2, 8
Put them in order: 2, 4, 4, 5, 6, 7, 8

And the result is:
Quartile 1 (Q1) = 4
Quartile 2 (Q2), which is also the Median, = 5
Quartile 3 (Q3) = 7

Percentiles

The value below which a percentage of data falls.

Example: You are the fourth tallest person in a group of 20
80% of people are shorter than you:

That means you are at the 80th percentile.
If your height is 1.85m then "1.85m" is the 80th percentile height in that group.

Box and Whisker Plot

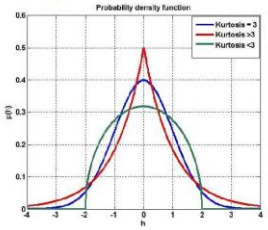
We can show all the important values in a "Box and Whisker Plot", like this:

Example: Box and Whisker Plot
4, 17, 7, 14, 18, 12, 3, 16, 10, 4, 4, 11
Put them in order:
3, 4, 4, 4, 7, 10, 11, 12, 14, 16, 17, 18
Cut it into quarters:
3, 4, 4 | 4, 7, 10 | 11, 12, 14 | 16, 17, 18
In this case all the quartiles are between numbers:
Quartile 1 (Q1) = (4+4)/2 = 4
Quartile 2 (Q2) = (10+11)/2 = 10.5
Quartile 3 (Q3) = (14+16)/2 = 15

Also:
The Lowest Value is 3,
The Highest Value is 18

kurtosis

A positive value tells you that you have heavy-tails (i.e. a lot of data in your tails).
A negative value means that you have light-tails (i.e. little data in your tails).
This heaviness or lightness in the tails usually means that your data looks flatter (or less flat) compared to the normal distribution. The standard normal distribution has a kurtosis of 3, so if your values are close to that then your graph's tails are nearly normal.



Probability and Statistics `

Univariate Analysis

Univariate analysis is the simplest form of data analysis where the data being analyzed contains only one variable. Since it's a single variable it doesn't deal with causes or relationships. The main purpose of univariate analysis is to describe the data and find patterns that exist within it

You can think of the variable as a category that your data falls into.

@learn.machinelearning

Bivariate Analysis

Bivariate analysis is used to find out if there is a relationship between two different variables. Something as simple as creating a scat terplot by plotting one variable against another on a Cartesian plane (think X and Y axis) can sometimes give you a picture of what the data is trying to tell you. If the data seems to fit a line or curve then there is a relationship or correlation between the two variables

@learn.machinelearning

Multivariate Analysis

Multivariate analysis is the analysis of three or more variables. There are many ways to perform multivariate analysis depending on your goals. Some of these methods include Additive Tree, Canonical Correlation Analysis, Cluster Analysis, Correspondence Analysis / Multiple Correspondence Analysis, Factor Analysis, Generalized ProcrusteanAnalysis, MANOVA, Multidimensio nal Scaling, Multiple Regression Analysis, Partial Least Square Regression, Principal Component Analysis / Regression / PARAFAC, and Redundancy Analysis.

@learn.machinelearning

Why scaling???

Most of the times, your dataset will contain features highly varying in magnitudes, units and range. But since, most of the machine learning algorithms use Eucledian distance between two data points in their computations, this is a problem.

If left alone, these algorithms only take in the magnitude of features neglecting the units. The results would vary greatly between different units, 5kg and 5000gms. The features with high magnitudes will weigh in a lot more in the distance calculations than features with low magnitudes

@learn.machinelearning

When scaling???

Any algorithm that computes distance or assumes normality, scale your features.
Some examples of algorithms where feature scaling matters are:

1. KNN
2. PCA
3. Graduent descent
4. Tree models
5. LDA

@learn.machinelearning

How scaling??

There are four common methods to perform Feature Scaling.

1.Standardisation
2.Mean Normalisation
3.Min-Max Scaling
4.Unit Vector

@learn.machinelearning

Probability and Statistics `

standerdization

Data standardization is the process of rescaling one or more attributes so that they have a mean value of 0 and a standard deviation of 1.

Standardization assumes that your data has a Gaussian (bell curve) distribution. This does not strictly have to be true, but the technique is more effective if your attribute distribution is Gaussian.

@learn.machinelearning

Normalization

Data normalization is the process of rescaling one or more attributes to the range of 0 to 1. This means that the largest value for each attribute is 1 and the smallest value is 0.

Normalization is a good technique to use when you do not know the distribution of your data or when you know the distribution is not Gaussian (a bell curve).

@learn.machinelearning

Min-Max Scaling

This scaling brings the value between 0 and 1.

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

@learn.machinelearning

standerdization

Standardisation replaces the values by their Z scores.

$$x' = \frac{x - \bar{x}}{\sigma}$$

This redistributes the features with their mean μ = 0 and standard deviation σ =1 . sklearn.preprocessing.scale helps us implementing standardisation in python.

@learn.machinelearning

Normalization

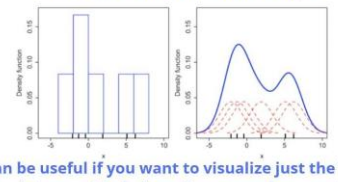
This distribution will have values between -1 and 1with μ=0. Standardisation and Mean Normalization can be used for algorithms that assumes zero centric data like Principal Component Analysis(PCA).

$$x' = \frac{x - \text{mean}(x)}{\max(x) - \min(x)}$$

@learn.machinelearning

kernel density estimation

kernel density estimation (KDE) is a non-parametric way to estimate the probability density function of a random variable. Kernel density estimation is a fundamental data smoothing problem where inferences about the population are made, based on a finite data sample.



This can be useful if you want to visualize just the "shape" of some data, as a kind of continuous replacement for the discrete histogram. It can also be used to generate points that look like they came from a certain dataset - this behavior can power simple simulations, where simulated objects are modeled off of real data.

@learn.machinelearning

Probability and Statistics `

kernel density estimation

The KDE algorithm takes a parameter, bandwidth, that affects how "smooth" the resulting curve is. Use the control below to modify bandwidth, and notice how the estimate changes.The KDE is calculated by weighting the distances of all the data points Changing the bandwidth changes the shape of the kernel: a lower bandwidth means only points very close to the current position are given any weight, which leads to the estimate looking squiggly; a higher bandwidth means a shallow kernel where distant points can contribute.

@learn.machinelearning

central limit theorem

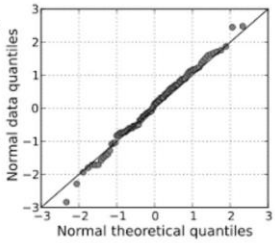
Suppose we draw a random sample of size n (x1, x2, x3, ... xn—1, xn) from a population random variable that is distributed with mean μ and standard deviation σ. Do this repeatedly, drawing many samples from the population, and.then calculate the $\bar{x}$ of each sample. We will treat the $\bar{x}$ values as another distribution, which we will call the sampling distribution of the mean ($\bar{x}$).central limit theorem is that no matter what the shape of the original (parent) distribution, the sampling distribution of the mean approaches a normal distribution.
A normal distribution is approached very quickly as n increases, Note that n is the sample size for each mean and not the number of samples
Remember in a sampling distribution of the mean the number of samples is assumed to be infinite.

@learn.machinelearning

Q--Q Plot

The quantile-quantile (q-q) plot is a graphical technique for determining if two data sets come from populations with a common distribution.
A q-q plot is a plot of the quantiles of the first data set against the quantiles of the second data set. By a quantile, we mean the fraction (or percent) of points below the given value.

A Q-Q plot is a scatterplot created by plotting two sets of quantiles against one another. If both sets of quantiles came from the same distribution, we should see the points forming a line that's roughly straight



@learn.machinelearning

central limit theorem

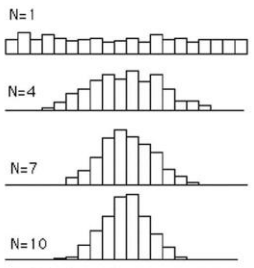
The central limit theorem states that when samples from a data set with a known variance are aggregated their mean roughly equals the population mean. Said another way, CLT is a statistical theory that states that given a sufficiently large sample size from a population with a finite level of variance, the mean of all samples from the same population will be approximately equal to the mean of the popu lation. Furthermore, all the samples will follow an approximate normal distributionpattern, with all variances being approximately equal to the variance of the population divided by each sample's size.

@learn.machinelearning

central limit theorem

Example
On the right are shown the resulting frequency distributions each based on 500 means. For n = 4, 4 scores were sampled from a uniform distribution 500 times and the mean computed each time. The same method was followed with means of 7 scores for n = 7 and 10 scores for n = 10.

When n increases
1. the distributions becomes more and more normal.
2. the spread of the distributions decreases.
Thumb rule is n=30



@learn.machinelearning

Advantages of Q--Q Plot

- The sample sizes do not need to be equal.
- Many distributional aspects can be simultaneously tested. For example, shifts in location, shifts in scale, changes in symmetry, and the presence of outliers can all be detected from this plot. For example, if the two data sets come from populations whose distributions differ only by a shift in location, the points should lie along a straight line that is displaced either up or down from the 45-degree reference line.

The q-q plot is similar to a probability plot. For a probability plot, the quantiles for one of the data samples are replaced with the quantiles of a theoretical distribution.

@learn.machinelearning

Probability and Statistics `

Advantages of Q--Q Plot

The q-q plot is used to answer the following questions:

- Do two data sets come from populations with a common distribution?
- Do two data sets have common location and scale?
- Do two data sets have similar distributional shapes?
- Do two data sets have similar tail behavior?

How to draw Q Q Plot

- Sort the values which we got from the population.
- Get the quantiles from that sorted values.
- Do the same with the other population values also
- Plot the both quantile values and check whether they are forming the 45 degree straight line. If not they are not from the same distribution.

@learn.machinelearning

Bernoulli distribution

The Bernoulli distribution essentially models a single trial of flipping a weighted coin. It is the probability distribution of a random variable taking on only two values, ("success") "1" and "0" ("failure") with complementary probabilities "P" and "1-P" respectively. The Bernoulli distribution therefore describes events having exactly two outcomes, which are ubiquitous in real life. Some examples of such events are as follows: a team will win a championship or not, a student will pass or fail an exam, and a rolled dice will either show a 6 or any other number.

$$P(X = x) = \begin{cases} p \ for \ x = 1 \\ 1 - p \ for \ x = 0 \end{cases}$$

@learn.machinelearning

Binomial Distribution

The binomial distribution is, in essence, the probability distribution of the number of heads resulting from flipping a weighted coin multiple times. It is useful for analyzing the results of repeated independent trials, especially the probability of meeting a particular threshold given a specific error rate, and thus has applications to risk management. For this reason, the binomial distribution is also important in determining statistical significance.

A Bernoulli trial, or Bernoulli experiment, is an experiment satisfying two key properties:

- There are exactly two complementary outcomes, success and failure.
- The probability of success is the same every time the experiment is repeated.

@learn.machinelearning

Binomial Distribution

A binomial experiment is a series of "n" Bernoulli trials, whose outcomes are independent of each other. A random variable,"X" , is defined as the number of successes in a binomial experiment. Finally, a binomial distribution is the probability distribution of "X".

For example, consider a fair coin. Flipping the coin once is a Bernoulli trial, since there are exactly two complementary outcomes (flipping a head and flipping a tail), and they are both 1/2 no matter how many times the coin is flipped. Note that the fact that the coin is fair is not necessary; flipping a weighted coin is still a Bernoulli trial.

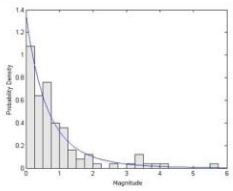
A binomial experiment might consist of flipping the coin 100 times, with the resulting number of heads being represented by the random variable "X". The binomial distribution of this experiment is the probability distribution of "X".

@learn.machinelearning

Power law Distribution

The power law (also called the scaling law) states that a relative change in one quantity results in a proportional relative change in another. The simplest example of the law in action is a square; if you double the length of a side (say, from 2 to 4 inches) then the area will quadruple (from 4 to 16 inches squared).

The power law can be used to describe a phenomenon where a small number of items is clustered at the top of a distribution (or at the bottom), taking up 95% of the resources. In other words, it implies a small amount of occurrences is common, while larger occurrences are rare. For example, where the distribution of income is concerned, there are very few billionaires; the bulk of the population holds very modest nest eggs.



@learn.machinelearning

covariance

covariance is a measure of the relationship between two random variables. The metric evaluates how much – to what extent – the variables change together. In other words, it is essentially a measure of the variance between the two variables. Unlike the correlation coefficient, covariance is measured in units. The units are computed by multiplying the units of the two variables. The variance can take any positive or negative values.

Positive covariance: Indicates that two variables tend to move in the same direction.

Negative covariance: Reveals that two variables tend to move in inverse directions.

@learn.machinelearning

Probability and Statistics `

covariance

covariance formula is similar to the formula for correlation and deals with the calculation of data points from the average value in a dataset. For example, the covariance between two random variables X and Y can be calculated using the following formula

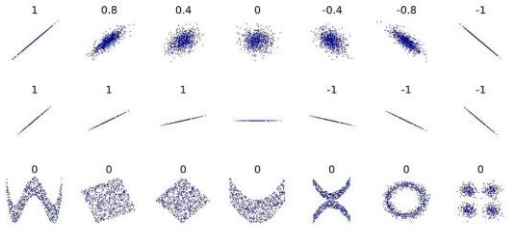
$$\mathrm{Cov}(X,Y) = \frac{\sum(X_i - \bar{X})(Y_j - \bar{Y})}{n}$$

Xi – the values of the X-variable
Yj – the values of the Y-variable
$\bar{X}$ – the mean (average) of the X-variable
$\bar{Y}$ – the mean (average) of the Y-variable
n – the number of the data points

@learn.machinelearning

correlation coefficient

More examples



source - wiki

@learn.machinelearning

Covariance vs. Correlation

The relationship between the two concepts can be expressed using the formula below:

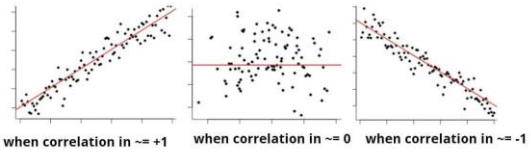
$$\rho(X,Y) = \frac{\mathrm{Cov}(X,Y)}{\sigma_X \sigma_Y}$$

ρ(X,Y) – the correlation between the variables X and Y
Cov(X,Y) – the covariance between the variables X and Y
σX – the standard deviation of the X-variable
σY – the standard deviation of the Y-variable

@learn.machinelearning

correlation

Correlation is a bivariate analysis that measures the strength of association between two variables and the direction of the relationship. In terms of the strength of relationship, the value of the correlation coefficient varies between +1 and -1. A value of ± 1 indicates a perfect degree of association between the two variables. As the correlation coefficient value goes towards 0, the relationship between the two variables will be weaker. The direction of the relationship is indicated by the sign of the coefficient; a + sign indicates a positive relationship and a – sign indicates a negative relationship.



when correlation in ~= +1 | when correlation in ~= 0 | when correlation in ~= -1

@learn.machinelearning

Covariance vs. Correlation

Covariance and correlation both mainly assess the relationship between variables.

Covariance measures the total variation of two random variables from their expected values. Using covariance, we can only gauge the direction of the relationship (whether the variables tend to move in tandem or show an inverse relationship). However, it does not indicate the strength of the relationship, nor the dependency between the variables.

correlation measures the strength of the relationship between variables. Correlation is the scaled measure of covariance. It is dimensionless. In other words, the correlation coefficient is always a pure value and not measured in any units.

@learn.machinelearning

spearman correlation coefficient

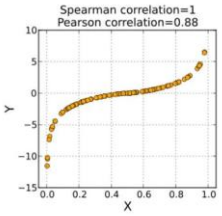
Spearman's rank correlation coefficient or Spearman's rho, named after Charles Spearman and often denoted by the Greek letter (rho). It is the nonparametric version of the Pearson correlation coefficient. Your data mustbe ordinal, interval or ratio. Spearman's returns a value from -1 to 1, where:
+1 = a perfect positive correlation between ranks
-1 = a perfect negative correlation between ranks
0 = no correlation between ranks.

- It assesses how well the relationship between two variables can be described using a monotonic function.
- while Pearson's correlation assesses linear relationships, Spearman's correlation assesses monotonic relationships

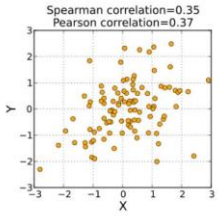
@learn.machinelearning

Probability and Statistics `

spearman correlation coefficient



A Spearman correlation of 1 results when the two variables being compared are monotonically related, even if their relationship is not linear. This means that all data-points with greater x-values than that of a given data-point will have greater y-values as well. In contrast, this does not give a perfect Pearson correlation.



When the data are roughly elliptically distributed and there are no prominent outliers, the Spearman correlation and Pearson correlation give similar values.

@learn.machinelearning

correlation vs causation

- People use these two words interchangeably, Which brings confusion to the people which word we need to use in what time. Lets remove the confusion and understand it properly when to use both
- Correlation refers to the degree of association between two random variables. See my previous posts to learn more about correlation.
- Causation is implying that A and B have a cause-and-effect relationship with one another. You're saying A causes B.

@learn.machinelearning

correlation vs causation

- Lets take an example where we use these two words.
- Suppose when you are using your mobile phone like you open many applications which are running in the background and you are trying to text someone but your mobile is freeze and you are unable to text. What you think of this because of freeze you are unable to text. But immediately you are able to text back. But is that the reason, "NO" because of lack or RAM your mobile is freeze and you are unable to text.
- here correlation is freeze and unable to text. cause is Lack of RAM
- Majority of the cases correlation, are just because of the coincidences. Just because it seems like one factor is influencing the other, it doesn't mean that it's actually does. Here in our example freezing doesn't allowing you to text. But it seems like it is doing that.
- Don't conclude too fast!
- There is a famous quote. "correlation does not imply causation"

@learn.machinelearning

confidence interval

- Lets start with an usecase, where we want to know how many people in India are watching cricket. One thing what we can do is asking everyone and making a note of it to get 100% accuracy. Some people in Quora are stating that at lest 40% of Indian population are watching cricket.
- Its impossible to ask those 40% people to know the result right?? One thing we can do is to take a sample of people like 10000 and ask them whether they watch or not. But we are not 100% sure this will be equal to the whole India. So what we can do is we can give some interval and say i'm 95% sure that the % of people in India watch cricket lies in this interval for example i can say 95% confident that it lies between 38 % to 42%.
- One important thing is we should now collect sample from same city or state. It doesn't represent the whole India

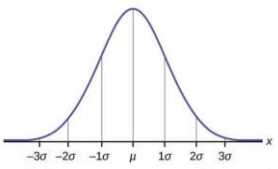
@learn.machinelearning

confidence interval

- So lets take 10000 people randomly from different places in India. And lets say the result we got is 42%. So what we can assume about the whole India.
- Lets take multiple samples and get the average of that samples. Almost every sample mean will be nearly equal to the real average(~=40).
- And then we try to plot all those means and guess what the graph looks very similar to the normal distribution. And we can say from that is 2.5% of the values are below lets say 38% and 2.5% values are above 42%.And mean is very close to the real percentage.
- If we know the standard deviation of the real population they it is easy to find the confidence interval using (mean)+-(standard deviation).
- If we don't know the what???. Swipe>>>>>>

@learn.machinelearning

confidence interval



- Lets look at this graph. From previous we know the mean but not standard deviation. As a thumb rule we can say that ~64% values will be between 1 standard deviation σ and ~95% values lies between 2 standard deviation σ.
- So lets say we got 39.5 as mean fro the last example. And we want to know where the real percentage falss between. So here i can say the 95% confident that the real percentage false between 39.5-2σ and 39.5+2σ.
- So this is the confidence interval. Interval is 39.5-2σ and 39.5+2σ. and confidence is 95%.

@learn.machinelearning

Probability and Statistics `

Bootstrapping

. It is a method which is used to estimate statistical methods on a population by resampling the data.
. And it uses a sampling with replacement technique to make sure the draw sample data every time should be present in population data to see that data again in other samples.
. The two important things in bootstrapping are the size of a sample and how many samples.
. The steps involved are. swipe>>>>>

@learn.machinelearning

Bootstrapping

. Draw a sample from the population.
. Perform statistics om that sample.
. Do that on multiple replacements samples.
. Calculate the mean of all those computed samples.

The two parameters need to take care of our sample size and no of repetitions.

Sample Size - We can take whole data as sample data. where few data is repeated and few are not present. if the dataset is huge and computational expensive then you can use 40% or 50%.

@learn.machinelearning

Bootstrapping

Repetitions - This can be your choice when you are trying to get meaning dull results like mean, standard deviation or variance. The thumb rule is to use between 20-30. if you have more resources and time then you can do more.

Bootstrapping does not make assumptions about the distribution of your data. You merely resample your data and use whatever sampling distribution emerges. Then, you work with that distribution, whatever it might be

@learn.machinelearning

conditional probability

- The conditional probability of event A is given by the probability of A given probability of event B is already completed.
- This is the prerequisite topic in probability where we use this in machine learning

$$P(\mathrm{A} \mid \mathrm{B}) = \frac{P(\mathrm{A \text{ and } B})}{P(\mathrm{B})}$$

- P(A|B) is Probability of event A given B has occurred.
- P(A and B) is probability of event A occurred and event B occurred.
- P(B) is probability of B

@learn.machinelearning

Probability and Statistics

Hypothesis Testing

- Person tests a statistical sample, with the goal of accepting or rejecting a null hypothesis. The test tells the person whether or not his primary hypothesis is true.

Key terms

1. Null hypothesis
2. Alternative hypothesis
3. Level of significance
4. Type I error
5. Type II error
6. One-tailed test
7. Two-tailed test

There are two types of statistical hypotheses.

- Null hypothesis. The null hypothesis, denoted by Ho, is usually the hypothesis that sample observations result purely from chance.
- Alternative hypothesis. The alternative hypothesis, denoted by H1 or Ha, is the hypothesis that sample observations are influenced by some non-random cause.

@learn.machinelearning

Hypothesis testing

- **Level of significance:** Refers to the degree of significance in which we accept or reject the null-hypothesis. 100% accuracy is not possible for accepting or rejecting a hypothesis, so we therefore select a level of significance that is usually 5%.
- **Type I error:** When we reject the null hypothesis, although that hypothesis was true. Type I error is denoted by alpha. In hypothesis testing, the normal curve that shows the critical region is called the alpha region.
- **Type II errors:** When we accept the null hypothesis but it is false. Type II errors are denoted by beta. In Hypothesis testing, the normal curve that shows the acceptance region is called the beta region.
- **One-tailed test:** When the given statistical hypothesis is one value like H0: μ1 = μ2, it is called the one-tailed test.
- **Two-tailed test:** When the given statistics hypothesis assumes a less than or greater than value, it is called the two-tailed test.

@learn.machinelearning

Hypothesis testing

Steps to determine whether to reject a null hypothesis or not.

- State the hypotheses. This involves stating the null and alternative hypotheses. The hypotheses are stated in such a way that they are mutually exclusive. That is, if one is true, the other must be false.
- Formulate an analysis plan. The analysis plan describes how to use sample data to evaluate the null hypothesis. The evaluation often focuses around a single test statistic.
- Analyze sample data. Find the value of the test statistic (mean score, proportion, t statistic, z-score, etc.) described in the analysis plan.
- Interpret results. Apply the decision rule described in the analysis plan. If the value of the test statistic is unlikely, based on the null hypothesis, reject the null hypothesis.

@learn.machinelearning

Hypothesis testing

Lets take an example.

A principal at a certain school claims that the students in his school are above average intelligence. A random sample of thirty students IQ scores have a mean score of 112. Is there sufficient evidence to support the principal's claim? The mean population IQ is 100 with a standard deviation of 15.

- Step 1: State the Null hypothesis. The accepted fact is that the population mean is 100, so: H0: μ=100.
- Step 2: State the Alternate Hypothesis. The claim is that the students have above average IQ scores, so: H1: μ > 100.
- Step 3: State the alpha level. If you aren't given an alpha level, use 5% (0.05).
- Step 4: Find the rejection region area (given by your alpha level above) from the z-table. An area of .05 is equal to a z-score of 1.645.
- Step 5: Find the test statistic using this formula: $Z = \frac{\overline{x} - \mu_0}{\sigma/\sqrt{n}}$
- For this set of data: z= (112.5-100) / (15/√30)=4.56.
- Step 6: If Step 5 is greater than Step 4, reject the null hypothesis. If it's less than Step 4, you cannot reject the null hypothesis. In this case, it is greater (4.56 > 1.645), so you can reject the null. test statistic using this formula:

Probability and Statistics

Blogs to read:

1. Probability & Statistics for Data Science (Series)
2. Probability concepts explained:
3. Machine Learning — Probability & Statistics

Videos to Watch:

1. PROBABILITY & STATISTICS 1 BASICS

Free courses:

1. Khan academy Probability and Statistics
2. Intro to Descriptive Statistics
3. Intro to Inferential Statistics
4. A visual introduction to probability and statistic