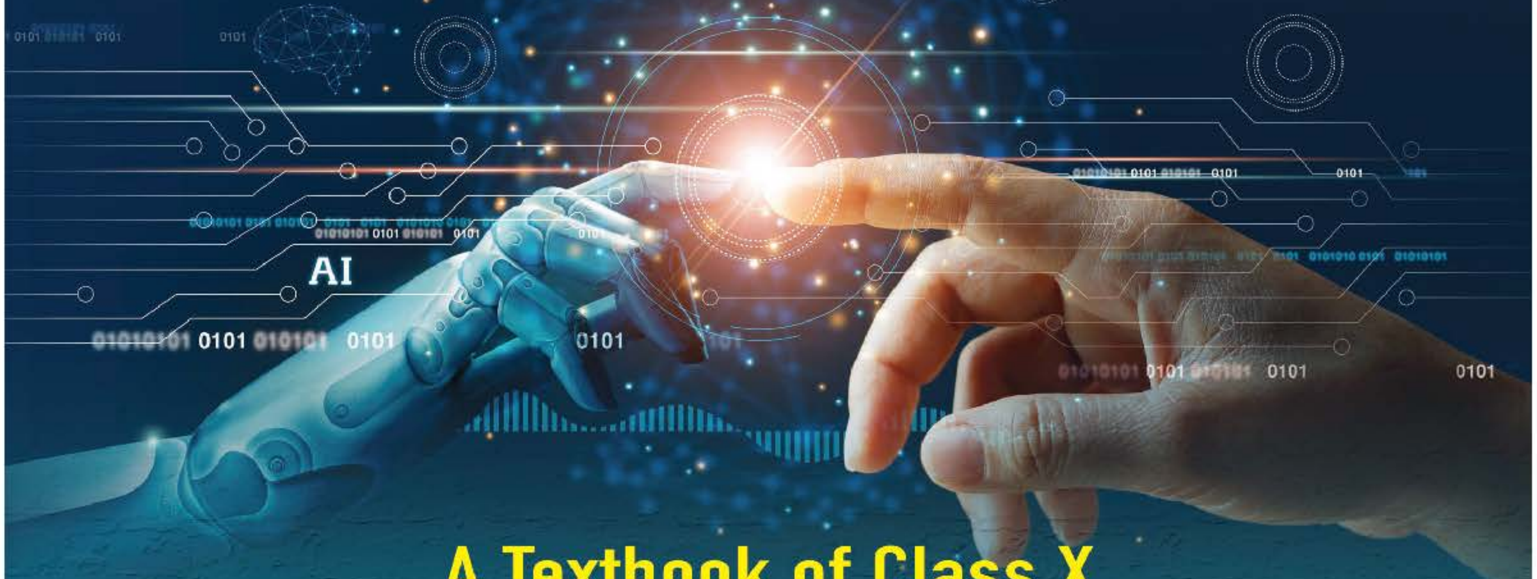
Code 417

Under NSQF

artificial intelligence



A Textbook of Class X

Part A: Employability Skills
Part B: Vocational Skills

SUMITA ARORA

DHANPAT RAI & Co.

VOCATIONAL SKILLS

PART B

©Sumita Arora

Unit 3

Advance Python

In This Unit

©Sumita Arora

SESSION 2

Python Basics in a New Flavour

- Tokens – Smallest Individual Units in a Program
- Barebones of a Python Program
- Variables and Assignments
- Data Types
- Operators

This section/content is not part of teacher-support PDF

©Sumita Arora

2.2 TOKENS – SMALLEST INDIVIDUAL UNITS IN A PROGRAM

The smallest individual unit in a program is known as a *Token* or a *lexical unit*.

Python has the following tokens :

(*i*) Keywords (*ii*) Identifiers (Names) (*iii*) Literals

(*iv*) Operators (*v*) Punctuators

Let us talk about these one by one.

Token

The smallest individual unit in a program is known as a **Token** or a Lexical Unit.

2.2.1 Keywords

Keywords are the words that convey a special meaning to the language compiler/interpreter. These are reserved for special purpose and must not be used as normal identifier names.

Keyword

A **Keyword** is a word having a special meaning reserved by the programming language.

Python programming language contains the following keywords :

False	assert	del	for	in	or	while
None	break	elif	from	is	pass	with
True	class	else	global	lambda	raise	yield
and	continue	except	if	nonlocal	return	
as	def	finally	import	not	try	

2.2.2 Identifiers (Names)

Identifiers are fundamental building blocks of a program and are used as the general terminology for the names given to different parts of the program *viz*. variables, objects, classes, functions, lists, dictionaries etc.

The following are some *valid* identifiers :

```
Myfile      DATE9_7_77      Z2T0Z9
MYFILE      _DS             _HJI3_JK
_CHK        FILE13
```

The following are some *invalid* identifiers :

`DATA-REC`	contains special character - (hyphen) (other than A to Z, a to z and _ (underscore))
`29CLCT`	starting with a digit
`break`	reserved keyword
`My.file`	contains special character dot (.)

2.2.3 Literals/Values

Literals (often referred to as constant-values) are data items that have a fixed value.

Python allows several kinds of literals :

(*i*) String literals (*ii*) Numeric literals

(*iii*) Boolean literals (*iv*) Special Literal *None*

(*v*) Literal Collections

1. String Literals

One can form string literals by enclosing text in both forms of quotes – *single quotes* or *double quotes*.

String Literal

A **String Literal** is a sequence of characters surrounded by quotes (single or double or triple quotes).

Following are some valid string literals in Python :

```
'Astha'         "Rizwan"        'Hello World'    "Amy's"
"112FBD291"     '1-x-0-w-25'    "129045"
```

©Sumita Arora

Table 2.1 *Escape Sequences in Python*

Escape sequence	*What it does [Non-graphic character]*
\\	Backslash (\)
\'	Single quote (')
\"	Double quote (")
\a	ASCII Bell (BEL)
\b	ASCII Backspace (BS)
\f	ASCII Formfeed (FF)
\n	New line character
\N{name}	Character named name in the Unicode database (Unicode only)
\r	Carriage Return (CR)
\t	Horizontal Tab (TAB)
\uxxxx	Character with 16-bit hex value xxxx (Unicode only)
\Uxxxxxxxx	Character with 32-bit hex value xxxxxxxx (Unicode only)
\v	ASCII Vertical Tab (VT)
\ooo	Character with octal value oo
\xhh	Character with hex value hh

©Sumita Arora

This section/content is not part of teacher-support PDF

This section/content is not part of teacher-support PDF

2. Numeric Literals

The numeric literals in Python can belong to any of the following *four* different numerical types :

int **(signed integers)**	often called just *integers* or *ints*, are positive or negative whole numbers with no decimal point.
long **(long integers)**	*long integers* or *longs*, are integers of unlimited size, written like integers and followed by an uppercase or lowercase L. *Integers* and *Long integers* literals can either be in ***decimal form*** (integers that do not begin with a zero) or in ***octal form*** (integers beginning with a 0 and containing only 0 to 7 digits) or in ***hexadecimal form*** (integers beginning with a 0x and containing only 0 to 9 digits and A to F or a to f letters).
float (floating point real values)	floats represent real numbers and are written with a decimal point dividing the integer and fractional parts. Floating point literals can either be in **fractional form** (*e.g.*, 17.26 or – 0.35 or 0.00017) or in **exponent form** (*e.g.*, 152E05, 1.52E07, – 0.172E–3)

3. Boolean Literals

A Boolean literal in Python is used to represent one of the two Boolean values *i.e.*, **True** (Boolean true) or **False** (Boolean false). A Boolean literal can either have value as **True** or as **False**.

4. Special Literal *None*

Python has one special literal, which is **None**. The **None** literal is used to indicate something that has not yet been created. It is equivalent to *null* of other languages.

©Sumita Arora

Arithmetic operators

+	Addition	-	Subtraction
*	Multiplication	/	Division
%	Remainder/ Modulus	**	exponent (raise to power)
//	Floor division		

Relational operators

<	Less than	>	Greater than
<=	Less than or equal to	>=	Greater than or equal to
==	Equal to	!=	Not equal to
<>	Not equal to		

Logical operators

and	Logical AND	or	Logical OR

©Sumita Arora

This section/content is not part of teacher-support PDF

Most common punctuators of Python language are :

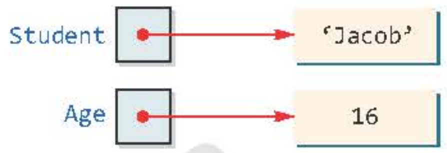
‘ “ # \ ( ) [ ] { } @ , : . ` = ;

2.4 VARIABLES AND ASSIGNMENTS

Variables represent named storage locations, whose values can be manipulated during program run.

In Python, to create a variable, just assign to its name the value of appropriate type. For example, to create a variable namely **Student** to hold student's name and variable **age** to hold student's age, you just need to write somewhat similar to what is shown below :

```
Student = 'Jacob'
Age = 16
```



Python will internally create labels referring to these values as shown here.

©Sumita Arora

This section/content is not part of teacher-support PDF

2.4.2 Multiple Assignments

Python is very versatile with assignments. Let's see how.

(*i*) ***Assigning same value to multiple variables.*** You can assign same value to multiple variables in a single statement, *e.g.*,

```
a = b = c = 10
```

It will assign value 10 to all three variables ***a***, ***b***, ***c***.

(*ii*) ***Assigning multiple values to multiple variables.*** You can even assign multiple values to multiple variables in single statement, *e.g.*,

```
x, y, z = 10, 20, 30
```

It will assign the values *order wise*, *i.e.*, first variable is given first value, second variable the second value and so on. That means, the above statement will assign value 10 to ***x***, 20 to **y** and 30 to ***z***.

If you want to swap values of *x* and *y*, you just need to write as follows :

```
x, y = y, x
```

©Sumita Arora

This section/content is not part of teacher-support PDF

Program 2.2 *Program to obtain length and breadth of a rectangle and calculate its area.*

```
# to input length and breadth of a rectangle and calculate its area
length = float( input("Enter length of the rectangle : "))
breadth = float( input("Enter breadth of the rectangle : "))

area = length * breadth

print ("Rectangle specifications ")
print ("Length = ", length, end = ' ')
print ("breadth = ", breadth)
print ("Area = ", area)
```

```
Enter length of the rectangle : 8.75
Enter breadth of the rectangle : 35.0

Rectangle specifications

Length =  8.75 Breadth =  35.0
Area =  306.25
```

©Sumita Arora

Program 2.3 *Write a program to read distance in miles and print in kilometres.*

```
miles = float(input("Enter a distance in miles:"))
kilometres = 1.609 * miles
print("The distance in kilometres is", kilometres)
```

The output produced would be :

```
Enter a distance in miles: 3.5
The distance in kilometres is 5.6315
```

2.5 DATA TYPES

Data types are means to identify the type of data and associated operations of handling it.

Python offers following built-in core data types :

(*i*) Numbers (*ii*) String (*iii*) List (*iv*) Tuple (*v*) Dictionary.

2.5.1 Numbers

The Number data types are used to store numeric values in Python. The Numbers in Python have following core data types (only applies to Python 2.x) :

Data type	***Characteristics***
(*i*) INTEGERS	***Integers are whole numbers with no fractional parts***
Plain integers	★ Normal integer representation. ★ Uses 32 bits (4 bytes) generally to store a value. ★ Represent numbers in the range − 2147483648 through 2147483647. *i.e.*, -2^{31} to $+2^{31}-1$
Long integers Suffix L makes an integer a long integer *e.g.*, 10L is a long integer but 10 is an integer.	★ Store the numbers larger than the range of plain integers. ★ More than 32 bits but no upper size limit. ★ Represent numbers less than -2^{31} or more than $+2^{31}-1$
Booleans	★ Represent the truth values ***False*** and ***True***. ★ Boolean values ***False*** and ***True*** internally behave like the values 0 and 1, respectively.
(*ii*) FLOATING-POINT NUMBERS	***Numbers having fractional part are floating-point numbers.*** ★ Floating point numbers have precision of 15 digits (double-precision). ★ Unlimited range (limited by memory availability).

©Sumita Arora

2.5.2 Strings

A string data type lets you old string data, *i.e.*, any number of valid characters into a set of quotation marks.

(Also called str type **strings)** The string type holds string of zero or more characters. These are written inside simple quotation marks (single or double both), *e.g.*, 'boy', 'Nevin', "Zig 1953", "Her's" etc.

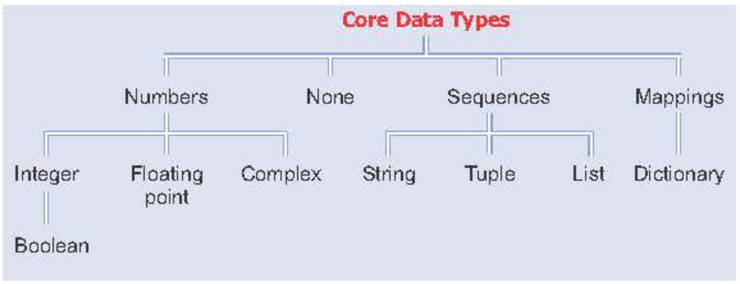
2.5.3 Lists

A List in Python represents a list of comma-separated values of any datatype between **square brackets** *e.g.*, following are some lists :

```
[1, 2, 3, 4, 5]
a = [ 'a', 'e', 'i', 'o' , 'u']
z = [ 'Neha', 102, 79.5]
```

Lists can be assigned to variables just like other datatypes and you can also change individual elements of lists.

Tuples and **Dictionaries** are also sequence type data types of Python, but covering them here is beyond the scope of the book. Following figure summarizes the core data types of Python.



2.6 GETTING USER INPUT IN PYTHON

To receive input from the user, typed through keyboard, Python provides the **input()** function, which is used as per the following syntaxes :

To get the *string* input

To get the string input, just store the result of the *input*( ) function :

```
<variable to store the input> = input([<message prompt string>])
```

e.g.,

```
str = input("Enter your name:" )
print("Hello", str)
```

```
Enter your name : Jiva
Hello Jiva
```

To get the *numeric* input

To get the numeric input, you need to use **float( )** or **int( )** function around the *input*( ) function :

```
<variable to store the input> = <float|int> (input([<message prompt string>]))
```

e.g.,

```
value = float(input("Enter packet's weight(kg):" ))
print(value, "kg")
```

```
Enter packet's weight(kg): 3.75
3.75 kg
```

The **float( )** around the **input( )** will yield floating point number (see above) and the **int( )** around the **input( )** will yield integer number from the input value (see below) :

e.g.,

```
age = int(input("Enter your age :"))
print("You are", age)
```

```
Enter your age : 15
You are 15
```

Program 2.5 *Program to obtain three numbers and print their sum.*

```
# to input 3 numbers and print their sum
num1 = int( input("Enter number 1 : "))
num2 = int( input("Enter number 2 : "))
num3 = int( input("Enter number 3 : "))
Sum = num1 + num2 + num3
print("Three numbers are : ", num1, num2, num3)
print("Sum is : ", Sum)
```

To print value stored in a named variable, give the variable name without quotes in print( )

The output produced by above program is as shown on the right.

```
Enter number 1 : 7
Enter number 2 : 3
Enter number 3 : 13
Three numbers are :  7 3 13
Sum is :  23
```

©Sumita Arora

This section/content is not part of teacher-support PDF

©Sumita Arora

Table 2.5 : *Operator Precedence*

Operator	***Description***	
( )	Parentheses (grouping)	Highest
**	Exponentiation	
+ *x*, − *x*	Positive, negative (unary + , −)	
*, /, //, %	Multiplication, division, floor division, remainder	
+, −	Addition, subtraction	
<, <=, >, >=, <>, !=, ==	Comparisons (Relational operators)	
not *x*	Boolean NOT	
and	Boolean AND	
or	Boolean OR	Lowest

This section/content is not part of teacher-support PDF

©Sumita Arora

Program 2.9 *Write a program to input a value in tonnes and convert it into quintals and kilograms.*
(1 tonne – 10 quintals 1 tonne = 1000 kg, 1 quintal = 100 kg)

```
tonnes = float( input("Enter tonnes :"))
quintals =  tonnes * 10
kgs = quintals * 100
print("Tonnes :", tonnes)
print("Quintals :", quintals)
print("Kilograms :", kgs)
```

```
Enter tonnes : 2.5
Tonnes : 2.5
Quintals : 25.0
Kilograms : 2500.0
```

Let us Revise

- *A token is the smallest individual unit in a program.*
- *Python provides following tokens : keywords, identifiers (names), Values (literals), punctuators, operators and comments.*
- *A keyword is a reserved word carrying special meaning and purpose.*
- *Identifiers are the user-defined names for different parts of the program.*
- *In Python, an identifier may contain letters (a to z, A to Z), digits (0 to 9) and a symbol underscore (_). However, an identifier must begin with a letter or underscore (_); all letters/digits in an identifier are significant.*
- *Literals are the fixed values.*
- *Python allows following literals : string literal, numeric (integer, floating-point) literals, Boolean literals, special literal None and literal collections.*
- *Operators are tokens that trigger some computation/action when applied to variables and other objects in an expression.*
- *Punctuators are symbols used to organize programing sentence structures and indicate the rhythm and emphasis of expressions, statements and program structure.*
- *A Python program can contain various components like expressions, statements, comments, functions, blocks and indentation.*
- *Comments are non-executable, additional information added in program for readability.*
- *In Python, comments begin with a # character.*
- *Comments can be single-line comment, multi-line comments and inline comments.*
- *A variable in Python is defined only when some value is assigned to it.*
- *Python supports dynamic typing i.e., a variable can hold values of different types at different times.*
- *The raw_input( ) is used to obtain input from user ; it always returns a string type of value.*
- *The input( ) function evaluates the data input and takes the result as numeric type.*
- *Output is generated through print statement.*
- *Data types are means to identify the type of data and associated operations of handling it.*
- *Python offers following built-in core data types :*
  (i) Numbers (ii) String (iii) List (iv) Tuple (v) Dictionary.
- *Operators are the symbols (or keywords sometimes) that represent specific operations.*
- *Arithmetic operators carry out arithmetic for Python. These are : unary +, unary –, +, –, *, /, //, % and **.*
- *Relational operators compare the values of their operands. These are >, <, = =, >=, <=, < > and ! = .*
- *Logical operators perform comparisons on the basis of truthness of an expression or value. These are '**or**', '**and**' and '**not**'.*

©Sumita Arora

SESSION 3

Python Conditionals and Loops

- The if Statement
- The if-else Statement
- The range( ) Function and Operators in, not in
- The for Loop
- The while Loop

This section/content is not part of teacher-support PDF

©Sumita Arora

3.2 THE if STATEMENT

The simplest form of ***if*** statement tests a condition and if the condition evaluates to *true*, it carries out some instructions and does nothing in case the condition evaluates to *false*.

The syntax (general form) of the *if* statement is as shown below :

```
if <conditional expression> :
    statement
    [statements]
```

where a statement may consist of a single statement, a compound statement, or just the ***pass*** statement (in case of empty statement).

Carefully look at the *if statement* ; it is also a *compound statement* having a *header* and a *body* containing indented statements.

For instance, consider the following code fragment :

Conditional expression

```
if ch == ' ' :
    spaces += 1
    chars += 1
```

*The **header** of if statement ; notice colon (:) at the end*

***Body of if** . Notice that all the statements in the **if-body** are indented at same level*

In an ***if statement***, if the *conditional expression* evaluates to *true*, the statements in the *body-of-if* are executed, otherwise ignored.

3.3 THE if-else STATEMENT

This form of *if* statement tests a condition and if the condition evaluates to *true*, it carries out statements indented below **if** and in case the condition evaluates to *false*, it carries out statements indented below **else**.

The syntax (general form) of the ***if-else*** statement is as shown below :

```
if <conditional expression> :
    statement
    [statements]
else :
    statement
    [statements]
```

For instance, consider the following code fragment :

```
if a >= 0 :
    print (a, "is zero or a positive number")
else :
    print (a, "is a negative number")
```

*See, the statements that are to be executed when condition is **true**, are indented below **if***

*See, the statements that are to be executed when condition is **false**, are indented below **else***

For any value more than zero (say 7) of variable *a*, the above code will print a message like :

```
7 is zero or a positive number
```

And for a value less than zero (say -5) of variable *a*, the above code will print a message like :

```
–5 is a negative number
```

©Sumita Arora

Note

There is one more if construct which is used when there is a condition in else part. It is used as :

```
if<condition> :
    <statement>
elif<condition> :
    <statements>
    ⋮
```

Solved Problem 7 uses this statement.

Program 3.1 *Write a program to input age from user and print if as per the input-age, one is eligible to vote or not.*

```
age = int(input("Enter age : "))
if age >= 18:
    print("18 and more, you are eligible to vote")
else:
    print("Under 18, You are not eligible to vote")
```

Sample run of the above program is shown below :

```
Enter age20
18 and more, you are eligible to vote
====== p1.py ======
Enter age16
Under 18, You are not eligible to vote
```

©Sumita Arora

Program 3.2 *Write a program to input two numbers and check if the two numbers are equal or not.*

```
a = int(input("Enter number 1 : "))
b = int(input("Enter number 2 : "))
if a == b:
    print("Both numbers are equal.")
else:
    print("Numbers are not equal.")
```

Sample run of above program is shown here.

```
Enter number 1 : 7
Enter number 2 : 7
Both numbers are equal.
====== p2.py =========
Enter number 1 : 7
Enter number 2 : 17
Numbers are not equal.
```

3.4 THE range( ) FUNCTION AND OPERATORS in, not in

Let us see how *range( )* function works. The common use of *range( )* is in the form given below :

```
range( <lower limit>, <upper limit>) # both limits should be integers
```

The function in the form **range**(**l**, **u**) will produce a list having values starting from $l, l+1$, $l+2 \ldots u-1$ (l and u being integers). Please note that the lower limit is included in the list but upper limit is not included in the list, *e.g.*,

```
range(0,5)
```

will produce list as [0, 1, 2, 3, 4] as these are the numbers in arithmetic progression (a.p.) that begins with lower limit 0 and goes up till *upper limit minus 1 i.e.*, 5 – 1 = 4.

```
range(5,0)
```

will return empty list [ ] as no number falls in the a.p. beginning with 5 and ending at 0 (difference $d = +1$).

```
range(12, 18)
```

will give a list as [12, 13, 14, 15, 16, 17].

©Sumita Arora

This section/content is not part of teacher-support PDF

For such lists, you can use following form of *range( )* function :

```
range( <lower limit>, <upper limit>, <step value>)  #all values
                                                    #should be integers
```

That is, function

```
range(l, u, s)              # l, u and s are integers
```

will produce a list having values as ***l, l*** **+** ***s, l*** **+2*****s*****, ... <=** ***u*** **−1**.

```
range(0, 10, 2)
```

will produce list as [0, 2, 4, 6, 8]

```
range (5, 0, -1)
```

will produce list as [5, 4, 3, 2, 1].

Note
A sequence in Python is a succession of values bound together by a single name *e.g.*, list, tuple, string. The range( ) returns a sequence of list type.

Another form of range( ) is :

```
range(<number>)
```

that creates a list from 0 (zero) to <number> − 1, *e.g.*, consider following *range( )* function :

```
range(5)
```

The above function will produce list as [0, 1, 2, 3, 4].

Operators in and not in

Let us also take about **in** operator, which is used with *range( )* in for loops.

To check whether a value is contained inside a list you can use **in** operator, *e.g.*,

```
3 in [1,2,3,4]
```

will return ***True*** as value 3 is contained in sequence [1, 2, 3, 4].

```
5 in [1,2,3,4]
```

will return ***False*** as value 5 is not contained in sequence [1, 2, 3, 4]. But

```
5 not in [1,2,3,4]
```

will return ***True*** as this fact is true that as value 5 is not contained in sequence [1, 2, 3, 4]. The operator **not in** does opposite of **in** operator.

You can see that in and not in are the operators that check for membership of a value inside a sequence. Thus, **in** and **not in** are also called **membership operators**.

©Sumita Arora

3.5 THE for LOOP

The ***for loop*** of Python is designed to process the items of any sequence, such as a list or a string, one by one.

The general form of ***for loop*** is as given below :

```
for <variable> in <sequence> :
    statements_to_repeat
```

Colon is important punctuator ; it tells that statements indented below are part of this construct.

For example, consider the following loop :

*The loop variable **a**. Variable a will be assigned each value of the list here, i.e., for the first time **a** will be **1**, then **4** and then **7**.*

```
for a in [1, 4, 7] :
    print(a)
```

*This is the **body of the for loop**. All statements in the body of the loop will be executed for each value of loop variable **a**, i.e., firstly for **a =1**; then for **a = 4** and then for **a=7***

Therefore, the output produced by above *for loop* will be :

```
1
4
7
```

©Sumita Arora

Program 3.6 *Write a program to print all items from a list containing some colour names in it.*

```
colist = ["Blue", "Green", "Yellow", "Orange", "Red", "Brown"]
for colname in colist:
    print(colname)
```

Sample run of above program is shown below :

```
Purple
Grey
Blue
Red
Black
```

©Sumita Arora

Program 3.8 Program to print table of a number, say 5.

```
num = 5
for a in range(1, 11) :
    print(num, 'x', a, '=', num * a)
```

The above code will print the output as shown below :

```
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50
```

This section/content is not part of teacher-support PDF

3.6 THE while LOOP

A ***while loop*** is a conditional loop that will repeat the instructions within itself as long as a conditional remains *true* (Boolean *True* or truth value $true_{tval}$). The general form of Python while loop is :

```
while <logicalexpression> :
    loop-body
```

where the loop-body may contain a *single statement* or *multiple statements* or *an empty statement* (*i.e.*, *pass* statement). The loop iterates while the *logicalexpression* evaluates to *true*. When the expression becomes *false*, the program control passes to the line after the loop-body.

To understand the working of ***while loop***, consider the following code :

```
a = 5
while a > 0 :
    print("hello", a)
    a = a - 3
print("Loop Over!!")
```

: means loop-body's statements are indented below.

*This condition is tested, if it is **true**, the loop-body is executed. After the loop-body's execution, the condition is tested again, loop-body is executed as long as condition is **true**.*

*The loop ends when the condition evaluates to **false***

All the statements indented below the **while**, are part of while loop's body.

The above code will print :

```
hello 5
hello 2
Loop Over!!
```

*These two lines are the result of **while loop's** body execution (which executed twice).*

*This line is because of print statement after the **while loop.***

©Sumita Arora

Program 3.11 *Write a Python Program to print squares of first 5 natural numbers.*

```
Num = 1
while Num <= 5 :
    print(Num * Num)
    Num = Num ± 1
```

The output produced by the above code is :

```
1
4
9
16
25
```

Program 3.12 *Write a program to input a number (num) and print the sum of numbers from 1 to num.*

```
sum = 0
num = int(input("Please enter a number: "))
for i in range(1, (num + 1) ):
    sum = sum + i
print("Sum of numbers 1 to", num,"is: ", sum)
```

Sample run of the above program is shown below :

```
Please enter a number: 8
Sum of numbers 1 to 8 is: 36
```

©Sumita Arora

This section/content is not part of teacher-support PDF

Let us Revise

- *Python provides selection or conditional statement in the form of* ***if*** *and if-else.*
- *The if..else statement tests an expression and depending upon its truth value one of the two sets-of-action is executed.*
- *The range() function generates a sequence of list type.*
- *The statements that allow a set of instructions to be performed repeatedly are iteration statements.*
- *Python provides two looping constructs –* ***for*** *and* ***while****. The* ***for*** *is a counting loop and* ***while*** *is a conditional loop.*

This section/content is not part of teacher-support PDF

©Sumita Arora

UNIT 4

Data Sciences

IN THIS UNIT

©Sumita Arora

SESSION 1

Introducing Data Sciences

- What is Data Science ?
- Applications of Data Science

©Sumita Arora

This section/content is not part of teacher-support PDF

1.2 WHAT IS DATA SCIENCE ?

Data Science

Data Science is a field that uses scientific (mathematical and statistical) methods, processes, algorithms and systems to extract knowledge and insights from huge volumes of structural and unstructured data to apply in AI applications.

1.3 APPLICATIONS OF DATA SCIENCE

1. Fraud and Risk Detection

- **For Fraud Detection,** the Data science provides many techniques for processing the huge datasets. For example, for a finance or banking company, if its dataset, containing the transactions taken place over a time, is plotted and analysed, the plots can easily pinpoint :
  - which transactions involve larger sums of money ?
  - which transactions involved larger sum of money for cash-out or money transfers ?

  Looking at these it becomes easier to detect the fraud, if any, in the transactions.
- **For Risk Detection** also, the data science provides many techniques to analyse the past expenditure trends and behaviour of customer, customer profiling. All this is done to figure out if the customer is loan-worthy or would return the loan or default the loan? For example, using the past transactions dataset of loan applicants (the customers), a *"Customer Profiling" subsystem*, using the data science tools analyses the customers' banking-related interactions (*e.g., monthly cash flows, loans, cards*) and their legal information (*e.g., demographics, employment, marital, financial* and *household information*) and then takes a decision if a customer is truly loan-worthy and would return the loan.

©Sumita Arora

This section/content is not part of teacher-support PDF

2. Healthcare

Medical and Healthcare is a diverse field involving diagnosis, drugs and research. Data science tools are useful in all fields of healthcare for getting useful insights and making crucial decisions.

- **Diagnosis with Medical Image Analysis.** Medical images can be transformed in the form of data and that huge data may be analysed to pinpoint flaws by getting to unusual points or features seeking attention. This helps in diagnosing known and unknown diseases and ailments. This has been useful in diagnosing various diseases like detecting tumours, organ delineation, and much more.
- **Genetics and Genomics Research.** A genome is an organism's complete set of genetic instructions, the entire sequence of DNA. Genomics is an area within genetics that concerns the sequencing and analysis of an organism's genome.

  Modern data science tools reduce the processing time for genome sequencing significantly and help analyse the reaction of genes to various medications.

This section/content is not part of teacher-support PDF

©Sumita Arora

3. Internet Search

Internet means crores and billions of connected computers and servers. Imagine how much data is stored and processed over the Internet? Searching on such a huge network means searching for a term or group of words in **Peta Bytes** (1000 × 1000 Giga Bytes) or **Exabytes** (1000 Peta bytes) or **Zetta Bytes** (1000 Exabytes) of data available on the Internet. Searching in such huge volumes of data requires data science tools which can process such huge volumes of data.

4. Targeted Advertising

For target advertising, the data science works with data based on browsing habits, past purchases, or any other recent activities and picks trends and most prominent feature and based on that target advertising takes place. So, two persons simultaneously browsing a shopping site on two different devices will see different advertisements depending upon their past searches and other variables.

©Sumita Arora

This section/content is not part of teacher-support PDF

5. Recommender Systems (RSs)

A Recommender System (RS) refers to a system that is capable of predicting the future preference of a set of items for a user, and recommending the top items. A recommender system is used by many ***online retail hubs*** like *Amazon*, ***entertainment and digital content sites*** such as *Netflix*, *Hotstar*, *Discovery*, ***Travel and hotel booking sites*** like *MakeMyTrip*, *Expedia* and so forth. The recommendations about the products, digital content, movies or hotels etc. not only help you find relevant products from billions of products available with them, but also adds a lot to the user experience. As the number of available options is too huge these days, people had a hard time selecting the items they actually want to see. This is where the recommender system comes in and serves its core purpose – identifying items useful for the user. The recommendations are made based on previous search results for a user.

The Recommender Systems are useful for the users in various ways, such as :

- Assistance in decision-making
- Assistance in comparison
- Assistance in discovery
- Assistance in exploration

©Sumita Arora

This section/content is not part of teacher-support PDF

6. Airline Route Planning

Thus, data science is helping airline companies in crucial decisions like :

- Popular demand of fliers
- Planning air routes according to demand
- Decisions with flight delays
- Inflight food supply choices
- Deciding about fares and discounts
- Loyal Customers' benefits

©Sumita Arora

This section/content is not part of teacher-support PDF

Let Us Revise

- *Data Science is a field that uses scientific methods, processes, algorithms and systems to extract knowledge and insights from many structural and unstructured data to apply in AI applications.*
- *The field of data science combines concepts and methods from various fields of statistics, mathematics, data analysis, machine learning, computer science and so forth.*
- *Data Science is a broader term that makes use of Data Analytics, which analyses and gathers insights from past data.*
- *Data science has found applications in all data heavy fields like Finance and Banking for fraud and risk detection, healthcare, Internet search, Recommender systems, air route planning, targeting advertising, weather prediction for agriculture sector and so on.*
- *Targeted advertising is a type of online advertising that is oriented toward audiences who share certain characteristics, depending on the product or person being promoted.*
- *A Recommender System (RS) refers to a system that is capable of predicting the future preference of a set of items for a user, and recommend the top items.*

©Sumita Arora

This section/content is not part of teacher-support PDF

SESSION 2

- Data Sciences with AI Project Cycle
- Data Collection

Revisiting AI Project Cycle (Data Sciences)

©Sumita Arora

This section/content is not part of teacher-support PDF

2.3 DATA COLLECTION

In the previous section, you saw that for our data science project, we collected the sales data required for the model training and testing, from the sale-invoices of the showroom. That is, we could get recorded real data. But depending upon the nature of your project, you might not get the recorded data. To collect data for such projects, you can use different ways of data collection as you learnt in class 9, *i.e.*, using one of the following, as per the needs of your project :

Table 2.1 : *Offline and Online Data Collection Sources*

Offline Data Collection	*Online Data Collection*
Sensors	Open-sourced Government Portals, such as : mospi.nic.in/data, surveyofindia.gov.in/
Surveys	Online survey sites
Interviews	Reliable Websites *e.g.*, Kaggle.com, Qunadl.com, grouplens.org, data.world
Focus Groups	Online Forums
Observations	Online Polls
Records and Documents	Open-sourced statistical websites like : data.gov.in, rbi.org.in/Scripts/Statistics.aspx, datasetsearch.research.google.com, trends.google.com

©Sumita Arora

While getting and collecting data from online and offline sources, you should keep the following things in mind :

- Always obtain permission from the rightful owner of data to use the data and cite the source of data.
- Always take data from the real, designated authentic sites. Do not take data from any random sites.
- Make sure your data source is reliable and you are legally allowed to obtain and use the data for what you want to do with it.
- For taking data from an open website, make sure that the data being downloaded is available in public domain and is not violating any copyright or other legal issues. Always ensure the authenticity of the website from where you are taking the data, even if the data available there is free.
- Always ensure not to use anyone's private data or violate any individual or firm's right of privacy.

2.3.1 Data Formats/Types Used in Data Science

In data science projects, data coming from various types of sources, having different types and formats is processed. Data used in data science projects can be in different formats. Most common data formats used in Data Science are :

- **CSV (Comma-separated values).** A CSV file, or a comma-separated values file, is a way of storing information in a tabular format in a plain text file. CSV is essentially a text file format and can be opened and used through any text editor.
- **Spreadsheets.** This is a tabular representation of data arranged in rows and columns and can be accessed and processed through various spreadsheet software like Microsoft Excel, OpenOffice Calc etc.

- **XLSX**. A file is a Microsoft Excel **Open XML Format Spreadsheet** file. It's a ZIP-compressed, XML-based spreadsheet file created by Microsoft Excel version 2007 and later.

  XLSX files organize data in cells that are stored in worksheets, which are in turn stored in workbooks.
- **JSON (JavaScript Object Notation).** JSON is a standard text-based format for representing structured data based on JavaScript object notation syntax. It is a lightweight data-interchange format used to exchange data between computers. JSON is language independent.
- **SQL (Structured Query Language).** The SQL data is mainly data stored in DBMS (database management systems) that use SQL as a query language to access and process its data.
- **XML (eXtensible Markup Language).** XML is a markup language which encodes documents by defining a set of rules in both machine-readable and human-readable format. XML is powerful for both web-scraping and general practice in parsing a structured document.

There are many other data types and formats used in data sciences but discussing about them here is beyond the scope of the book, hence, we are not covering them here. With this we have come to the end of our session.

©Sumita Arora

This section/content is not part of teacher-support PDF

Let Us Revise

- *Like any other AI project, a data science project also undergoes the phases of AI project cycle where each phase is data centric.*
- *After collecting data for a data science project, data is first cleaned, standardised and the missing data is handled.*
- *Then the data is analysed and visualised to know about the trends and key characteristic of data.*
- *Then for data modelling, the treated dataset is divided into training dataset and testing dataset and the AI based model is trained using the training dataset using one of the training techniques - supervised / unsupervised or reinforced learning.*
- *For model evaluation, the testing dataset is used and the predicted values are compared with the actual results to determine the efficacy and efficiency of the model developed and trained.*
- *For collecting data for data science projects, there are many online and offline sources.*
- *There are many data types and formats used in data science projects such as CSV, XLSX, spreadsheets, SQL, XML, JSoN etc.*

©Sumita Arora

SESSION 3

Python for Data Sciences

- Numpy
- Pandas
- Matplotlib
- Basic Statistics

©Sumita Arora

3.2 NUMPY

NumPy, which stands for **Numerical Python**, is a Python library that provides functionality (in terms of functions and routines) to do the following :

- Create a multidimensional array object (called **ndarray** or **NumPy array**).
- Providing tools for working with ndarrays, *i.e.*, performing mathematical and logical operations.

Note

NumPy is the core library for scientific computing in Python that can work with high performance ndarrays and thus, it is widely used in data science projects too.

Let us now briefly talk about the core component upon which NumPy functions work, the NumPy arrays.

3.2.1 NumPy Arrays

NumPy Array

A NumPy array is simply a grid that contains values of the same/homogeneous type.

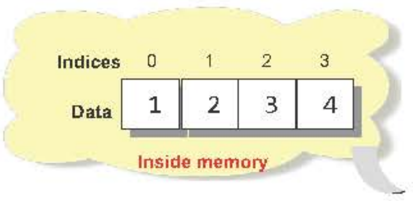
An array in general refers to a named group of homogeneous (of same type) elements. For instance, if you store the similar details of all sections together, *e.g.*, if you store *number of students* in each section of class X in a school, in a common name ***Students***, containing 5 entries as [34, 37, 36, 41, 40] then ***Students*** is actually an array that represents number of students in each section of class X. Like lists, you can access individual elements by giving *index* with array name *e.g.*, **Students[1]** will give details about 2nd section, **Students[3]** will give you details about 4th section and so on. Like lists, arrays also have first index as 0 (zero).

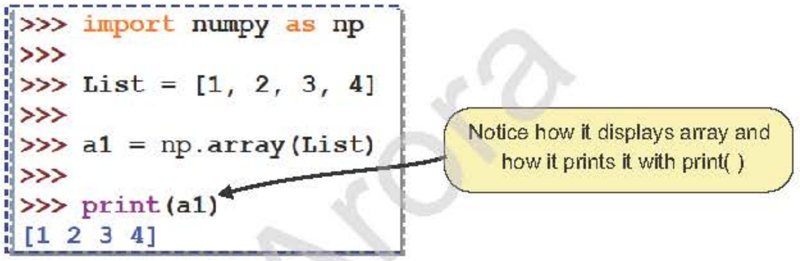
Consider the following code :

```
import numpy as np
List = [1, 2, 3, 4]
a1 = np.array(List)
print(a1)
```

Now, you can use numpy functions either as numy.functionName or np.functionName, e.g., numpy.array( ) or np.array( )

It will create a NumPy array from the given list





```
>>> import numpy as np
>>>
>>> List = [1, 2, 3, 4]
>>>
>>> a1 = np.array(List)
>>>
>>> print(a1)
[1 2 3 4]
```

Notice how it displays array and how it prints it with print( )

Individual elements of the above array can be accessed just like you access a list's, *i.e.*,

Note
The [ ] after the object is known as the indexing operator.

`<array-name>[<index>]`

That is, ***a1[0]*** will give you ***1***, ***a1[1]*** will give you ***2***, ***...a1[3]*** will give you ***4***.

Although NumPy Arrays look similar to **Python Lists** (you learnt about them in class 9), yet they are different from one another.

Following section talks about the same.

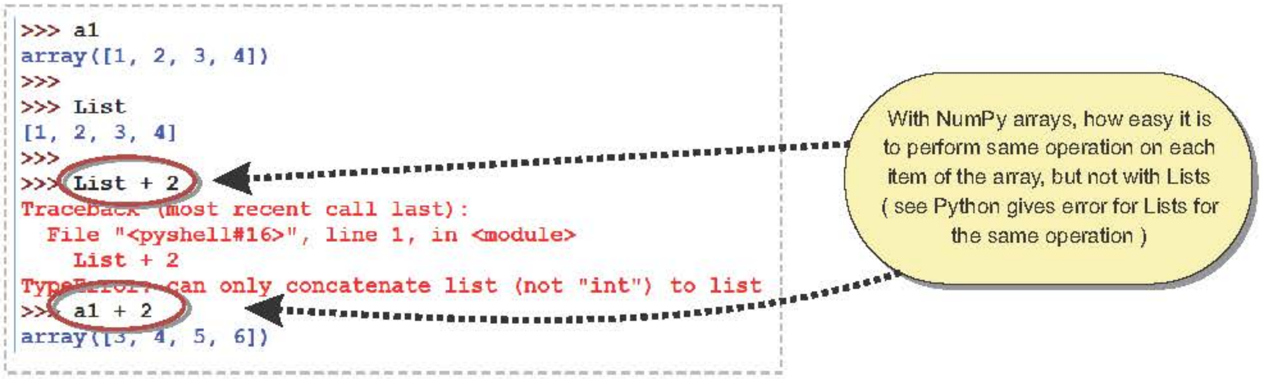
3.2.2 NumPy Arrays vs. Python Lists

Differences of NumPy Arrays with Python Lists

The key differences between a NumPy array (also called **ndarray**) and a list are :

- Unlike Python lists, once a NumPy array is created, you cannot change its size. You will have to create a new array or overwrite the existing one. That is, internally a new ndarray will be created rather than modifying existing ndarray.
- Every NumPy array can contain elements of homogenous types *i.e.*, all its elements have one and only one (same) *dtype* (data type), unlike Python lists.
- An equivalent NumPy array occupies much less space than a *Python list* or *list of lists*.
- NumPy arrays support **vectorized operations**, *i.e.*, if you apply a function, it is performed on every item (*element by element*) in the array unlike lists.

©Sunita Arora



```
>>> a1
array([1, 2, 3, 4])
>>>
>>> List
[1, 2, 3, 4]
>>>
>>> List + 2
Traceback (most recent call last):
  File "<pyshell#16>", line 1, in <module>
    List + 2
TypeError: can only concatenate list (not "int") to list
>>> a1 + 2
array([3, 4, 5, 6])
```

With NumPy arrays, how easy it is to perform same operation on each item of the array, but not with Lists (see Python gives error for Lists for the same operation)

This section/content is not part of teacher-support PDF

©Sumita Arora

PART B

3.3 PANDAS

Pandas or **Python Pandas** is Python's library for data analysis. **Pandas** has derived its name from "**panel data system**", which is an ecometrics term for multi-dimensional, structured data sets. Today, Pandas has become a popular choice for data analysis and Data Sciences.

The Pandas library has ***two primary data structures***, **Series** (1-dimensional) and DataFrame (2-dimensional), that can handle the vast majority of typical use cases in finance, statistics, social science, and many areas of engineering. Pandas is built on top of NumPy and is intended to integrate well within a scientific computing environment with many other 3rd party libraries.

Here are just a few of the things that pandas does well :

- Easy handling of missing data (represented as NaN, *i.e.*, Not a Number) in floating point as well as non-floating point data.
- Size mutability : columns can be inserted and deleted from DataFrame and higher dimensional objects.
- Automatic and explicit data alignment : objects can be explicitly aligned to a set of labels, or the user can simply ignore the labels and let *Series*, *DataFrame*, etc. automatically align the data for you in computations.
- Intelligent label-based slicing, fancy indexing, and subsetting of large data sets.
- Intuitive merging and joining data sets.
- Flexible reshaping and pivoting of data sets.
- Robust IO tools for loading data from flat files (*e.g.*, CSV), Excel files, databases, and saving/loading data from the ultrafast HDF5 format".

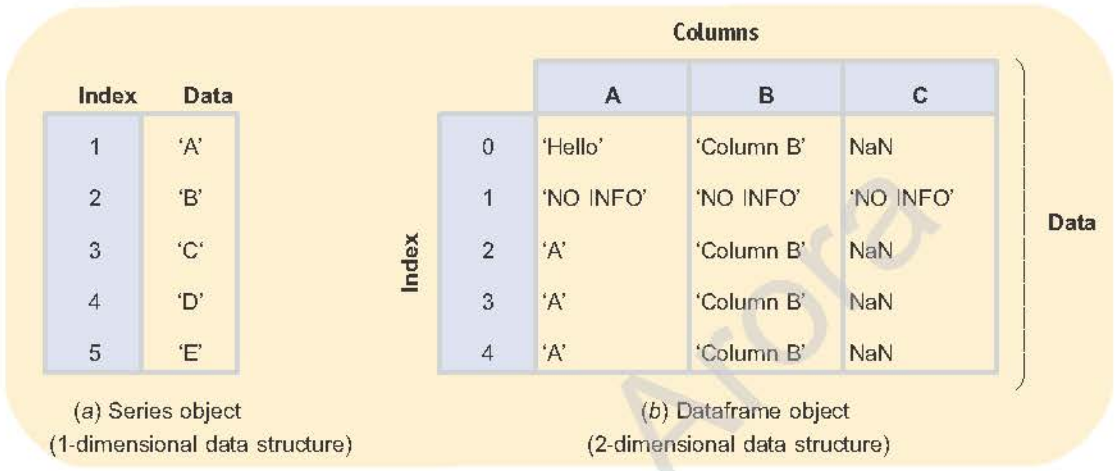
The above text has been taken from Pandas' official documentation.

©Sumita Arora

This section/content is not part of teacher-support PDF

Two Basic Pandas' Data Structures Series and DataFrame

Series is 1-dimensional data structure of Python Pandas and **Dataframe** is a 2-dimensional data structure of Python Pandas. *Pandas* also supports another data structure called **Panel**, but that is beyond the scope of this book. Thus, our discussion will remain limited to Series and DataFrame data structures only. Following Fig. 3.1 shows a Series and a DataFrame object of Python Pandas.



Index	Data
1	'A'
2	'B'
3	'C'
4	'D'
5	'E'

	A	B	C
0	'Hello'	'Column B'	NaN
1	'NO INFO'	'NO INFO'	'NO INFO'
2	'A'	'Column B'	NaN
3	'A'	'Column B'	NaN
4	'A'	'Column B'	NaN

Columns
Index
Data
(a) Series object
(1-dimensional data structure)
(b) Dataframe object
(2-dimensional data structure)

Figure 3.1 Pandas' two basic data structures

P A R T B

©Sumita Arora

3.4 MATPLOTLIB

Matplotlib is a Python library used for Data Visualization.

You can create bar-plots, scatter-plots, histograms and a lot more visualisations with Matplotlib. Matplotlib is a popular choice for data visualisations via Python for the following reasons :

- Create publication quality plots.
- Make interactive figures that can zoom, pan, update.
- Customize visual style and layout.
- Export to many file formats.
- Embed in *JupyterLab* and *Graphical User Interfaces*.

1. Mean

The mean is the average of the numbers. To calculate *mean* :

Just add up all the numbers then divide by count of numbers (how many total numbers there are), *i.e.,* ***sum/count.***

For example, to calculate the mean of numbers **7, 13, 22** :

(*i*) add the numbers : $7+13+22=42$, *i.e.*, sum = 42.

(*ii*) count of numbers is 3 ; divide sum by count : $\frac{42}{3}=14$.

The **mean** refers to the average of the numbers

Thus, the mean of above given three numbers is *14*.

2. Median

The median is the ***middle*** *number in a set of data that is ordered from least to greatest.* To calculate *median* :

Arrange the values of the dataset in increasing order and pick the middle most value. If there is an even number of middle data, you take the average of the middle two numbers to find the median.

For example, to calculate the median of numbers **7, 13, 22** :

(*i*) arrange the numbers in increasing order.

(*ii*) consider the middle number from the arranged set of numbers, *e.g.*, for the given set 7, 13, 22, the middle number is 13, which is the median.

©Sumita Arora

3. Mode

The ***mode*** *is the number that occurs most often.* To calculate *mode* :

Arrange the dataset in increasing order and find the most repeating number.

For example, to calculate the median of numbers **7, 13, 22, 13** :

(*i*) arrange the numbers in increasing order : 7, 13, 13, 22.

(*ii*) find the number which repeats the most, which is 13 in this case – it is the mode for the given set.

Let us now understand what these statistical techniques of *range* and *standard deviation* are.

1. Range

The **Range** is the difference between the lowest and highest values of a given set of values.

To calculate the range :

Just take the difference between the highest and lowest values of the data set.

For example, to calculate range for a given set of values 7, 13, 15, 22 :

(*i*) find the lowest value, which is 7 in this case and the highest value, which is 22 in this case.

(*ii*) calculate range as highest – lowest, *i.e.*, $22 - 7 = 15$ is the range.

2. Variance and Standard Deviation

In order to understand standard deviation, you must know what variance is. So let us first talk about **Variance**.

Variance

The numbers 7, 13 and 22 have mean as 14 as you saw earlier.

Now each of these numbers has a difference or distance from *mean* as :

Number 7's difference from mean : $14 - 7 = 7$

Number 13's difference from mean : $14 - 13 = 1$

Number 22's difference from mean : $14 - 22 = -8$

©Sumita Arora

Standard Deviation

The **Standard Deviation** (shown with Greek letter sigma σ) is a measure of how spread out numbers are. It is calculated as the ***square root of the Variance*** (*i.e.*, $\sqrt{\text{Variance}}$).

In the previous section, you calculated the variance for numbers 7, 13 and 22 as ***38***.

∴ Standard deviation for the numbers 7, 13 and 22 will be : $= \sqrt{38} = 6.164 \approx 6$

This section/content is not part of teacher-support PDF

3.5.3 Basic Statistics with Python

This section will briefly discuss how you can use various statistics functions to calculate different statistical values. We shall use NumPy library's statistical functions to do this.

You can use these functions as (after importing numpy library) :

```
numpy.<function>(<ndarray>)
```

OR

```
np.<function>(<ndarray>)      #if you have imported numpy as np
```

Following examples will make it clear. All the following examples will use the ndarray **ary1** as given below :

import numpy library first of all

```
>>> import numpy as np
>>> ary1 = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9])
>>> ary1
array( [1, 2, 3, 4, 5, 6, 7, 8, 9] )
```

Function	***Purpose***	***Example***
mean( )	Calculates mean from all the values of ndarray (*whole ndarray*)	`>>> np.mean(ary1)` `5.0` ← *Result*
median( )	Calculates median from all the values of ndarray (*whole ndarray*)	`>>> np.median(ary1)` `5.0` ← *Result*
var( )	Calculates variance from all the values of ndarray (*whole ndarray*)	`>>> np.var(ary1)` `6.666666666666667` ← *Result*
std( )	Calculates standard deviation from all the values of ndarray (*whole ndarray*)	`>>> np.std(ary1)` `2.581988897471611` ← *Result*

©Sumita Arora

Let Us Revise

- *NumPy is the core library for scientific computing in Python that can work with high performance ndarrays and thus, it is widely used in data science projects too.*
- *The main data structure in NumPy is the **ndarray**, which is a shorthand name for N-dimensional array, container of items of the same type.*
- *NumPy arrays store homogeneous data unlike list that store heterogeneous data and are faster than lists and take less memory than lists.*
- *Pandas is a software library written for the Python programming language for data manipulation and analysis, and is popular choice for data sciences.*
- *Matplotlib is a Python library used for Data Visualization. Data Visualization is an essential component of data sciences.*
- *Data visualization is the graphical representation of information and data. By using visual elements like charts, graphs, and maps, data visualization tools provide an accessible way to see and understand trends, outliers, and patterns in data.*
- *You can create bar-plots, scatter-plots, histograms and a lot more visualisations with Matplotlib.*

©Sumita Arora

This section/content is not part of teacher-support PDF

SESSION 4

K-Nearest Neighbour Model

- Personality Prediction
- Understanding K-Nearest Neighbour (KNN) Model

©Sumita Arora

This section/content is not part of teacher-support PDF

This section/content is not part of teacher-support PDF

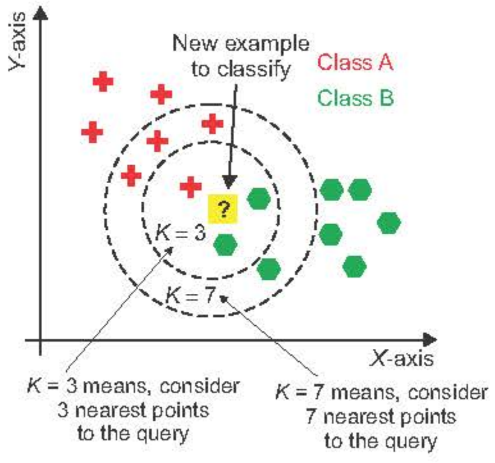
©Sumita Arora

The saying used for people *"A person is known by the company he keeps"* applies to data points too and this ideology is used by the KNN algorithm. KNN is a supervised learning algorithm and it mainly works as follows:

(*i*) Take the unlabelled point (also called *query*) to be classified into a category.

(*ii*) Find K most similar labelled points (closest neighbours) nearest to the unlabelled point among available sample points in a cell contain specified number of points (some number or value K determines how many points should be taken in cell).

(*iii*) Notice which class to the already labelled points belong to, in the marked cells ? (consider adjacent figure).


Y-axis
New example to classify
Class A
Class B
?
K = 3
K = 7
X-axis
K = 3 means, consider 3 nearest points to the query
K = 7 means, consider 7 nearest points to the query

(*iv*) Assign the most frequent class among those neighbours to our unlabelled point (query point).

(*a*) For example, in the above figure, yellow is the query point, and we want to know which class it belongs to [***Class A*** : Red (✚) or **Class B** : Green (⬢)].

With value $K = 3$, the 3 nearest neighbours of the yellow point would be taken in a cell and the inner circle makes this cell (*see* the above figure) containing 3 labelled points in it.

(*b*) With value $K = 7$, the 7 nearest neighbours of the yellow point would be taken in a cell and the outer circle makes this cell (*see* the above figure) containing 7 labelled points in it (3 in inner circle + 4 outside inner circle).

(*c*) Determine, to which class most of labelled points belong to – in the above shown case, with $K = 7$, **4 labelled points belong to Class A** and only **3 labelled points belong to Class B** and as per majority ***Class A wins here***.

(*d*) Assign the majority class to the query point, thus, the class assigned to our *unlabelled query point* is **Class A**.

©Sumita Arora

Thus, KNN modelling works around four parameters:

(*i*) ***Features.*** The variables based on which similarity between two points is calculated.

(*ii*) ***Distance function.*** Distance metric to be used for computing similarity between points.

(*iii*) ***Neighbourhood (K).*** Number of neighbours to search for.

(*iv*) ***Scoring function.*** The function which finds the majority score and class for the query point.

KNN

KNN (K-Nearest Neighbour) algorithm is a Supervised Learning algorithm that classifies a new data point into the target class, counting on the features of its neighbouring data points.

4.3.2 Applications of KNN Models

K Nearest Neighbour algorithm falls under the Supervised Learning category and is used for **classification** (most commonly) and **regression** based problems.

KNN models are used in many different fields and areas such as :

- **For Recommender Systems** based on similar tastes or requirements. Many companies make a personalized recommendation for its consumers, such as Netflix, Amazon, YouTube, and many more.
- **For Looking for Similar Documents.** If documents are close to each other, that means the documents contain identical topics.
- **For Text Categorisation.** If two texts contain similar types of terms they belong top same genre *e.g.,* two text containing more mathematical or scientific terms are scientific documents.
- **For Detecting Frauds** where one such example is Credit Card fraud detection where the spending behaviour of fraudster is way different from the real card owner and this is detected through KNN.
- **In Health and Medicine** for predicting whether a patient, hospitalized due to a heart attack, will have a second heart attack. The prediction is to be based on demographic, diet and clinical measurements for that patient. Similarly, to identify the risk factors for prostate cancer, based on clinical and demographic variables and many other health applications.
- **For Estimating Soil Water Parameters** by analysing characteristics of soil and water and looking for similarities.

Some other applications of KNN are : Forecasting stock market; Currency exchange rate; Bank bankruptcies; Understanding and managing financial risk; Trading futures; Credit rating; Loan management; Bank customer profiling; Money laundering analyses and many others.

©Sumita Arora

This section/content is not part of teacher-support PDF

LET US REVISE

- *KNN (K Nearest Neighbour) algorithm classifies a new data point into the target class, counting on the features of its neighbouring data points.*
- *K Nearest Neighbour algorithm falls under the Supervised Learning category and is used for classification (most commonly) and regression based problems.*
- *K refers to the number of neighbours to be considered for a query point.*
- *K is usually taken as an odd number to always have a tiebreaker.*
- *Low value of K results in less table results.*
- *Higher value of K results in stable and more accurate results.*
- *Very high value of K results in errors and underfitting.*
- *KNN algorithm uses methods and techniques to determine an optimal value for K.*

©Sumita Arora

This section/content is not part of teacher-support PDF

UNIT 5

Computer Vision

IN THIS UNIT

©Sumita Arora

SESSION 1

Introducing Computer Vision and Its Applications

- What is Computer Vision ?
- Applications of Computer Vision

©Sumita Arora

This section/content is not part of teacher-support PDF

Computer Vision (CV)

Computer Vision (CV) is a subset of artificial intelligence that makes computers, machines and devices visually enabled by giving them capability to analyse and understand the image captured by camera.

1.3 APPLICATIONS OF COMPUTER VISION

1. Facial Recognition

A facial recognition system is a computer vision based technology that identifies or verifies or matches a digital image (live or stored) of a human face against a database of stored face images.

For example, while entering into a house, a facial recognition system's camera can capture your face and then decide if you are an authorised entrant in the house or not. If not, it may raise alarm or inform authorities as programmed.

©Sumita Arora

2. Face Filters

Modern popular apps like Instagram and Snapchat provide options to apply face filters. A face filter is a feature that allows you to edit your photo with one click, by simply applying preset (predesigned) edits or changes to the facial image loaded on it.

3. Image Search on Search Engines

For example, you have seen a beautiful sofa on the Internet and now you want to find where it can be available in the furniture stores of your city. For this, you can paste the image of sofa in the search engine's search box and it will fetch the matching results. From the results you can find the furniture store of your area selling this sofa.

Google reverse image search, officially called **Google Search by Image**, is a service provided by Google that allows a user to search for images using an image as the starting point, rather than using a typed or spoken keyword (search query).

4. Computer Vision in Retail

Self-checkout or cashier-less stores use computer vision and deep learning technologies to automatically detect the prices and calculate the bill of products a shopper picks.

5. Inventory Management

Other than self-checkout, shoppers also expect precise information about the availability of certain products as they browse an online store. Again, because of computer vision technology, when a shopper buys an item and makes payment, the CV system deducts the quantity of the item from the stock. Thus, the inventory keeps getting updated in real time. The retailer can then determine in real time the status of inventory and stock.

By automating inventory cycle counts with computer vision, retail businesses can update their inventory system in real-time to develop an omnichannel retail experience.

©Sumita Arora

6. Self-Driving Cars

The self-driving cars are often talked about these days. In Self-Driving Cars, Computer Vision is one of the most important technology used along with a good quality fast camera and automatic route navigation in a self-driving car.

7. Google Lens/Translate App

Google Lens is an AI-powered technology that uses your smartphone camera and deep machine learning to not only detect the object in front of the camera lens but understand this and offer options such as scanning, translation, shopping and more.

Google Lens also facilitates **Google translate**, by pointing to a text of any language and translating it to the language of your choice using Computer Vision and other AI technologies and Optical Character Resolution (OCR).

8. Medical Imaging

As you know that Computer Vision is a technology based on image processing and synthesis. It usually involves machine learning and allows AI to simulate human vision. Thus, it is now widely used in medical imaging in the form of many applications.

9. Livestock Farming

Computer Vision systems can monitor animals such as cattle, sheep, pigs, or others with cameras. Neural networks are used to analyse video feeds in real-time. This helps the farmers to take timely decisions about the livestock's health, state and need etc.

©Sumita Arora

This section/content is not part of teacher-support PDF

Let Us Revise

- *Computer Vision (CV) is a subset of artificial intelligence that makes computers, machines and devices visually enabled by giving them capability to analyse and understand the image captured by camera.*
- *A facial recognition system is a computer vision based technology that identifies or verifies or matches a digital image (live or stored) of a human face against a database of stored face images.*
- *A face filter is a computer generated effect that applies preset (predesigned) edits or changes to a loaded facial image.*
- *Reverse Image Search refers to a search where the search query is formed using a pasted image and the search engine looks for matching or similar images from around the Web.*
- *Google reverse image search facility is officially known as Google Search by Image.*
- *Computer vision technology has found applications in many areas such as Facial recognition, face filters, reverse image search, CV in retail, inventory management, self-driving cars, lens/translate apps, medical imaging, livestock farming, crop monitoring and many others.*

This section/content is not part of teacher-support PDF

©Sumita Arora

SESSION 2

Computer Vision Concepts and OpenCV

- Concepts of Computer Vision
- Python OpenCV

This section/content is not part of teacher-support PDF

©Sumita Arora

2.2.3 Computer Vision Tasks

Computer vision tasks are the ways to identifying, labelling, determining their position and location along with categorising them in various types or groups. The computer vision tasks are very important as these help computer vision do what it does. Just as our eyes can figure out objects, their types, location and categories from an image, CV tasks do just the same.

The computer vision tasks vary when an image has just one object in it and when an image has multiple objects in it. Various computer tasks[1] that take place are shown in Fig 2.5.

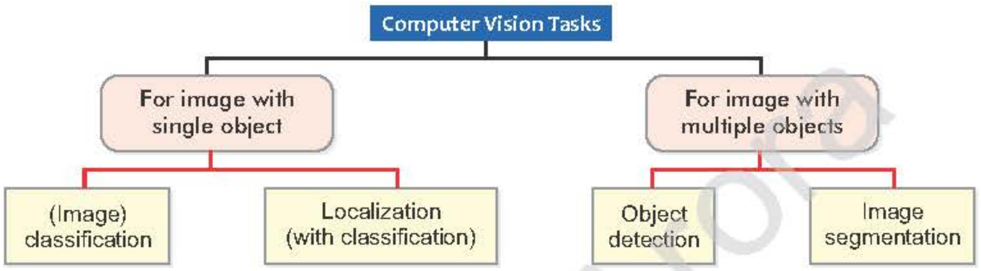

Computer Vision Tasks
For image with single object
For image with multiple objects
(Image) classification
Localization (with classification)
Object detection
Image segmentation

Figure 2.5 Simple Computer Vision Tasks

©Sumita Arora

Let us briefly discuss what these CV tasks are :

1. Classification

   image classification. It allows for the classification of a given image as belonging to one of a set of predefined categories or classes. Each class (or category) has some distinct features and characteristics. For example, given an image of a pet, the computer has to identify which pet is it – a dog or a cat ? CV task **image classification** will perform it.

Grapes
Apple
Avocado
Strawberry
Pineapple
Which type of fruit is this ?

1. Please note that we are discussing simple tasks here but there are advanced and other complex computer vision tasks too, which are beyond the scope of the book.

2. Localisation

   This task tells the location or position of the object in an image. The goal of localisation is to find the location of a single object in an image. Localisation of an object takes place along with classification.

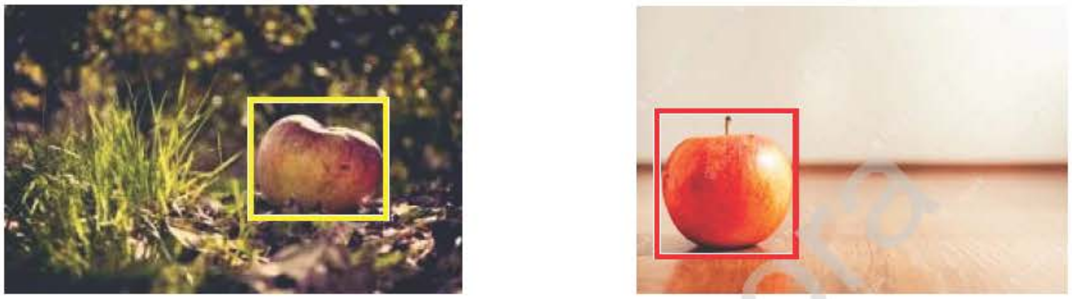


Figure 2.7 Localisation Computer-Vision task

3. Object Detection

   The purpose of object detection is, therefore, to find and then classify a different number of objects in an image.

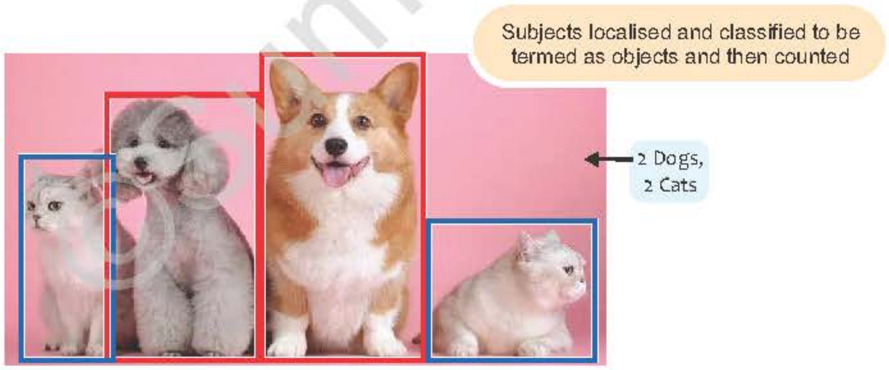

Subjects localised and classified to be termed as objects and then counted
2 Dogs, 2 Cats

Figure 2.8 Object Detection Computer-Vision task

© Smita Arora

4. Image Segmentation

   This task refers to dividing of an image into its sub- components so that objects can be separated from its background and other objects. In other words, an image is separated in different segments in this task.

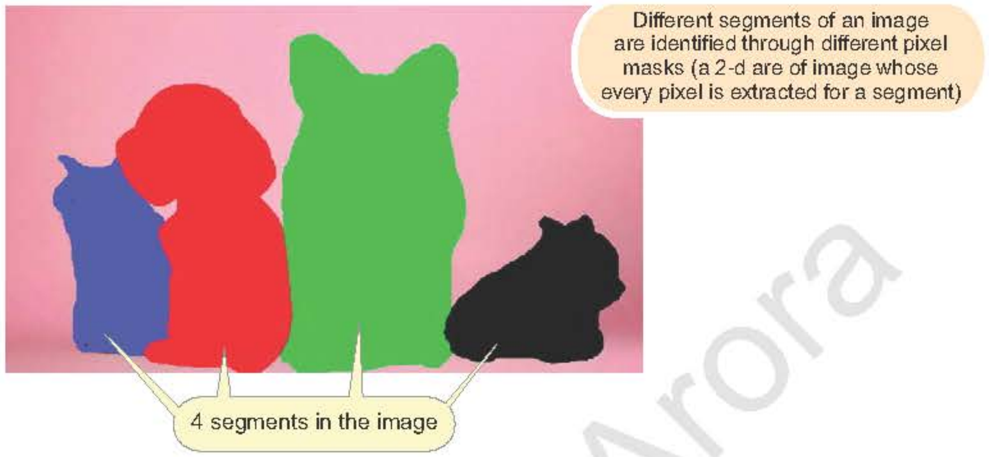

Different segments of an image are identified through different pixel masks (a 2-d are of image whose every pixel is extracted for a segment)
4 segments in the image

Figure 2.9 Image segmentation

2.3 PYTHON OpenCV LIBRARY

OpenCV (Open Source Computer Vision Library) is an open-source library that includes several hundreds of computer vision algorithms. This library is available for a variety of languages such as C, C++, Python etc. The OpenCV library offers various modules, such as : *Core functionality (core), Image Processing (imgproc), Video Analysis (video), 2D Features Framework (features2d), Object Detection (objdetect)* and many others.

In order to work with OpenCV in Python programs or scripts, one needs to import it using import command as :

```
import cv2
```

©Sumita Arora

This section/content is not part of teacher-support PDF

Let Us Revise

- *A pixel (short for picture element) represents the smallest piece of the screen that can be controlled individually in terms of colour and intensity.*
- *A bitmapped graphic is stored as an array of dots, or pixels. Each pixel gets assigned a specific colour.*
- ***Image Resolution** refers to the quality and size of an image in terms of the number of pixels it contains, typically as 'width x height'.*
- *The amount of information that is stored about a pixel determines **its colour depth** (pixel value), which controls how precisely the pixel's colour can be specified.*
- *A **grayscale image** is an image in which the only colours are shades of grey.*
- *An RGB image is a coloured image that is formed by the three-colour information (Red, Green, Blue information) stored for each pixel of a digital image.*
- *RGB value (0,0,0) represents black colour and RGB value (255, 255, 255) represents white colour.*
- *An image feature is a measurable piece of data in your image which is unique to this specific object, such as a distinct colour, specific shape or an image segment*
- *Computer Vision Tasks refer to the internal processing of image that helps in understanding and describing a scene encapsulated in an image, such as identifying, labelling, locating, segmenting the objects in it.*
- *Common computer vision tasks are : classification, localisation, object detection and image segmentation.*
- *Image Classification refers to the act of identifying and classifying a given image as belonging to one of a set of predefined categories or classes.*
- *Localisation refers to identifying the position or location of an object in image. Location performs classification along with localising the object.*
- *Object detection refers to a combined action of localisation and classification, carried out on different items/things/subjects of interest in an image/video to identify and label them as objects of specific types and know their count.*
- *The purpose of object detection is to find and then classify a different number of objects in an image. It is commonly used in applications such as image retrieval and automated vehicle parking systems.*
- *All computer vision tasks are carried out with the help of algorithms. That is computer vision takes place through a combination of algorithms like classification algorithms; localisation algorithms, detection algorithms and segmentation algorithms etc.*
- *OpenCV (Open Source Computer Vision Library) is an open-source library that includes several hundreds of computer vision algorithms.*

©Sumita Arora

SESSION 3

- What is Convolutional Operator ?
- What is CNN ?

Understanding Convolution Operator and CNN

©Sumita Arora

3.2 WHAT IS CONVOLUTIONAL OPERATOR ?

In image processing, ***convolution*** is the process of transforming an image by applying a similar effect through a special array namely **kernel** over each pixel and its local neighbours across the entire image. Here the kernel plays a special role whose size and values depend upon the type of operation being performed.

This section/content is not part of teacher-support PDF

(*i*) Place the *Kernel Matrix* (say ***K***) over each pixel of the image by placing the centre of the kernel matrix over the pixel (*see* Fig. 3.2, step 1) and determine the *mapped matrix* (say ***M***) in the image where the neighbouring pixels are also taken along the source pixel (Fig. 3.2).

(*ii*) Multiply each value of the Kernel with the corresponding pixel of the mapped matrix in the image (*i.e.*, perform *M***K*). (*see* Fig. 3.2, step 2).

(*iii*) Sum the resulting multiplied values, call it **weighted sum.** (*see* Fig. 3.2, step 3)

(*iv*) Store the weighted sum as the new value of the centre pixel. (*see* Fig. 3.2, step 4)

(*v*) Repeat this process across with each pixel in the entire image.

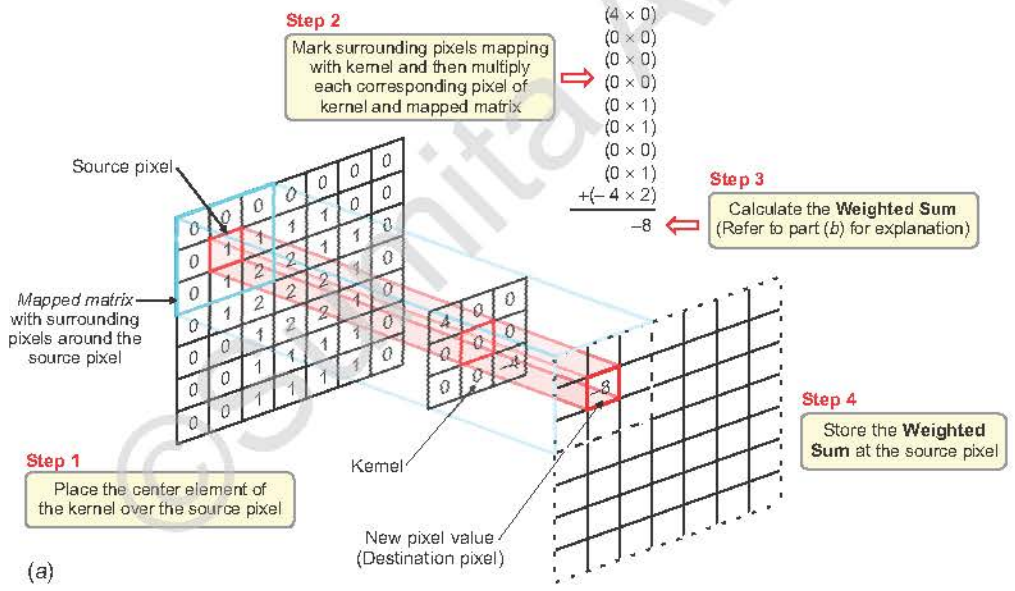

Step 2
Mark surrounding pixels mapping with kernel and then multiply each corresponding pixel of kernel and mapped matrix
(4 × 0)
(0 × 0)
(0 × 0)
(0 × 0)
(0 × 1)
(0 × 1)
(0 × 0)
(0 × 1)
+(– 4 × 2)
–8
Step 3
Calculate the Weighted Sum (Refer to part (b) for explanation)
Source pixel
Mapped matrix with surrounding pixels around the source pixel
Step 4
Store the Weighted Sum at the source pixel
Step 1
Place the center element of the kernel over the source pixel
Kernel
New pixel value (Destination pixel)
(a)

Figure 3.2 (*a*) The convolution operation

©Sumita Arora

Kernel

Kernel plays an important role in convolution as its value determines the final value of then pixel in the image and accordingly the appearance of image changes. A Kernel is a small matrix used for applying various effects on images through convolution such as for blurring, sharpening, embossing, edge detection, and more.

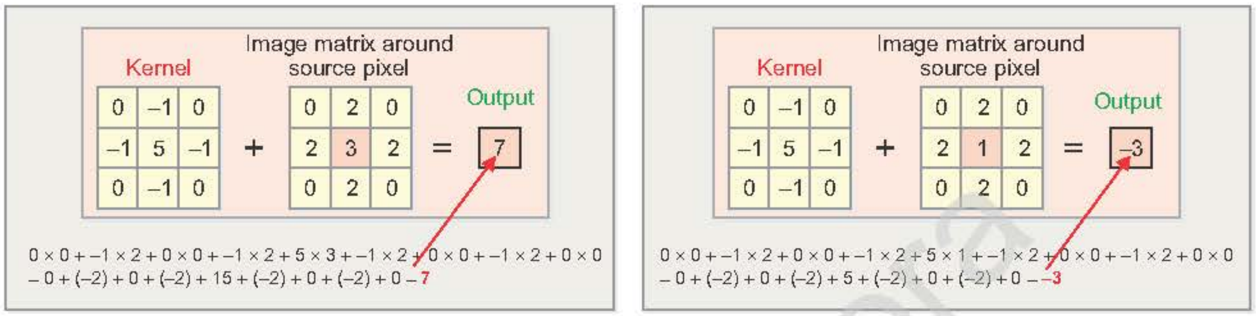


Figure 3.3 The convolution weighted sum value changes with the changing kernel values

©Sumita Arora

This section/content is not part of teacher-support PDF

This section/content is not part of teacher-support PDF

©Sumita Arora

3.3.2 Convolutional Neural Network

Convolutional Neural Networks (CNN) are a type of deep-learning based artificial neural network that are mainly used to process and analyse visual imagery. CNN's deep learning algorithm takes visual inputs (images), extracts and identifies various image features, assigns importance to features and other aspects of the image and finally identifies image distinctly by differentiating it from other.

The CNN architecture consists of mainly the following layers :

- Convolutional layer
- ReLU Layer
- Pooling layer
- Fully Connected Layer

Figure 3.6 shows these four layers of CNN architecture.

CNN

Convolutional Neural Networks (CNNs) are a type of deep-learning based artificial neural network that are mainly used to process, analyse and differentiate visual imagery.

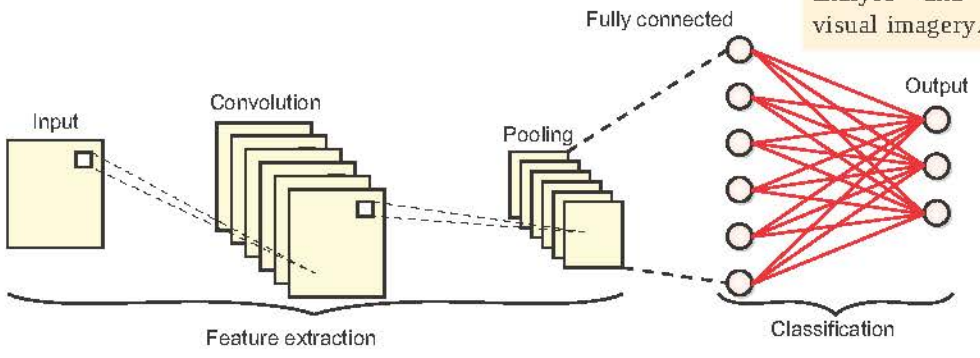
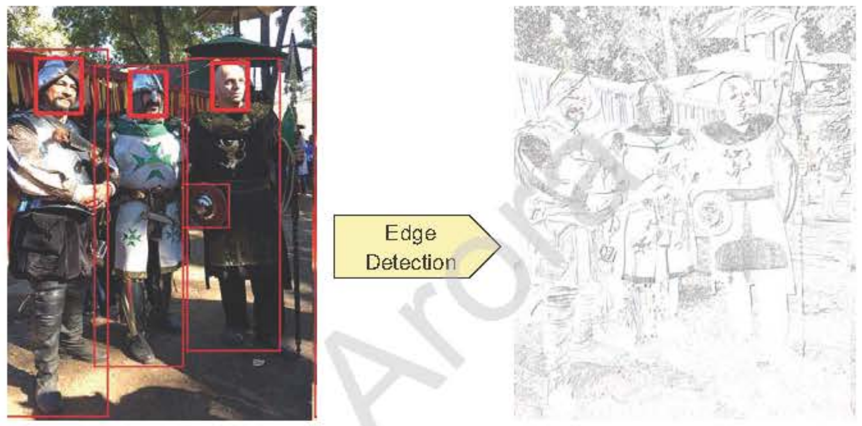


Figure 3.6 Four layers of CNN architecture

1. Convolutional layer

As per its name, this layer uses convolution operation on the images with the help of several kernels to produce several features such as *edges, colour, gradient orientation etc.* This way collected outputs (features) are gathered in the form of **Feature Map** or the **Activation Map**.

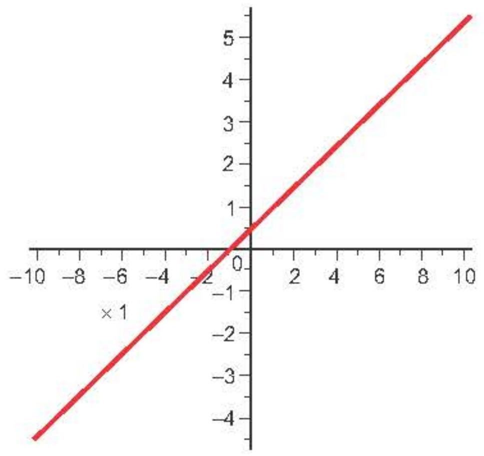
Convolutional layer uses convolution operation on the images with the help of several kernels to produce several features such as edges, colour, gradient orientation etc.

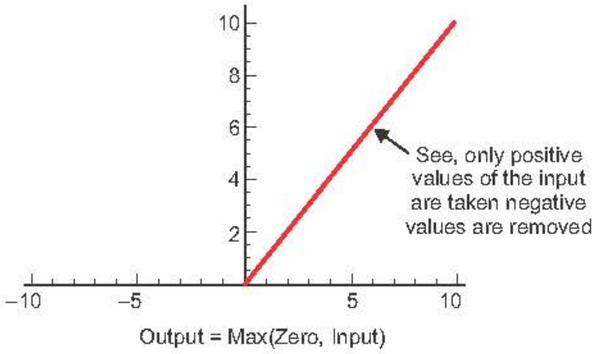


©Sunita Arora

2. ReLU (Rectified Linear activation fUnction) Layer

In a neural network, the activation function is responsible for transforming the summed weighted input from the node into the output for that input. In CNN, the role of activation function is performed by the ***rectified linear activation function*** or **ReLU**. It is a piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero. In other words, it removes the inputs falling in negative range.





By removing the negative input values, ReLU makes the edges more obvious in the images and make them a better feature for processing in further layers.

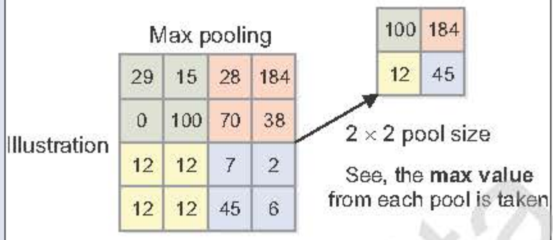
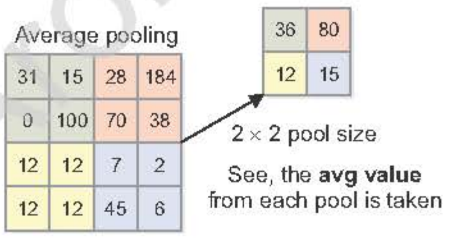
3. Pooling layer

Pooling refers to the down-sampling of the data by bringing the overall dimensions of the images, thus the name. The pooling layer is responsible for downsizing the image while retaining important features.

Two types of pooling is performed :

(*i*) **Maximum Pooling** where the maximum value from a pool is taken (*see* table below).

(*ii*) **Average Pooling** where the average of all values from a pool is taken (*see* table below).

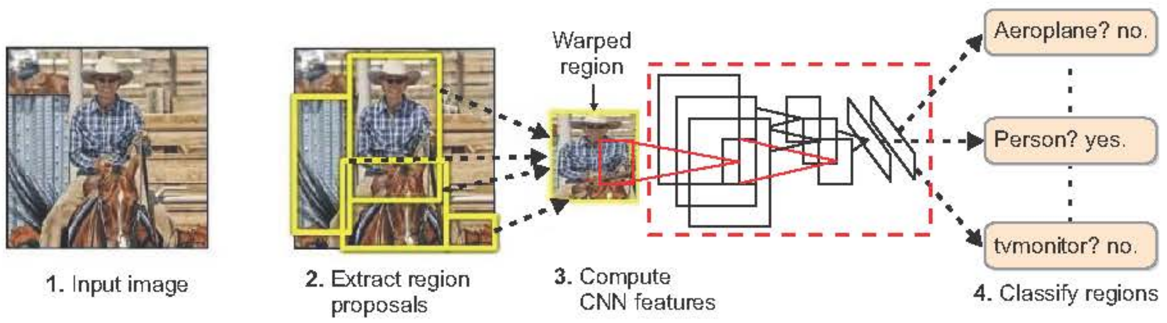
Type	***Max pooling***	***Average pooling***
Purpose	Each pooling operation selects the maximum value of the current view	Each pooling operation averages the values of the current view
Illustration	 Illustration Max pooling 29 15 28 184 0 100 70 38 12 12 7 2 12 12 45 6 100 184 12 45 2 × 2 pool size See, the **max value** from each pool is taken	 Average pooling 31 15 28 184 0 100 70 38 12 12 7 2 12 12 45 6 36 80 12 15 2 × 2 pool size See, the **avg value** from each pool is taken
Comments	★ Preserves detected features ★ Most commonly used	★ Down-samples feature map ★ Used in LeNet

©Sumita Arora

4. Fully Connected layer

This layer performs classification based on the features by previous layer.

Following figure summarises the functioning of a CNN.



Let Us Revise

- *Convolution is the process of transforming an image by applying a similar effect through a special array namely kernel over each pixel and its local neighbours across the entire image.*
- *In convolution operation, the kernel is a matrix of values whose size and values determine the transformation effect of the image through convolution process.*
- *Convolution is one of the most important operations in signal and image processing. It could operate in 1D (e.g., speech processing), 2D (e.g., image processing) or 3D (video processing).*
- *Convolutional Neural Networks (CNN) are a type of deep-learning based artificial neural network that are mainly used to process, analyse and differentiate visual imagery.*
- *The CNN architecture consists of mainly these layers: Convolutional layer, ReLU Layer, Pooling layer, and Fully Connected Layer.*
- *Convolutional layer uses convolution operation on the images with the help of several kernels to produce several features such as edges, colour, gradient orientation etc.*
- *The ReLU layer performs rectified linear activation function that removes the inputs falling in negative range.*
- *The pooling layer down-samples the information of each feature from each convolutional layer by limited them to only containing the most necessary data.*
- *Pooling refers to the down-sampling of the data by bringing the overall dimensions of the images.*
- *Mostly two types of pooling takes place : **Maximum pooling** (where the maximum value of a pool is taken) and **average pooling** (where the average of all values of a pool is taken).*
- *The Fully connected layer performs classification based on the features by previous layer by image flattening, assigning weights, predicting labels and then finally classifying the images based on their labels.*

©Sumita Arora

UNIT 6

Natural Language Processing

IN THIS UNIT

©Sumita Arora

SESSION 1

Introduction to Natural Language Processing

©Sumita Arora

1.2 WHAT IS NLP ?

Natural Language Processing (NLP) is a branch of *artificial intelligence* (AI) that enables computers to process human language in the form of text or voice data and to 'understand' its full meaning. NLP combines computational linguistics with statistical, machine learning, and deep learning models and brings the natural language capacity to computers. Recall using "Okay Google" with your Android phones or interacting with Siri on iPhones or iPads or interacting with Amazon Alexa. All these are examples of NLP.

Natural Language Processing

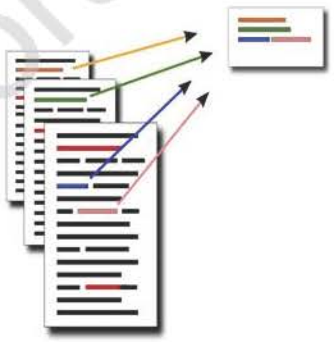
Natural Language Processing (NLP) is a branch of artificial intelligence that enables computers to process human language in the form of text or voice data, 'understand' its full meaning and mimic human conversation.

1.3 NLP APPLICATIONS

NLP based products and applications are all around us. Some common NLP applications are :

1. Automatic Text Summarisation

It is an NLP technique where a computer program shortens longer texts and generates summaries to pass the intended message as the most important information is retained. Text summarisation is most helpfully applied in academic, research, or healthcare settings. Online and digital Newsletters use this for personalised content, agencies apply this on social media posts to make the persona sketch of the person.



Automatic Text Summarisation

Automatic Text Summarisation is an NLP technique where a computer program shortens longer texts and generates summaries to pass the intended message as the most important information is retained.

©Sumita Arora

2. Sentiment Analysis

Sentiment analysis is arguably the most exciting feature of natural language processing. It refers to analysing the text or speech-to-text to recognise sentiment or emotion expressed in it. Through NLP, sentiment analysis categorises words as positive, negative or neutral. NLP enabled sentiment analysis understands the nuances and emotions in human voices and text, giving organisations unparalleled insight.

Sentiment Analysis

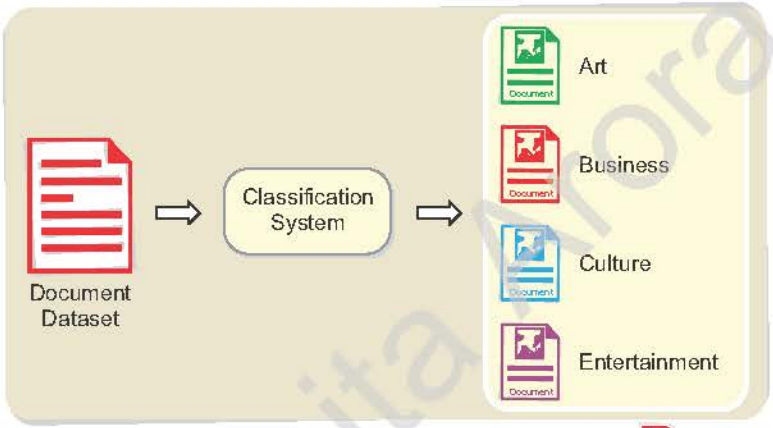
Sentiment Analysis refers to the use of linguistic analysis using AI to detect emotional and language tones in written text or speech-to text.

3. Text Classification

Text classification is the process of understanding, analysing and categorising unstructured text into organised groups (such as *'Sport article'*, *'Technical review'*, *'Art article'*, *'Political Essay'* and so forth) using NLP and other AI technologies. Using natural language processing models, data could be sorted and organised based on predetermined tags and categories. For example, the text classification algorithms are heavily used for email spam filters or to detect hate speeches/posts on social media, detecting political bias through media posts and content, and so forth.

Text Classification

Text Classification is the process of understanding, analysing and categorising unstructured text into organised groups using NLP and other AI technologies based on predetermined tags and categories.



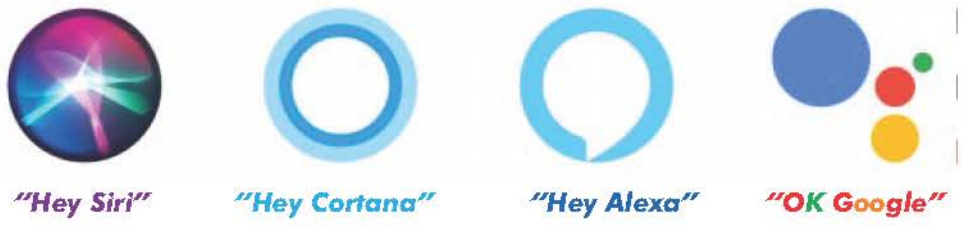
P A R T B

4. Smart Assistants

Think of Siri and Alexa — these virtual smart assistants rely on natural language processing to understand inflection and tone to complete their tasks. These are NLP based virtual assistants which are automated to communicate in human voice, mimicking human interaction.

Virtual Assistants

Virtual Assistants are NLP based programs that are automated to communicate in human voice, mimicking human interaction to help ease your day-to-day tasks, such as showing weather reports, creating remainders, making shopping lists etc.



Virtual assistants are NLP based programs that help you ease your day-to-day tasks, such as showing weather reports, creating reminders, making shopping lists etc. They can take voice or text commands or an invoking word or wake word to activate the listener, followed by the command (*e.g.*, '**Alexa, What is the capital of India?**'). Today we see so many virtual assistants around us, such as Apple's *Siri*, Amazon's *Alexa*, *Google Assistant* and Microsoft's *Cortana* and many others.

©Sumita Arora

5. Digital Phone Calls

We all hear *"this call may be recorded for training purposes,"* but rarely do we wonder what goes behind that. Other than for training, these recordings go into the database for an NLP system to learn from and improve in the future. Automated systems direct customer calls to a service representative or online chatbots, which respond to customer requests with helpful information. This is an NLP practice that many companies, including large telecommunications providers have put to use.

Note

Modern day **chatbots** that you often see in the form of a text box when you open a website or contact a customer service and that interact with you in the text box and based on the words used by you, they either connect you with a customer support executive or redirect to another webpage or link, are classic example of NLP applications today.

©Sumita Arora

This section/content is not part of teacher-support PDF

Let Us Revise

- *Natural Language Processing (NLP) is a branch of artificial intelligence or AI that enables computers to process human language in the form of text or voice data and to 'understand' its full meaning.*
- *Some common NLP applications are : Automatic Text Summarisation, sentiment analysis, text classification, smart assistants, digital phone calls and so forth.*
- *Automatic Text Summarisation is an NLP technique where a computer program shortens longer texts and generates summaries to pass the intended message as the most important information is retained.*
- *Sentiment Analysis refers to use of linguistic analysis using AI to detect emotional and language tones in written text or speech-to text.*
- *Text classification is the process of understanding, analysing and categorising unstructured text into organised groups using NLP and other AI technologies based on predetermined tags and categories.*
- *Virtual assistants are NLP based programs that are automated to communicate in human voice, mimicking human interaction to help ease your day-to-day tasks, such as showing weather reports, creating remainders, making shopping lists etc.*
- *Modern day chatbots are classic example of NLP applications today.*

This section/content is not part of teacher-support PDF

©Sumita Arora

SESSION 2

Revisiting AI Project Cycle (NLP)

- AI Project Cycle for an NLP Application
- Understanding Chatbots

©Sumita Arora

This section/content is not part of teacher-support PDF

This section/content is not part of teacher-support PDF

2.3 UNDERSTANDING CHATBOTS

A chatbot (also called **bot** sometimes) is a computer program which mimics conversation between users, usually powered by artificial intelligence (the program behind a chatbot uses AI based algorithm(s)).

2.3.1 Types of Chatbots

Broadly chatbots are categorised into following *two* categories :

1. Simple Chatbots (Script bots)

Script chatbots are very simple and have limited capabilities, and are usually rule-based bots. They are task-specific where the bot poses questions based on predetermined options and the customer can choose from the options until they get answers to their query.

These chatbots have programmed scripts, which are based on the predefined rules and do not use any AI to automatically learn new things. Thus, these chatbots will not make any inferences from its previous interactions. These chatbots are best suited for straightforward dialogues. They are very simple to build and train.

Many websites use these types of chatbots. For example, when you order on *Zomato* or *Swiggy*, you may encounter such type of chatbots where the chatbot may ask your preference of food type, for how many number of people, additional requirements etc. and place an order.

Script Bots

Script Bots are simple chatbots which use scripts based on pre-defined text rules and do not use AI to learn and adapt to new things.

©Sumita Arora

2. Smart Chatbots (AI based Smart bots)

AI-enabled smart chatbots are designed to simulate near-human interactions with customers. They can have free-flowing conversations and understand intent, language, and sentiments. These chatbots require use of AI with programming to help it understand the context of interactions. They are much harder to implement and execute, and need a lot of data to learn.

Smart Bots

Smart Bots are AI enabled chatbots that mimic human conversations, understand intent and context of the conversation and keep learning and adapting with new things.

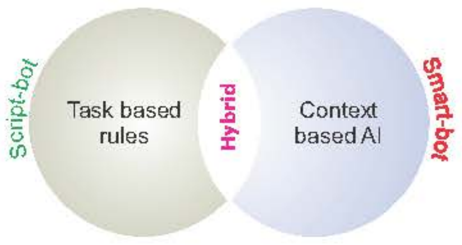


Figure 2.3 Types of Chatbots

For example, you must have interacted with Siri or Google assistant through chat. This is an example of a smart bot, where it learns from every human interaction. These can give you most relevant response as per your context, considering your preferences or past choices, because these use AI based programs.

3. Hybrid Chatbots

Hybrid chatbots combine the features of both the above types of chatbots. Hybrid chatbots have some rule-based tasks, and they can also understand intent and context of the chat.

©Sumita Arora

This section/content is not part of teacher-support PDF

Let Us Revise

- *Like any other AI project, an NLP project also undergoes the phases of AI project cycle.*
- *After collecting data for a data science project, data is pre-processed where it is cleaned, standardised and normalised.*
- *Then the data is analysed and visualised to know about the trends and key characteristic of data.*
- *Then for data modelling, the treated dataset is divided into training dataset and testing dataset and the AI based model is trained using the training dataset using one of the training techniques - supervised/unsupervised or reinforced learning.*
- *For model evaluation, the testing dataset is used and the predicted values are compared with the actual results to determine the efficacy and efficiency of the model developed and trained.*
- *A chatbot is a computer program that simulates and processes human conversation (either written or spoken), allowing humans to interact with digital devices as if they were communicating with a real person.*
- *There are mainly three types of chatbots : script bots (rule-based scripts), smart bots (AI enabled programs) and hybrid.*

©Sumita Arora

This section/content is not part of teacher-support PDF

SESSION 3

Human vs. Computer-Languages and NLP

- Human Language vs. Computer Language
- Concepts of NLP
- Bag of Words (BoW)

©Sumita Arora

This section/content is not part of teacher-support PDF

This section/content is not part of teacher-support PDF

©Sumita Arora

3.3.1 Text Normalisation

As mentioned, the text normalisation simplifies the text (text input or speech-to-text result) for further processing. Basically, the text normalisation divides the text into smaller components called **tokens** (usually the words in the text) and groups related tokens together.

Text Normalisation

Text Normalisation is a process to reduce the variations in text's word forms to a common form when the variations mean the same thing.

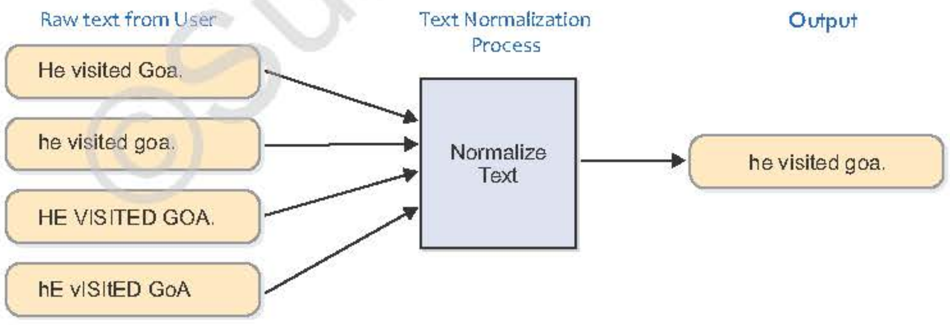


Figure 3.1 Text Normalisation

There are many steps that are performed in the text normalisation process. But not all the text types need all the steps. Broadly the steps in text normalisation are listed below.

3.3.1A Sentence Segmentation

The sentence segmentation is the process of dividing the whole text into smaller components, *i.e.*, individual sentences. This is done to understand the thought or idea of each individual sentence. *For example*,

Sentence Segmentation

Sentence Segmentation is the process of dividing the whole text into smaller components, *i.e.*, individual sentences.

```
Hello world. AI is fun
to know. It has starting
impacting our lives in
many ways. Many more
revolutionary technologies
will soon evolve out of it.
```

$\Rightarrow$

- Hello world.
- AI is fun to know.
- It has starting impacting our lives in many ways.
- Many more revolutionary technologies will soon evolve out of it.

3.3.1B Tokenisation

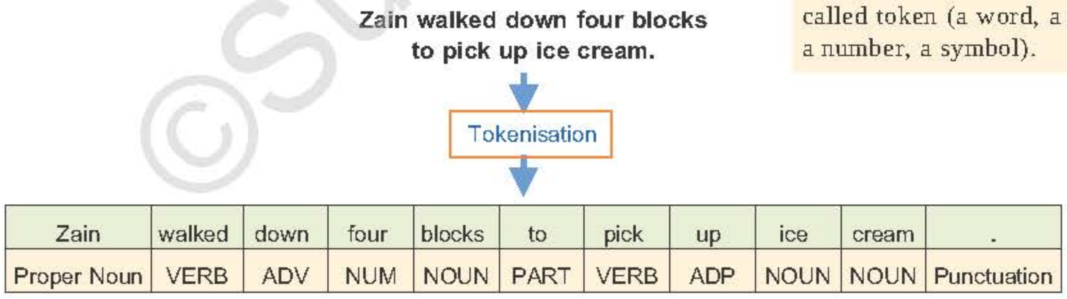
Tokenisation is the next step where the sentences are further divided into smaller units called tokens, which are words, phrases, numbers or symbols (special characters like :, –, . etc.). The individual basic units in a sentence are to understand and analyse the content.

Figure 3.2 illustrates the tokenisation process.

Token & Tokenisation

A **Token** is a well-defined semantic unit inside a sentence and contributes to overall meaning of the sentence. A token may represent a word, a phrase, a number or a symbol.

Tokenisation is the process of splitting up of individual sentences into smaller units called token (a word, a phrase, a number, a symbol).



Zain	walked	down	four	blocks	to	pick	up	ice	cream	.
Proper Noun	VERB	ADV	NUM	NOUN	PART	VERB	ADP	NOUN	NOUN	Punctuation

Figure 3.2 Tokenisation

©Sumita Arora

This section/content is not part of teacher-support PDF

3.3.1C Removing Stop Words, Special Characters and Numbers

The function of this step is to find unimportant words as per the overall meaning and retain the important words in the text. For example, consider the following two sentences which are conveying the same meaning to the computer (Not as per grammar, though).

```
Delhi is the capital and the most populous city.
Delhi capital most populous city
```

Tell me, won't the computer understand the same meaning from both the above ? And this is what this step does, *i.e.*, removes the words that if removed, won't affect the meaning of the sentence. In other words, it removes the filler words that appear very frequently like *"and"*, *"the"*, and *"a"*. These words are often termed as *"stop words"* in NLP.

This step, along with removing the stop words, also removes the redundant special characters and numbers, which are not contributing to the overall meaning.

Stop Words

Stop Words are the words in any language which do not add much meaning to a sentence. They can safely be ignored without sacrificing the meaning of the sentence.

3.3.1D Stemming and Lemmatisation

Stemming

Stemming means converting a word into its stem (root form). In other words, the process of extracting the root form of the word is known as stemming. For NLP, Stemming takes place by removing prefixes and suffixes at the beginning or end of each word, *e.g.*,

Word in the sentence	*Stem*	*Remarks*
Studied	studi	affix ed removed
Standardize	standard	affix ize removed
Simplified	simplifi	affix ed removed
Drives	drive	affix s removed

Stemming

The process of extracting the root form of the word by removing affixes, is known as **stemming**. The words extracted through stemming are called *stems*.

Please note that the stems obtained this way (refer to above table) are *not actual words*. This is because the **stem** (root) is the part of the word to which the inflectional (*changing/deriving*) affixes are added such as (-ed, -ize, -s, -de, mis etc.). So, stemming a word or sentence may result in words that are not actual words. Stems are created by removing the suffixes or prefixes used with a word.

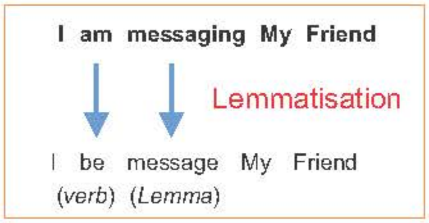
Stem

A **Stem** (root) is the part of the word to which the inflectional (changing/deriving) affixes are added such as (*-ed*, *-ize*, *-s*, *-de*, *mis* etc.) A stem may or may not be equal to a dictionary word.

©Sumita Arora

Lemmatisation

Lemmatisation is the process of converting a word to its actual root form linguistically (as per the language). Lemmatisation ensures that the root word belongs to the language. In Lemmatisation, the root word is called ***Lemma*** and it actually exists in the language dictionary. *For example,*



Lemmatisation

Lemmatisation is the process of converting a word to its actual root form linguistically (as per the language). The words extracted through lemmatisation are called **lemmas**.

Lemma

A **Lemma** is the base, root form of an inflectional word and that exists in a dictionary unlike stems.

Unlike *stem*, a *lemma* is a real word and exists in the dictionary ; stem, on the other hand, may or may not exist in the language dictionary.

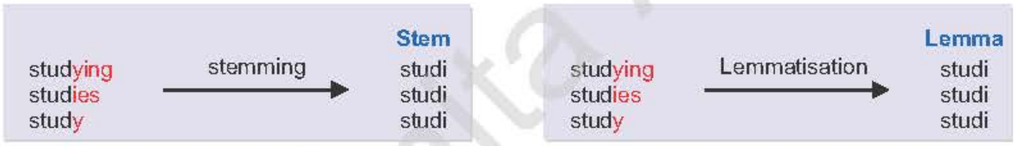


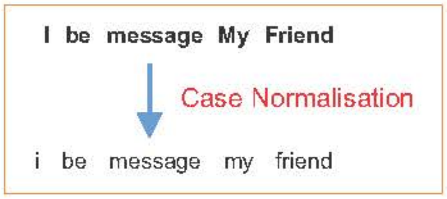
Figure 3.3 Stemming vs. Lemmatisation

©Sumita Arora

This section/content is not part of teacher-support PDF

3.3.1E Case Normalisation

In this step, the case of all words is changed so that all words are in the same case. Mostly, these are converted to *lower case*. This is done to ensure that if any machine is case sensitive, it should not affect the overall result by considering two same words in different cases as different words.



Case Normalisation

Case Normalisation refers to conversion of all the words in the same case (often lowercase).

3.4 BAG OF WORDS (BoW)

A Bag-of-Words (BoW) model is a way of extracting features from text for use in modelling, with many AI algorithms. A bag-of-words is a representation of text that describes the occurrence of words within a document.

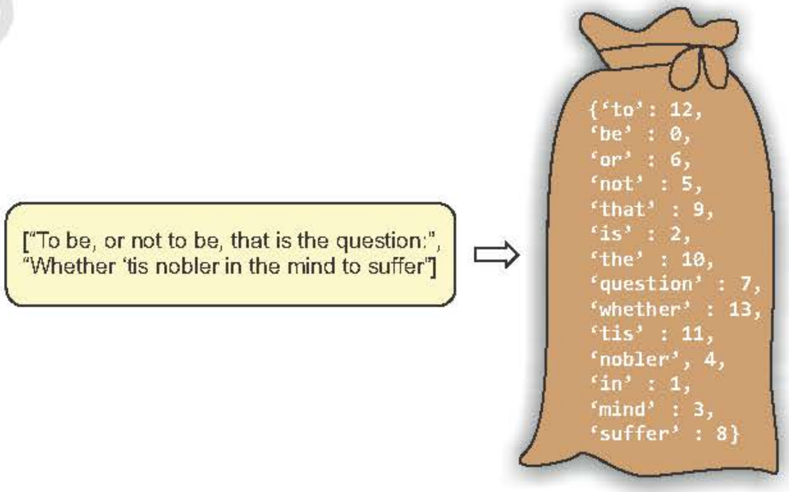
A Bag of Words contains *two* things (Fig. 3.4.) :

(*i*) A vocabulary of known words

(*ii*) A measure of the presence of known words (such as *frequency of words*)

It is called a "**bag" of words**, because it contains just the collection of words without any information about the order or structure of words in the document. It only tells that the known words occur in the document, not their position in the document.

©Sumita Arora


["To be, or not to be, that is the question:", "Whether 'tis nobler in the mind to suffer"]

```
{'to': 12,
'be' : 0,
'or' : 6,
'not' : 5,
'that' : 9,
'is' : 2,
'the' : 10,
'question' : 7,
'whether' : 13,
'tis' : 11,
'nobler', 4,
'in' : 1,
'mind' : 3,
'suffer' : 8}
```

Figure 3.4 A Bag of Words

3.4.1 Steps to Implement Bag-of-Words Model

Step 1 : Text Normalisation

In the first step, the data is to be collected and then pre-processed. Say, we have the following text available :

```
"I am in the Search of Wisdom.
 I am in the Age of Wisdom.
 But sometimes I feel otherwise.
 As if I am in the Age of Foolishness."
```

Dividing in individual paragraphs (Pre-processing 1)

```
"I am in the Search of Wisdom."
"I am in the Age of Wisdom."
"But sometimes I feel otherwise."
"As if I am in the Age of Foolishness."
```

- "am" is converted to its basic verb form
- "the" being a *stop word* is removed. (Please note that after text normalisation, not all the stop words get removed. Some of them still remain present.)
- Case of all words converted to lower case
- Punctuation marks removed
- In this, wrong spellings and slags are converted to real words, *e.g.*, "goood" and "gud" would be replaced with "good"; similarly, "right" and "rite" will be replaced with "right"; "2morrow" with "tomorrow" and so on. In our example above, there are no misspellings or slags, so this step is not required.

So, our corpus of documents, after pre-processing all the documents, now contains :

```
Document 1: "i be in search of wisdom."
Document 2: "i be in age of wisdom"
Document 3: "but sometimes i feel otherwise"
Document 4: "as if i be in age of foolishness"
```

Step 2 : Design the Vocabulary

Now we can make a list of all of the words in our model vocabulary with the unique words left after pre-processing of all the documents. So, the unique words in our Dictionary for the corpus will be :

```
"i", "be", "in", "search", "of", "wisdom", "age", "but", "sometimes", "feel",
"otherwise", "as", "if", "foolishness"
```

So our dictionary contains a vocabulary of 14 words from a corpus containing 28 words.

©Sumita Arora

P A R T B

Step 3 : Create Document Vectors

The next step is to score the words in each document by recording the frequency of words. For this we shall take our dictionary of words and check how many times the dictionary's word appears in that document. Number of times a word occurs in a document is known as its frequency.

For instance, for the document1 from our corpus above, the ***document-vector*** will be :

"i"	"be"	"in"	"search"	"of"	"wisdom"	"age"	"but"	"sometimes"	"feel"	"otherwise"	"as"	"if"	"foolishness"
1	1	1	1	1	1	0	0	0	0	0	0	0	0

"i"	"be"	"in"	"best"	"of"	"times"	"worst"	"age"	"wisdom"	"foolishness"
1	1	1	1	1	1	0	0	0	0

So, all the four document vectors will be as shown below in Document Vector Table.

Document Vector Table

	i	be	in	search	of	wisdom	age	but	sometimes	feel	otherwise	as	if	foolishness
Doc1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
Doc2	1	1	1	0	1	1	1	0	0	0	0	0	0	0
Doc3	1	0	0	0	0	0	0	1	1	1	1	0	0	0
Doc4	1	1	1	0	1	0	1	0	0	0	0	1	1	1
Total	4	3	3	1	3	2	2	1	1	1	1	1	1	1

©Sumita Arora

Step 4 : Calculate TF-IDF

In the final steps, we want that our data be converted into numbers so that these can be interpreted by the computer.

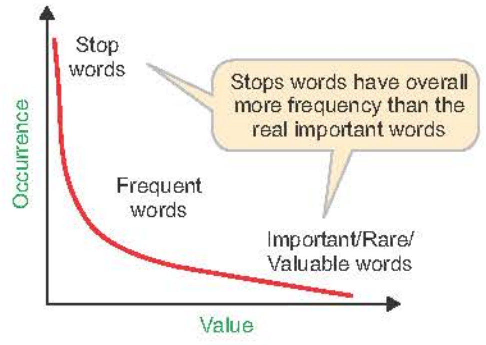


Figure 3.5 Frequency of Stop Words *vs*. Rare Words

1. TF (Term Frequency)

It is the frequency of a word in one document. As the document vector table stores the frequency of each word of the vocabulary in each document, we can take this value from *Document vector table* easily. Thus, for our corpus shown above, the TF will be :

TF (Term Frequency) for the corpus, extracted from *Document Vector Table*

i	be	in	search	of	wisdom	age	but	sometimes	feel	otherwise	as	if	foolishness
1	1	1	1	1	1	0	0	0	0	0	0	0	0
1	1	1	0	1	1	1	0	0	0	0	0	0	0
1	0	0	0	0	0	0	1	1	1	1	0	0	0
1	1	1	0	1	0	1	0	0	0	0	1	1	1

2. IDF (Inverse Document Frequency)

It refers to how common or rare a word is in the entire document set. This metric is calculated as by taking the total number of documents, dividing it by the number of documents that contain a word, and calculating the logarithm. See below for explanation.

(*i*) Take total frequency of each word in total corpus (all the documents), *i.e.*,

	i	be	in	search	of	wisdom	age	but	sometimes	feel	otherwise	as	if	foolishness
Total	4	3	3	1	3	2	2	1	1	1	1	1	1	1

(*ii*) Now divide the total number of documents in the corpus (4 in our case) with this total frequency, *i.e.*,

IDF for each word (W)

i	be	in	search	of	wisdom	age	but	sometimes	feel	otherwise	as	if	foolishness
4/4	4/3	4/3	4/1	4/3	4/2	4/2	4/1	4/1	4/1	4/1	4/1	4/1	4/1

(*iii*) Now calculate the TFIDF for word (W) as per the formula :

```
TFIDF(W) = TF(W) * log (IDF(W))
```

To calculate the log, use the calculator and this way, the **log(IDF(W)** will be as shown in the table below.

TF-IDF

TF-IDF (term frequency-inverse document frequency) is a statistical measure that evaluates how relevant a word is to a document in a collection of documents.

Calculated values of Log(IDF(W)

i	be	in	search	of	wisdom	age	but	sometimes	feel	otherwise	as	if	foolishness
0	0.124	0.124	0.602	0.124	0.301	0.301	0.602	0.602	0.602	0.602	0.602	0.602	0.602

The closer this value **log(IDF(W))** is to 0, the more common a word is, *e.g.*, in the above corpus, 'I', is the most common word, followed by 'be', 'in', 'of',

©Sumita Arora

Finally, let us calculate TF-IDF for our corpus by multiplying the **TF** with **log(IDF)**

TF(W) * log (IDF(W)) values

i	be	in	search	of	wisdom	age	but	sometimes	feel	otherwise	as	if	foolishness
1*0	1*0.124	1*0.124	1*0.602	1*0.124	1*0.301	0*0.301	0*0.602	0*0.602	0*0.602	0*0.602	0*0.602	0*0.602	0*0.602
1*0	1*0.124	1*0.124	0*0.602	1*0.124	1*0.301	1*0.301	0*0.602	0*0.602	0*0.602	0*0.602	0*0.602	0*0.602	0*0.602
1*0	0*0.124	0*0.124	0*0.602	0*0.124	0*0.301	0*0.301	1*0.602	1*0.602	1*0.602	1*0.602	0*0.602	0*0.602	0*0.602
1*0	1*0.124	1*0.124	0*0.602	1*0.124	0*0.301	1*0.301	0*0.602	0*0.602	0*0.602	0*0.602	1*0.602	1*0.602	1*0.602

Finally the TF-IDF values computed are :

i	be	in	search	of	wisdom	age	but	sometimes	feel	otherwise	as	if	foolishness
0	0.124	0.124	0.602	0.124	0.301	0	0	0	0	0	0	0	0
0	0.124	0.124	0	0.124	0.301	0.301	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0.602	0.602	0.602	0.602	0	0	0
0	0.124	0.124	0	0.124	0	0.301	0	0	0	0	0.602	0.602	0.602

Final inference from the BoW and TF-TDF can be summarised in terms of the following:

(*i*) Words with high TF (Term frequency) values have the least values in TF-IDF and these most occurring words are actually the ***stop words*** and make no contribution towards the real message/intent of the text.

(*ii*) Words with high TF-IDF value are :

(*a*) the important words

(*b*) and not commonly repeated words across the documents

(*iii*) Thus, the computer considers the words with high TF-IDF values as the important and valuable words for finding the intent for NLP. The higher the TF-IDF value, the more important the word is for a given corpus.

Stop Words have high TF (Term frequency) values and low TF-IDF values. Stop words do not contribute to the actual message/ intent of the text.

TF-IDF values help the computer understand which words are to be considered during NLP processing.

©Sumita Arora

Let Us Revise

- *Both human languages and computer languages have syntax and semantics and a specific structure.*
- *But human languages have* ***morphology****,* ***context-sensitivity*** *and convey the* ***intent*** *even when mispronounced, whereas computer languages have no morphology and no context-sensitivity.*
- *NLP models mostly use Text Normalisation to reach to the intent of the message.*
- *Text normalisation is a process to reduce the variations in text's word forms to a common form when the variations mean the same thing.*
- *Text normalisation uses the following steps :* ***sentence segmentation, tokenisation, removing punctuation & stop words, case normalisation, stemming and lemmatisation.***
- *Sentence segmentation is the process of dividing the whole text into smaller components, i.e., individual sentences.*
- *The whole collection of words from all the documents being processed, is called* ***corpus.***
- *A token is a well-defined semantic unit inside a sentence and contributes to overall meaning of the sentence. A token may represent a word, a phrase, a number or a symbol.*
- *Tokenisation is the process of splitting up of individual sentences into smaller units called token (a word, a phrase, a number or a symbol).*
- *Tokenisation splits up the words, numbers and punctuation symbols, and delivers the building blocks of the text, crucial for NLP.*
- *Stop words are the words in any language which do not add much meaning to a sentence. They can safely be ignored without sacrificing the meaning of the sentence.*
- *The process of extracting the root form of the word, by removing affixes, is known as stemming. The words extracted through stemming are called stems.*
- *A Stem (root) is the part of the word to which the inflectional (changing/deriving) affixes are added such as (-ed, -ize, -s, -de, mis etc.)*
- *Lemmatisation is the process of converting a word to its actual root form linguistically (as per the language). The words extracted through lemmatisation are called lemmas.*
- *A lemma is the base form of all its inflectional forms, whereas a stem isn't. This is why regular dictionaries are lists of lemmas, not stems.*
- *Case normalisation refers to conversion of all the words in the same case (often lowercase).*
- *Case normalisation ensures that the case-sensitivity of the machine does not consider same words as different because of their having different cases.*
- *A Bag-of-Words model (BoW) is a model used for extracting features from text for use in modelling, with many AI algorithms.*
- *TF-IDF (term frequency-inverse document frequency) is a statistical measure that evaluates how relevant a word is to a document in a collection of documents.*

©Sumita Arora

UNIT 7

Evaluation

IN THIS UNIT

©Sumita Arora

SESSION 1

Model Evaluation

Evaluating an AI Model

©Sumita Arora

1.2 EVALUATING AN AI MODEL

The final task is to test the trained AI model on new real-life data and see how it is performing. The aim of evaluating an AI model, which has been developed and trained during modelling, is to measure its performance through some evaluation metrics. These metrics are able to evaluate a model and calculate some performance score, based on which the efficiency of an AI model is determined.

The purpose of these functions is to provide a mathematical estimate as to how far we are from making correct predictions. Finally, if the model performs well on unseen new real-life data, the deployment stage is started where it is put to use in real-life application.

1.2.1 Model Evaluation Metrics

Recall from *Session 7, Unit 2* that **metrics** is standardised way of measurement to assess something for accuracy and performance. For AI models too, there are metrics to test and evaluate a developed AI model.

There are different types of metrics used to assess the AI models, such as :

- ***Classification Metrics*** (*Confusion Matrix, Accuracy, Precision, Recall, F1-score, ...*) for evaluating classification based AI models ;
- ***Regression Metrics*** (*MSE, MAE, ...*) for evaluating regression based AI models ;
- ***Deep Learning Related Metrics*** (*Inception score, Frechet Inception distance, ...*) for evaluating deep-learning based AI models.

1.2.2 Confusion Matrix

Recall that a **Confusion Matrix** is a technique using a chart or table for summarizing the performance of a classification based AI model by listing the predicted values of an AI model and the **actual/correct outcome values** in a confusion table.

- the Actual Value (True/False) represents the actual result of the AI model (observed or measured).
- the Predicted Value (Positive/Negative) is the value of the outcome/result of the AI model, produced on the basis of its algorithm and learning.

Using these values, you create a confusion matric for your AI model in the following format :

Confusion Matrix Format

Actual Values	**Predicted Values**	
	Positive	***Negative***
Positive (1)	No. of ***True Positives*** (*TP*)	No. of ***False Negatives*** (*FN*) ***Type II error***
Negative (0)	No. of ***False Positives*** (*FP*) ***Type I error***	No. of ***True Negatives*** (*TN*)

©Sumita Arora

This section/content is not part of teacher-support PDF

Using the confusion matrices, you need to compute the following values to evaluate an AI model :

- **Accuracy rate.** This is the percentage of times the predictions out of all the observations are correct.

  The Formula to determine **Accuracy** is :

$$\text{Accuracy} = \frac{\text{Number of correct predictions } (TP+TN)}{\text{Total number of predictions made } (TP+TN+FP+FN)} \times \mathbf{100\%}$$

- **Precision rate.** This is the rate at which the desirable predictions turn out to be correct (True Positives out of all positives).

  The formula for **Precision** rate is : $\dfrac{TP}{(TP+FP)}$

  In percentage, Precision rate is : $\dfrac{TP}{(TP+FP)} \times 100\%$

©Sumita Arora

- **Recall.** It is a rate of correct positive predictions to the overall number of positive instances in the dataset.

  The formulas for calculating Recall are :

$$\text{Recall} = \frac{\text{Predictions actually positive}}{\text{Actual positive values in the dataset}} = \frac{TP}{(TP+FN)}$$

$$\text{(In percentage)} = \frac{TP}{(TP+FN)} \times 100\%$$

- **F1 Score.** It is a measure of balance between precision and recall. It is computed as per the following formula :

$$F1 = 2.\frac{\text{precision.recall}}{\text{precision + recall}} = \frac{TP}{TP+\frac{1}{2}(FP+FN)}$$

Example 1.1 *Given the **confusion matrix** for an AI model below, compute all the classification metrics (**accuracy, precision, recall, f1 score**) for the model.*

Confusion Matrix

Actuals	61 TN	3 FP
	3 FN	104 TP

Solution

Accuracy $= \dfrac{(TP + TN)}{(TP + FN + TN + FP)}$

$$= \frac{(104 + 61)}{(104 + 3 + 61 + 3)} = \frac{165}{171} = \mathbf{0.965}$$

Precision $= \dfrac{TP}{(FP + TP)}$

$$= \frac{104}{3 + 104} = \frac{104}{107} = \mathbf{0.972}$$

Recall $= \dfrac{TP}{(FN + TP)}$

$$= \frac{104}{3 + 104} = \frac{104}{107} = \mathbf{0.972}$$

F1 Score $= \dfrac{2 * \text{precision} * \text{recall}}{\text{precision} + \text{recall}}$

$$= \frac{2 * 0.972 * 0.972}{0.972 + 0.972} = \frac{1.89}{1.944} = \mathbf{0.972}$$

Since F1 Score is high, we can say that this AI model will work efficiently.

©Sumita Arora

Let Us Revise

- *There are different types of metrics used to assess the AI models, such as :* ***Classification Metrics*** *(Confusion Matrix, Accuracy, Precision, Recall, F1-score, ...) for evaluating classification based AI models;* ***Regression Metrics*** *(MSE, MAE) ...) for evaluating regression based AI models; and* ***Deep Learning Related Metrics*** *(Inception score, Frechet Inception distance) ...) for evaluating deep-learning based AI models.*
- *The classification metrics are simple and can be applied to evaluate varied types of models.*
- *A Confusion Matrix is a technique using a chart or table for summarising the performance of a classification based AI model by listing the predicted values of an AI model and the actual/correct outcome values.*
- *In* ***True Positive (TP)****, both predicted value of the AI model and actual value are positive.*
- *In* ***True Negative (TN)****, both predicted value of the AI model and actual value are negative.*
- *In* ***False Positive (FP)****, the predicted value of an AI model is positive but actual value is negative.*
- *In* ***False Negative (FN)****, the predicted value of an AI model is negative but actual value is positive.*
- ***Accuracy rate*** *is the percentage of times the predictions out of all the observations are correct.*
- ***Precision rate*** *is the rate at which the desirable predictions turn out to be correct (True Positives out of all positives).*
- ***Recall*** *is a rate of correct positive predictions to the overall number of positive instances in the dataset.*
- ***F1 Score*** *is a measure of balance between precision and recall.*

©Sumita Arora