**A Project/Dissertation Review-ETE
Report**

on

FACE DETECTION SYSTEM

Submitted in partial fulfillment of

the requirement for the award of

the degree of

B.Tech In Computer Science and Engineering

(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

Under The Supervision of

Mr Abhay Kumar
**Designation Asst.
Professor**
Submitted By

Hariom Kumar
19021011759
19SCSE1010601
Shivam
19021011758
19SCSE1010600

**SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE AND
ENGINEERING, GALGOTIAS UNIVERSITY, GREATER NOIDA
INDIA MONTH, YEAR**
<u>**CERTIFICATE**</u>

The Final Thesis/Project/ Dissertation Viva-Voce examination Hariom kumar& Shivam has been held on and his/her work is recommended for the award of Btech in Computer Science and Engineering.

Signature of Examiner(s)   **Signature of Supervisor(s)**

Signature of Project Coordinator   **Signature of Dean**

Date:

Place: Greater Noida

Acknowledgement

I/We hereby certify that the work which is being presented in the thesis/project/dissertation, entitled

"FACE DETECTION SYSTEM" in partial fulfillment of the requirements for the award of the

Btech in Computer Science and Engineering submitted in the School of Computing Science and Engineering of Galgotias University, Greater Noida, is an original work carried out during the period of month, Year to Month and Year, under the supervision of Name… Designation, Department of Computer Science and Engineering/Computer Application and Information and Science, of School of Computing Science and Engineering , Galgotias University, Greater Noida

The matter presented in the thesis/project/dissertation has not been submitted by me/us for the award of any other degree of this or any other places.

Hariom kumar(19SCSE1010601

Shivam(19SCSE1010 600

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Supervisor Name

Mr Abhay Kumarj.

Designation Asst. Professor

ABSTRACT

Face appearance changes over a amount of your time and mirror as ageing, that is yet one more challenge in biometric authentication system. With the increasing age, the external body part options, shapes/lines, and different aspects additionally amendment. it's finished visual observation and image retrieval once an extended amount.

For accuracy checking, the dataset for a special people of individuals over a amount of your time is calculated. Here, the popularity method depends on feature extraction, basic options like wrinkles, marks, eyebrows, hairstyles, etc.

First face detection is employed to phase the face from the image background. within the second step the metameric face image is aligned to account for face cause, image size and photographic properties, like illumination and grayscale. the aim of the alignment method is to change the correct localization of facial expression within the third step, the facial feature extraction.

Features like eyes, nose and mouth area unit pinpointed and measured within the image to represent the face. The therefore established feature vector of the face is then, within the fourth step, matched against a info of faces.

In current work we have a tendency to tend to developed the system to guage the face detection and recognition ways that area unit thought-aboutto be a bench mark. Some ways performed consistently over fully completely different datasets whereas different ways behave very haphazardly however supported average experimental results performance is evaluated, five datasets been used for this purpose.

TABLE OF CONTENTS

CHAPTER-1 (INTRODUCTION)

1. Introduction

In 1995, the support vector machine (SVM) was projected by Vapnik and Cortez. Support vector machine is associate degree rule specifically for little sample, high dimensional biometric identification downside.Face detection system is a very useful and efficient software/ program which can be very helpful in many fields. With the face detection software the criminals can be easily identified , it can also be used in the attendance system for the students. The biometrics is used in the face detection system to make it ease. A biometrics is a system or process that can be used to identify the individual characteristicssuch as expressions, way of looking , face shape etc.

In the proposed system we will use the technique of the supervised learning which is a part of Machine learning algorithm. First of all we will train the dataset with different faces of persons. The machine will learn with the help of supervised learning. Thenthe project will be tasted with the help of test dataset, if the image (face) will be available in the provided dataset or training dataset then the output will be correct and it will give all the information about that person whose face has been detected.

CHAPTER-1(INTRODUCTION)

02 . Formulation of Problem

The main problem in the making or to make use of this software is the ageing problem, expressions problems. With the increase in the age the face of a person also changes, sometimes with the different expressions of a persons the software cannot identify the face it will give the error, the person/ face identity is not identified or could not found in the given dataset.

This project is intended to identify a person using the images previously taken. The identification will be done according the previous images of different persons.

CHAPTER-1(INTRODUCTION)

Tools and Technology Used

The tools and technology that we will use for the proposed system will be asfollows:-

- We will use python   language   for the proposed   system of facedetection.

- We will use the technology of machine learning for thisproposedproject.

- For this proposed project we would require the deep learning ofmachine learning algorithms.

- Mainly here supervised learning is used.

- Biometric analysis will be used to detect the different faceexpression.

- Biometric will also cope up with the ageing problem.

The first step in developing anything is to state the requirements. This applies just as much to leading edge research as to simple programs and to personal programs, as well as to large team efforts. Being vague about your objective only postpones decisions to a later stage where changes are much more costly.

The problem statement should state what is to be done and not how it is to be done. It should be a statement of needs, not a proposal for a solution. A user manual for the desired system is a good problem statement. The requestor should indicate which features are mandatory and which are optional, to avoid overly constraining design decisions. The requestor should avoid describing system internals, as this restricts implementation flexibility. Performance specifications and protocols for interaction with external systems are legitimate requirements. Software engineering standards, such as modular construction, design for testability, and provision for future extensions, are also proper.

Many problems statements, from individuals, companies, and government agencies, mixture requirements with design decisions. There may sometimes be a compelling reason to require a particular computer or language; there is rarely justification to specify the use of a particular algorithm. The analyst must separate the true requirements from design and implementation decisions disguised as requirements. The analyst should challenge such pseudo requirements, as they restrict flexibility. There may be politics or organizational reasons for the pseurequirements, but at least the analyst should recognize that these externally imposed design decisions are not essential features of the problem domain.

A problem statement may have more or less detail. A requirement for a conventional product, such as a payroll program or a billing system, may have considerable detail. A requirement for a research effort in a new area may lack many details, but presumably the research has some objective, which should be clearly stated.

Most problem statements are ambiguous, incomplete, or even inconsistent. Some requirements are just plain wrong. Some requirements, although precisely stated, have unpleasant consequences on the system behavior or impose unreasonable implementation costs. Some requirements seem reasonable at first but do not work out as well as the request or thought. The problem statement is just a starting point for understanding the problem, not an immutable document. The purpose of the subsequent analysis is to fully understand the problem and its

implications. There is no reasons to expect that a problem statement prepared without a fully analysis will be correct.

The analyst must work with the requestor to refine the requirements so they represent the requestor's true intent. This involves challenging the requirements and probing for missing information. The psychological, organizational, and political considerations of doing this are beyond the scope of this book, except for the following piece of advice: If you do exactly what the customer asked for, but the result does not meet the customer's real needs, you will probably be blamed anyway.
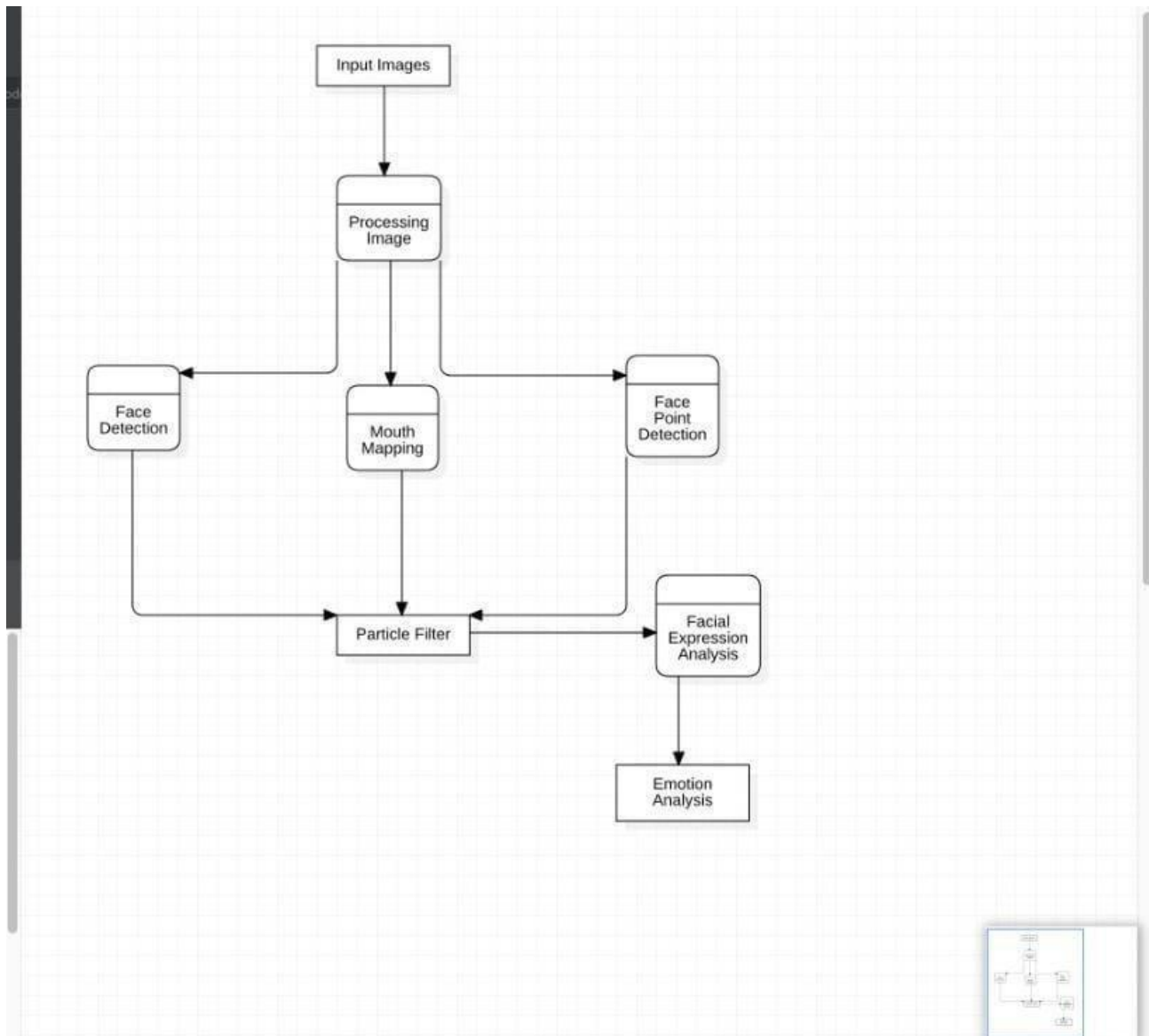
CHAPTER-2 (Literature Survey/ Project Design)

Face recognition may be a biometric approach that employs automatic ways to verify or recognise the identity of a living person supported his/ her physiological characteristics[1]. In general, a biometric

authentication system makes use of either physiological characteristics (such as a fingerprint, iris pattern, or face) or behaviour patterns (such as hand-writing, voice, or key-stroke pattern) to spot a person[4].

Because of human inherent affection of his/her eyes, some individuals square measure reluctant to use eye identification systems. Facerecognition has the advantage of being a passive, non intrusive system to verify personality in an exceedingly "natural" and friendly method. In general, biometric devices are

often explained with a three- step procedure (1) a detector takes associate degree observation. the sort of detector and its observation rely on the sort of biometric devices used[6]. Due to the various development of machine learning, the computingsurroundings, and recognition systems, many researchers have worked on pattern recognition and identification via altogether totally different statistics practice varied building mining model ways. Theplanned model used for countenance recognition that utilizes prominence maps totransfer data from associate absolute offer to a target network byfor the most part "hiding" nonrelevant info[7]. The plannedmethodology is freelance of the used model since the experience is entirely transferred via augmentation of the computer file.
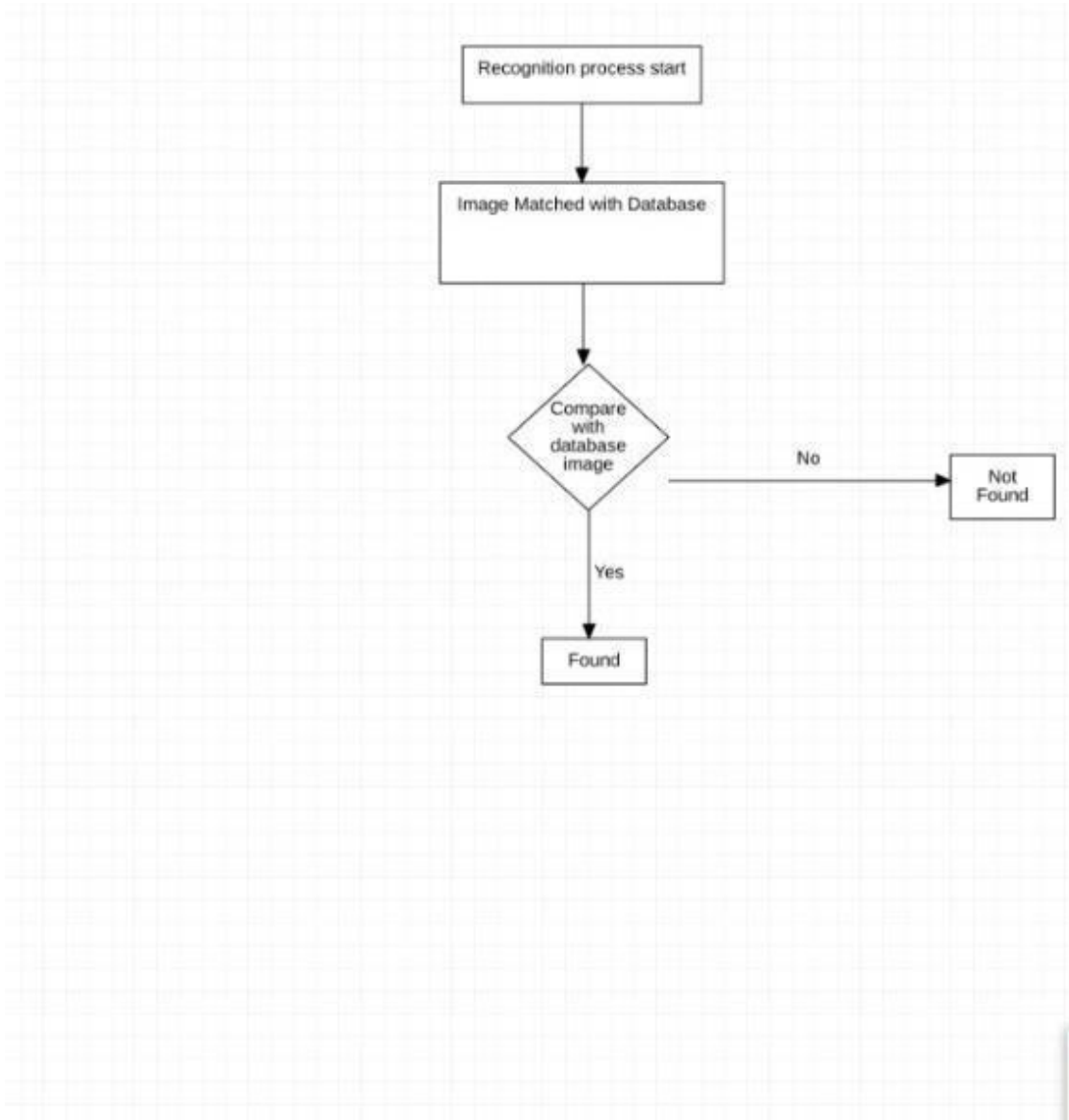
Al-Waisy, et al. Planned a multimodal profound learning system that depends on close to half presentation for k-based face acknowledgment. They consolidated the focal points of neighborhood camp-made

component descriptors with the DBN to report face acknowledgment in free circumstances. They planned a multimodal close to component extraction approach enthusiastic about consolidating the upsides of type activity with the curvelet modification, which they said because it the curvelet–fractal approach[13].
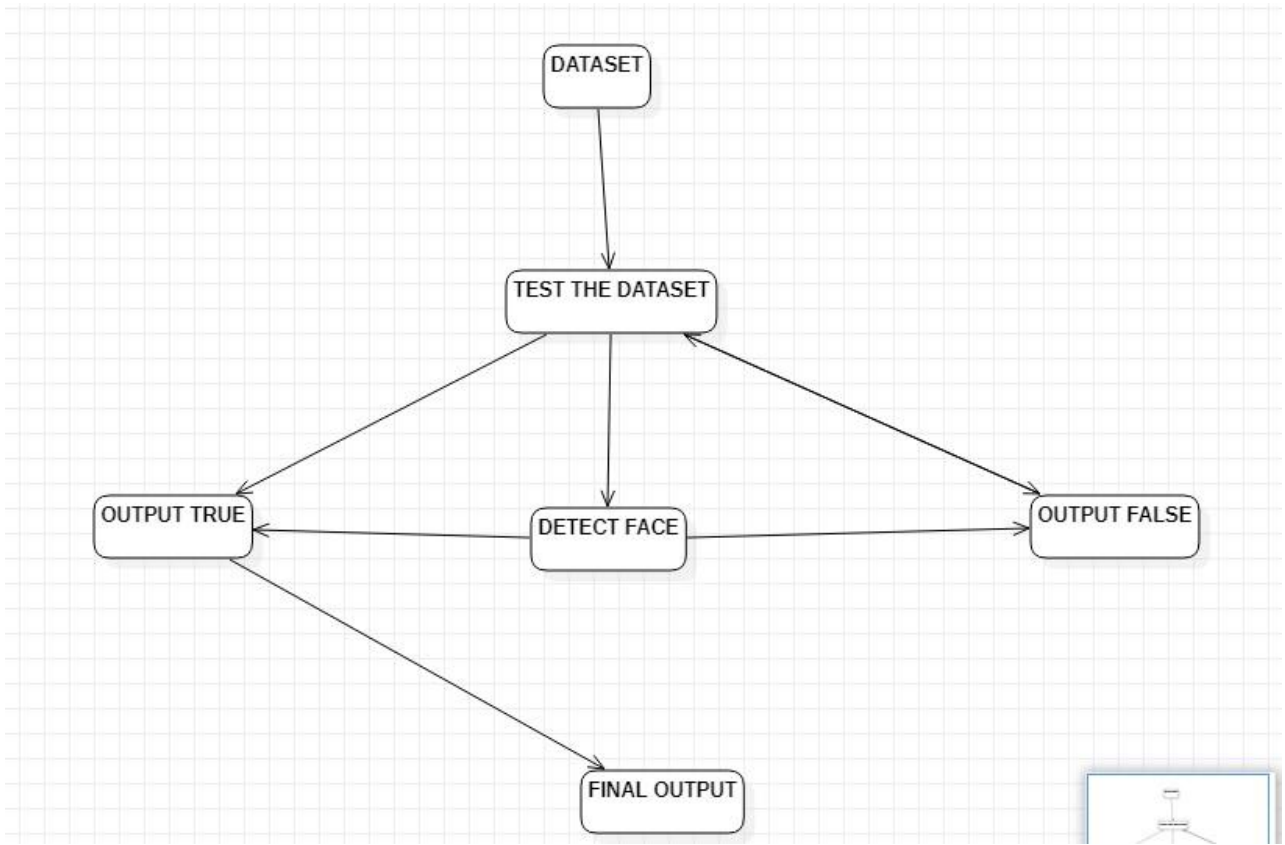
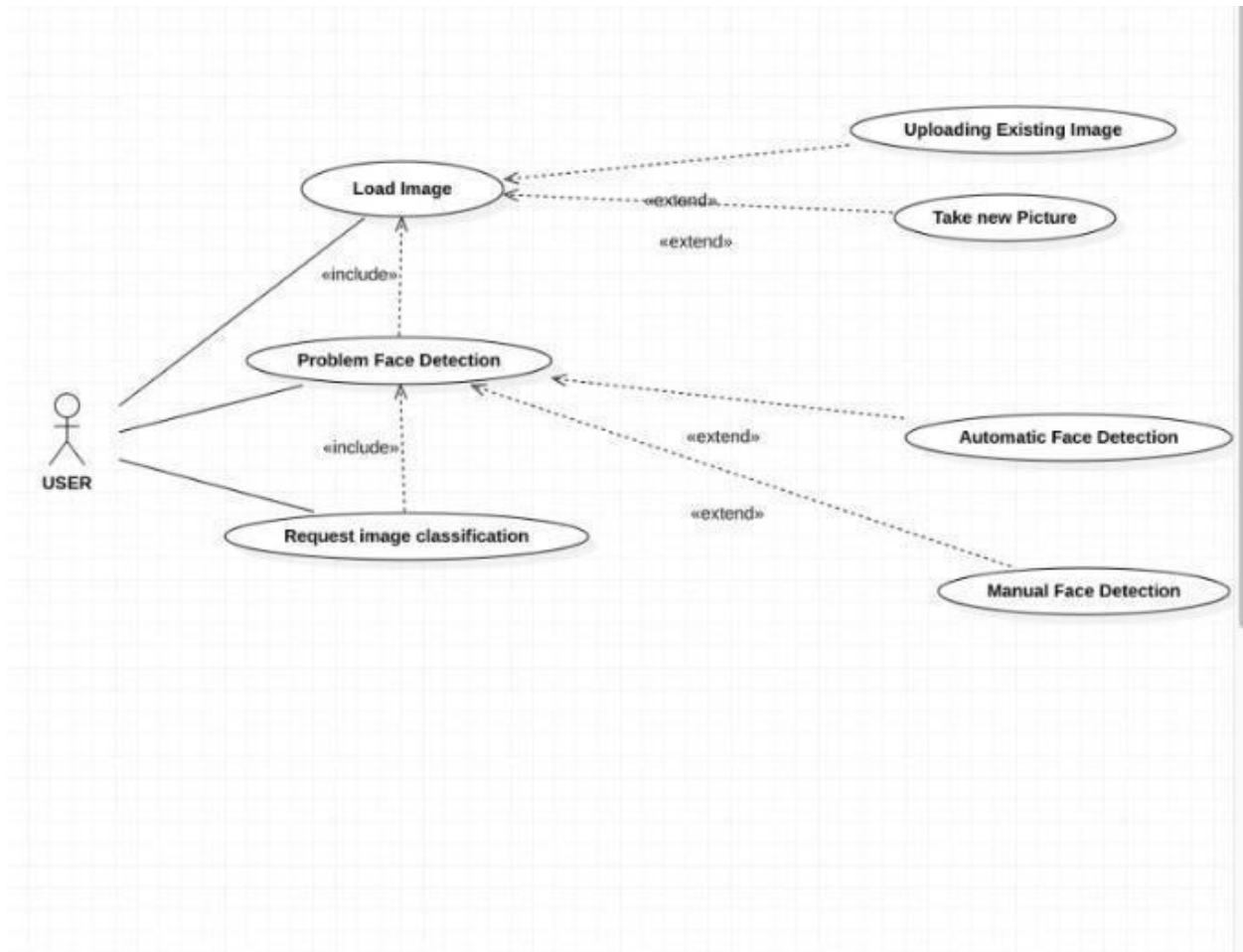# CHAPTER-3(PROJECT DESIGN):-

DATA FLOW DIAGRAM (FOR PROPOSED SYSTEM):-

FLOWCHART (FOR PROPOSED SYSTEM):-

```
        ┌─────────────────────────┐
        │ Recognition process start│
        └─────────────────────────┘
                    │
                    ▼
        ┌─────────────────────────┐
        │ Image Matched with Database│
        └─────────────────────────┘
                    │
                    ▼
               ╱╲
             ╱    ╲
           ╱ Compare ╲         No        ┌─────────┐
          ╱   with    ╲─────────────────▶│  Not    │
           ╲ database ╱                   │  Found  │
             ╲ image ╱                    └─────────┘
               ╲╱
                │
                │ Yes
                ▼
            ┌────────┐
            │ Found  │
            └────────┘
```

ACTIVITY DIAGRAM (FOR PROPOSED SYSTEM):-

USE CASE DIAGRAM (FOR PROPOSED SYSTEM):-

FEATURES OF LANGUAGE USED:-

Python is an interpreted high-level general-purpose programming language. Its design philosophy emphasizes code readability with its use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.[31]

Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library.[32]

Guido van Rossum began working on Python in the late 1980s, as a successor to the ABC programming language, and first released it in 1991 as Python 0.9.0.[33] Python 2.0 was released in 2000 and introduced new features, such as list comprehensions and a cycle- detecting garbage collection system (in addition to reference counting). Python 3.0 was released in 2008 and was a major revision of the language that is not completely backward- compatible. Python 2 was discontinued with version 2.7.18 in 2020.[34]

Python consistently ranks as one of the most popular programming languages.

Python was conceived in the late 1980s[39] by Guido van Rossum at Centrum Wiskunde & Informatica

(CWI) in the Netherlands as a successor to the ABC programming language, which was inspired by SETL,[40] capable of exception handling and interfacing with the Amoeba operating system.[11] Its implementation began in December 1989.[41] Van Rossum shouldered sole responsibility for the project, as the lead developer, until 12 July 2018, when he announced his "permanent vacation" from his responsibilities as Python's "benevolent dictator for life", a title the Python community bestowed upon him to reflect his long-term commitment as the project's chief decisionmaker.[42] In January 2019, active Python core developers elected a five-member "Steering Council" to lead the project.[43][44]

Python 2.0 was released on 16 October 2000, with many major new features, including a cycle- detecting garbage collector (in addition to reference counting) for memory management and support for Unicode.[45]

Python 3.0 was released on 3 December 2008. It was a major revision of the language that is not completely backward-compatible.[46] Many of its major features were backported to

Python 2.6.x[47] and 2.7.x version series. Releases of Python 3 include the 2to3 utility, which automates the translation of Python 2 code to Python 3.[48]

Python 2.7's end-of-life date was initially set at 2015 then postponed to 2020 out of concern that a large body of existing code could not easily be forward-ported to Python 3.[49][50] No more security patches or other improvements will be released for it.[51][52] With Python 2's end-of-life, only Python 3.6.x[53] and later are supported.

Python 3.9.2 and 3.8.8 were expedited[54] as all versions of Python (including 2.7[55]) had security issues, leading to possible remote code execution[56] and web cache poisoning.[57]

Design philosophy and features

Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming (including by metaprogramming[58] and metaobjects (magic methods)).[59] Many other paradigms are supported via extensions, including design by contract[60][61] and logic programming.[62]

Python uses dynamic typing and a combination of reference counting and a cycle-detecting garbage collector for memory management.[63] It also features dynamic name resolution (late binding), which binds method and variable names during program execution.

Python's design offers some support for functional programming in the Lisp tradition. It has filter, map and reduce functions; list comprehensions, dictionaries, sets, and generator expressions.[64] The standard library has two modules (itertools and functools) that implement functional tools borrowed from Haskell and Standard ML.[65]

The language's core philosophy is summarized in the document *The Zen of Python* (*PEP 20*), which includes aphorisms such as:[66]

- Beautiful is better than ugly.
- Explicit is better than implicit. • Simple is better than complex.
- Complex is better than complicated.
- Readability counts.

Rather than having all of its functionality built into its core, Python was designed to be highly extensible (with modules). This compact modularity has made it particularly popular as a means of adding programmable interfaces to existing applications. Van Rossum's vision of a small core language with a large standard library and easily extensible interpreter stemmed from his frustrations with ABC, which espoused the opposite approach.[39] It is often described as a "batteries included" language due to its comprehensive standard library.[67]

Python strives for a simpler, less-cluttered syntax and grammar while giving developers a choice in their coding methodology. In contrast to Perl's "there is more than one way to do it" motto, Python embraces a "there should be one— and preferably only one —obvious way to do it" design philosophy.[66] Alex Martelli, a Fellow at the Python Software Foundation and Python book author, writes that "To describe something as 'clever' is *not* considered a compliment in the Python culture."[68]

Python's developers strive to avoid premature optimization, and reject patches to non-critical parts of the CPython reference implementation that would offer marginal increases in speed at the cost of clarity.[69] When speed is important, a Python programmer can move time-critical functions to extension modules

written in languages such as C, or use PyPy, a just-in-time compiler. Cython is also available, which translates a Python script into C and makes direct C- level API calls into the Python interpreter.

Python's developers aim for the language to be fun to use. This is reflected in its name—a tribute to the British comedy group Monty Python[70]—and in occasionally playful approaches to tutorials and reference materials, such as examples that refer to spam and eggs (a reference to a Monty Python sketch) instead of the standard foo and bar.[71][72]

A common neologism in the Python community is *pythonic*, which can have a wide range of meanings related to program style. To say that code is pythonic is to say that it uses Python idioms well, that it is natural or shows fluency in the language, that it conforms with Python's minimalist philosophy and emphasis on readability. In contrast, code that is difficult to understand or reads like a rough transcription from another programming language is called *unpythonic*.[73][74]

Users and admirers of Python, especially those considered knowledgeable or experienced, are often referred to as *Pythonistas.*

Indentation

Python uses whitespace indentation, rather than curly brackets or keywords, to delimit blocks. An increase in indentation comes after certain statements; a decrease in indentation signifies the end of the current block.[78] Thus, the program's visual structure accurately represents the program's semantic structure.[79] This feature is sometimes termed the offside rule, which some other languages share, but in most languages indentation does not have any semantic meaning. The recommended indent size is four spaces.[80]

Statements and control flow

Python's statements include (among others):

- The assignment statement, using a single equals sign =.
- The if statement, which conditionally executes a block of code, along with else and elif (a contraction of else-if).
- The for statement, which iterates over an iterable object, capturing each element to a local variable for use by the attached block.

- The while statement, which executes a block of code as long as its condition is true.
- The try statement, which allows exceptions raised in its attached code block to be caught and handled by except clauses; it also ensures that clean-up code in a finally block will always be run regardless of how the block exits.
- The raise statement, used to raise a specified exception or re-raise a caught exception.
- The class statement, which executes a block of code and attaches its local namespace to a class, for use in object-oriented programming.
- The def statement, which defines a function or method.
- The with statement, which encloses a code block within a context manager (for example, acquiring a lock before the block of code is run and releasing the lock afterwards, or opening a file and then closing it), allowing resource-acquisition-is- initialization (RAII)like behavior and replaces a common try/finally idiom.[81]
- The breakstatement, exits from a loop.
- The continue statement, skips this iteration and continues with the next item.
- The del statement, removes a variable, which means the reference from the name to the value is deleted and trying to use that variable will cause an error. A deleted variable can be reassigned.
- The pass statement, which serves as a NOP. It is syntactically needed to create an empty code block.
- The assert statement, used during debugging to check for conditions that should apply.
- The yield statement, which returns a value from a generator function and yield is also an operator. This form is used to implement coroutines.
- The `return` statement, used to return a value from a function.
- The `import` statement, which is used to import modules whose functions or variables can be used in the current program.

The assignment statement (=) operates by binding a name as a reference to a separate, dynamicallyallocated object. Variables may subsequently be rebound at any time to any object. In Python, a variable name is a generic reference holder and does not have a fixed data type associated with it. However, at a given time, a variable will refer to *some* object, which will have a type. This is referred to as dynamic typing and is contrasted with statically- typed programming languages, where each variable may only contain values of a certain type.

Python does not support tail call optimization or first-class continuations, and, according to Guido van Rossum, it never will.[82][83] However, better support for coroutine-like functionality is provided, by extending Python's generators.[84] Before 2.5, generators were lazy iterators; information was passed

unidirectionally out of the generator. From Python 2.5, it is possible to pass information back into a generator function, and from Python 3.3, the information can be passed through multiple stack levels.[85]

Expressions

Some Python expressions are similar to those found in languages such as C and Java, while some are not:

- Addition, subtraction, and multiplication are the same, but the behavior of division differs. There are two types of divisions in Python. They are floor division (or integer division) // and floating-point/division.[86] Python also uses the ** operator for exponentiation.

- From Python 3.5, the new @ infix operator was introduced. It is intended to be used by libraries such as NumPy for matrix multiplication.[87][88]

- From Python 3.8, the syntax :=, called the 'walrus operator' was introduced. It assigns values to variables as part of a larger expression.[89]

- In Python, == compares by value, versus Java, which compares numerics by value[90] and objects by reference.[91] (Value comparisons in Java on objects can be performed with the equals()method.) Python's is operator may be used to compare object identities (comparison by reference). In Python, comparisons may be chained, for example a <= b <= c.

- Python   the words   uses   and, or, not for its boolean operators rather than the symbolic &&, ||, !   used in Java and C.

- Python has a type of expression termed a *list comprehension* as well as a more general expression termed a *generator expression*.[64]

- Anonymous functions are implemented using lambda expressions; however, these are limited in that the body can only be one expression.

- Conditional expressions in Python are written as x **if** c **else** y[92] (different in order of operands from the c ? x : y operator common to many other languages).

- Python makes a distinction between lists and tuples. Lists are written as [1, 2, 3], are mutable, and cannot be used as the keys of dictionaries (dictionary keys must be immutable in Python). Tuples are written as (1, 2, 3), are immutable and thus can be used as the keys of dictionaries, provided all elements of the tuple are immutable.

  The + operator can be used to concatenate two tuples, which does not directly modify their contents, but rather produces a new tuple containing the elements of both provided tuples. Thus, given the variable t initially equal to (1, 2, 3), executing t = t + (4, 5) first evaluates t + (4, 5), which yields (1, 2, 3, 4, 5), which is then assigned back to t, thereby effectively "modifying the contents" of t,

while conforming to the immutable nature of tuple objects. Parentheses are optional for tuples in unambiguous contexts.[93]

- Python features *sequence unpacking* wherein multiple expressions, each evaluating to anything that can be assigned to (a variable, a writable property, etc.), are associated in an identical manner to that forming tuple literals and, as a whole, are put on the left- hand side of the equal sign in an assignment statement. The statement expects an *iterable* object on the right-hand side of the equal sign that produces the same number of values as the provided writable expressions when iterated through and will iterate through it, assigning each of the produced values to the corresponding expression on the left.[94]

- Python has a "string format" operator %. This functions analogously to printf format strings in C, e.g. "spam=%s eggs=%d" % ("blah", 2) evaluates to "spam=blah eggs=2". In Python 3 and 2.6+, this was supplemented by the format() method of the str class, e.g. "spam={0} eggs={1}".format("blah", 2). Python 3.6 added "f-strings": blah = "blah"; eggs = 2; f'spam={blah} eggs={eggs}'.[95]

- Strings in Python can be concatenated, by "adding" them (same operator as for adding integers and floats). E.g. "spam" + "eggs" returns "spameggs". Even if your strings contain numbers, they are still added as strings rather than integers. E.g. "2" + "2" returns "22".

- Python has various kinds of string literals:

  ◦ Strings delimited by single or double quote marks. Unlike in Unix shells, Perl and Perlinfluenced languages, single quote marks and double quote marks function identically. Both kinds of string use the backslash (\) as an escape

    character. String interpolation became available in Python 3.6 as "formatted string literals".[95]

  ◦ Triple-quoted strings, which begin and end with a series of three single or double quote marks. They may span multiple lines and function like here documents in shells, Perl and Ruby.

  ◦ Raw string varieties, denoted by prefixing the string literal with an r. Escape sequences are not interpreted; hence raw strings are useful where literal backslashes are common, such as regular expressions and Windows-style paths. Compare "@-quoting" in C#.

- Python has array index and array slicing expressions on lists, denoted as a[key], a[start:stop] or a[start:stop:step]. Indexes are zero-based, and negative indexes are relative to the end. Slices take elements from the *start* index up to, but not including, the *stop* index. The third slice parameter, called *step* or *stride*, allows elements to be skipped and reversed. Slice indexes may be omitted, for example a[:] returns a copy of the entire list. Each element of a slice is a shallow copy.

In Python, a distinction between expressions and statements is rigidly enforced, in contrast to languages such as Common Lisp, Scheme, or Ruby. This leads to duplicating some functionality. For example:

- List comprehensions vs. for-loops
- Conditional expressions vs. if blocks
- The eval() vs. exec() built-in functions (in Python 2, exec is a statement); the former is for expressions, the latter is for statements.

Statements cannot be a part of an expression, so list and other comprehensions or lambda expressions, all being expressions, cannot contain statements. A particular case of this is that an assignment statement such as a = 1 cannot form part of the conditional expression of a conditional statement. This has the advantage of avoiding a classic C error of mistaking an assignment operator = for an equality operator == in conditions: **if** (c = 1) { ... } is syntactically valid (but probably unintended) C code but **if** c = 1: ... causes a syntax error in Python.

Methods

Methods on objects are functions attached to the object's class; the syntax `instance.method(argument)` is, for normal methods and functions, syntactic sugar `Class.method(instance, argument)`. Python methods have an

for `self` parameter to access instance data, in contrast to the implicit self (or this) in explicit

some other object-oriented programming languages (e.g., C++, Java, Objective-C, or Ruby). [96] Apart from this, Python also provides methods, often called *dunder methods* (due to their names beginning and ending with double-underscores), to allow user-defined classes to modify how they are handled by native operations such as length, comparison, in arithmetic operations, type conversion, and many more.[97]

Typing:-

The standard type hierarchy in Python 3

Python uses duck typing and has typed objects but untyped variable names. Type constraints are not checked at compile time; rather, operations on an object may fail, signifying that the given object is not of a suitable type. Despite being dynamically-typed, Python is strongly- typed, forbidding operations that are not well-defined (for example, adding a number to a string) rather than silently attempting to make sense of them.

Python allows programmers to define their own types using classes, which are most often used for objectoriented programming. New instances of classes are constructed by calling the class (for example, `SpamClass()` or EggsClass()), and the classes are instances of the metaclass `type` (itself an instance of itself), allowing metaprogramming and reflection.

Before version 3.0, Python had two kinds of classes: *old-style* and *new-style*.[98] The syntax of both styles is the same, the difference being whether the class object is inherited from, directly or indirectly (all newstyle classes inherit from objectand are instances of type). In

versions of Python 2 from Python 2.2 onwards, both kinds of classes can be used. Oldstyle classes were eliminated in Python 3.0.

The long-term plan is to support gradual typing[99] and from Python 3.5, the syntax of the language allows specifying static types but they are not checked in the default implementation, CPython. An experimental optional static type checker named *mypy* supports compile-time type checking.

USES OF PYTHON:-

Python can serve as a scripting language for web applications, e.g., via mod_wsgi for the Apache web server.[185] With Web Server Gateway Interface, a standard API has evolved to facilitate these applications. Web frameworks like Django, Pylons, Pyramid, TurboGears, web2py, Tornado, Flask, Bottle and Zop e support developers in the design and maintenance of complex applications. Pyjs and IronPython can be used to develop the client-side of Ajax-based applications. SQLAlchemy can be used as a data mapper to a relational database. Twisted is a framework to program communications between computers, and is used (for example) by Dropbox.

Libraries such as NumPy, SciPy and Matplotlib allow the effective use of Python in scientific computing,[186][187] with specialized libraries such as Biopython and Astropy providing domain- specific functionality. SageMath is a computer algebra system with a notebook interface programmable in Python: its library covers many aspects of mathematics, including algebra, combinatorics, numerical mathematics, number theory, and calculus.

[188] OpenCV has Python bindings with a rich set of features for computer vision and image processing.[189]

Python is commonly used in artificial intelligence projects and machine learning projects with the help of libraries like TensorFlow, Keras, Pytorch and Scikitlearn.[190][191][192][193] As a scripting language with modular architecture,

simple syntax and rich text processing tools, Python is often used for natural language processing.[194]

Python can also be used to create games, with libraries such as Pygame, which can make 2D games.

Python has been successfully embedded in many software products as a scripting language, including in finite element method software such as Abaqus, 3D parametric modeler
like FreeCAD, 3D animation packages such as 3ds Max, Blender, Cinema

4D, Lightwave, Houdini, Maya, modo, MotionBuilder, Softimage, the visual effects compositor Nuke, 2D imaging programs like GIMP,[195] Inkscape, Scribus and Paint Shop Pro,

[196] and musical notation programs like scorewriter and capella. GNU Debugger uses Python as a pretty printer to show complex structures such as C++ containers. Esri promotes Python as the best choice for writing scripts in ArcGIS.[197] It has also been used in several video games,[198][199] and has been adopted as first of the three available programming languages in Google App Engine, the other two being Java and Go.[200]

Many operating systems include Python as a standard component. It ships with most Linux distributions,[201] AmigaOS 4 (using Python 2.7), FreeBSD (as a

package), NetBSD, OpenBSD (as a package) and macOS and can be used from the command line (terminal). Many Linux distributions use installers written in Python: Ubuntu uses the Ubiquity installer, while Red Hat Linux and Fedora use the Anaconda installer. Gentoo Linux uses Python in its package management system, Portage.

Python is used extensively in the information security industry, including in exploit development.[202][203]

Most of the Sugar software for the One Laptop per Child XO, now developed at Sugar Labs, is written in Python.[204] The Raspberry Pi single-board computer project has adopted Python as its main user-programming language.

LibreOffice includes Python, and intends to replace Java with Python. Its Python Scripting Provider is a core feature[205] since Version 4.0 from 7 February 2013.

FEATURES OF PYTHON:-

1. Easy to code:

Python is a high-level programming language. Python is very easy to learn the language as compared to other languages like C, C#, Javascript, Java, etc. It is very easy to code

in python language and anybody can learn python basics in a few hours or days. It is also a developer-friendly language.

2. **Free and Open Source:**

Python language is freely available at the official website and you can download it from the given download link below click on the **Download Python** keyword.

Since it is open-source, this means that source code is also available to the public. So you can download it as, use it as well as share it.

3. **Object-Oriented Language:**

One of the key features of python is Object-Oriented programming. Python supports object-oriented language and concepts of classes, objects encapsulation, etc.

4. **GUI Programming Support:**

Graphical User interfaces can be made using a module such as PyQt5, PyQt4, wxPython, or Tk in python.
PyQt5 is the most popular option for creating graphical apps with Python.

5. **High-Level Language:**

Python is a high-level language. When we write programs in python, we do not need to remember the system architecture, nor do we need to manage the memory.

6. **Extensible feature:**

Python is a Extensible language. We can write us some Python code into C or C++ language and also we can compile that code in C/C++ language.

7. **Python is Portable language:**

Python language is also a portable language. For example, if we have python code for windows and if we want to run this code on other platforms such as Linux, Unix, and Mac then we do not need to change it, we can run this code on any platform.

8. **Python is Integrated language:**

Python is also an Integrated language because we can easily integrated python with other languages like c, c++, etc.

9. **Interpreted Language:**

Python is an Interpreted Language because Python code is executed line by line at a time. like other languages C, C++, Java, etc. there is no need to compile python code this makes it easier to debug our code. The source code of python is converted into an immediate form called bytecode.

10. **Large Standard Library**

Python has a large standard library which provides a rich set of module and functions so you do not have to write your own code for every single thing. There are many libraries present in python for such as regular expressions, unit-testing, web browsers, etc.

11. **Dynamically Typed Language:**

Python is a dynamically-typed language. That means the type (for example- int, double, long, etc.) for a variable is decided at run time not in advance because of this feature we don't need to specify the type of variable.

CHAPTER-4 (MODULES DESCRIPTION)

Well structured designs improve the maintainability of a system. A structured system is one that is developed from the top down and modular, that is, broken down into manageable components. In this project we modularized the system so that they haveminimal effect on each other.

This application is designed into five independent modules which take care of different tasks efficiently.

1. **User Interface Module.**

2. **Admin Module.**

3. **Client Module.**

4. **Database Operations Module.**

5. **Splitting and Merging Module.**

6. **Identify Module.**

User Interface Module:

Actually every application has one user interface for accessing the entire application. In this application also we are providing one user interface for accessing this application. The user interface designed completely based on the end users. It is provide friendly accessing to the users. This user interface has attractive look and feel. Technically I am using the swings in core java for preparing this user interface.

Admin Module:

User requirements	Elaboration	Further Elaboration
Create	Assign new user id & password for an employee.	

Delete	Administrator can delete theuser id & password of unwanted employee.	
Update	First the details of employees are to be obtained by using user id &password.	After obtaining the original details the updated details aresubmitted.

Client Module:

User requirements	Elaboration	Further Elaboration
Login	Employee log in to home page by entering id &password.	
Adding details	Personal details of criminal store in to data base	Images are cropped and saved in database.
Update process	Enter criminal id and obtain his details	Update the details and images of existing criminal
Delete process	Enter criminal id	Delete the details and image of unwanted criminal
Logout	Logout in to the home page	

Splitting and Merging Module:

Requirements	Elaboration	Further Elaboration
View clippings	View all clips and select the clip shown by eyewitness	Compare the clippings with images of criminals
Construction	Construct the face of criminalby clubbing all freezed clippings	

This document play a vital role in the development of life cycle (SDLC) as it describes the complete requirement of the system. It means for use by developers and will be the basic during testing phase. Any changes made to the requirements in the future will have to go through formal change approval process.

SPIRAL MODEL was defined by Barry Boehm in his 1988 article, "A spiral Model of Software Development and Enhancement. This model was not the first model to discuss iterative development, but it was the first model to explain why theiteration models.

As originally envisioned, the iterations were typically 6 months to 2 years long. Each phase starts with a design goal and ends with a client reviewing the progress thus far. Analysis and engineering efforts are applied at each phase of the project, with an eye toward the end goal of the project.

The steps for Spiral Model can be generalized as follows:

- The new system requirements are defined in as much details as possible. This usually involves interviewing a number of users representing all the external or internal users and other aspects of the existing system.

- A preliminary design is created for the new system.

- A first prototype of the new system is constructed from the preliminary design. This is usually a scaled-down system, and represents an approximation of the characteristics of the final product.

- A second prototype is evolved by a fourfold procedure:

  1. Evaluating the first prototype in terms of its strengths, weakness, andrisks.

  2. Defining the requirements of the second prototype.

  3. Planning an designing the second prototype.

  4. Constructing and testing the second prototype.

- At the customer option, the entire project can be aborted if the risk is deemed too great. Risk factors might involve development cost overruns, operatingcost miscalculation, or any other factor that could, in the customer's judgment, result in a less-than-satisfactory final product.

- The existing prototype is evaluated in the same manner as was the previous prototype, and if necessary, another prototype is developed from it according to the fourfold procedure outlined above.

- The preceding steps are iterated until the customer is satisfied that the refined prototype represents the final product desired.

- The final system is constructed, based on the refined prototype.

- The final system is thoroughly evaluated and tested. Routine maintenance is carried on a continuing basis to prevent large scale failures and to minimize down time.

SDLC METHDOLOGIES:

This document play a vital role in the development of life cycle (SDLC) as it describes the complete requirement of the system. It means for use by developers and will be the basic during testing phase.   Any changes made to the requirements in the future will have to go through formal change approval process.

SPIRAL MODEL was defined by Barry Boehm in his 1988 article, "A spiral Model of Software Development and Enhancement.   This model was not the first model to discuss iterative development, but it was the first model to explain why the iteration models.
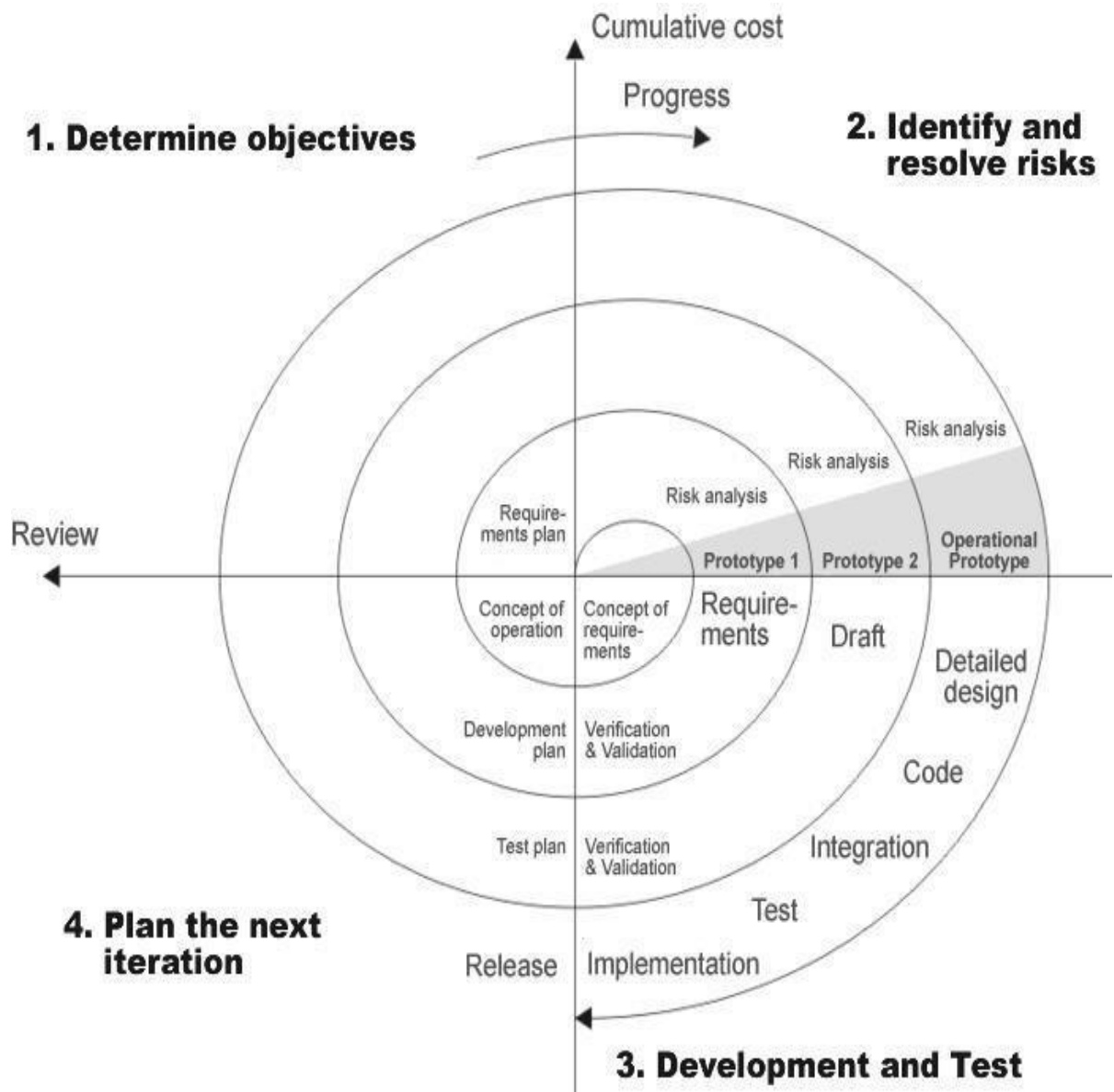
As originally envisioned, the iterations were typically 6 months to 2 years long. Each phase starts with a design goal and ends with a client reviewing the progress thus far. Analysis and engineering efforts are applied at each phase of the project, with an eye toward the end goal of the project.

The steps for Spiral Model can be generalized as follows:

- The new system requirements are defined in as much details as possible. This usually involves interviewing a number of users representing all the external or internal users and other aspects of the existing system.
- A preliminary design is created for the new system.

- A first prototype of the new system is constructed from the preliminary design. This is usually a scaled-down system, and represents an approximation of the characteristics of the final product.

- A second prototype is evolved by a fourfold procedure:

  1. Evaluating the first prototype in terms of its strengths, weakness, and risks.

  2. Defining the requirements of the second prototype.

  3. Planning an designing the second prototype.

  4. Constructing and testing the second prototype.

- At the customer option, the entire project can be aborted if the risk is deemed too great. Risk factors might involve development cost overruns, operating-cost

miscalculation, or any other factor that could, in the customer's judgment, result in a less-than-satisfactory final product.

- The existing prototype is evaluated in the same manner as was the previous prototype, and if necessary, another prototype is developed from it according to the fourfold procedure outlined above.

- The preceding steps are iterated until the customer is satisfied that the refined prototype represents the final product desired.

- The final system is constructed, based on the refined prototype.

- The final system is thoroughly evaluated and tested. Routine maintenance is carried on a continuing basis to prevent large scale failures and to minimize down time.

ADVANTAGES:

- Estimates(i.e. budget, schedule etc .) become more relistic as work progresses, because important issues discoved earlier .

- It is more able to cope with the changes that are software development generally entails.

- Software engineers can get their hands in and start woring on the core of a project earlier.

APPLICATION DEVELOPMENT
N-TIER APPLICATIONS

N-Tier Applications can easily implement the concepts of Distributed Application Design and Architecture. The N-Tier Applications provide strategic benefits to Enterprise Solutions. While 2-tier, client-server can help us create quick and easy solutions and may be used for Rapid Prototyping, they can easily become a maintenance and security night mare

The N-tier Applications provide specific advantages that are vital to the business continuity of the enterprise. Typical features of a real life n-tier may include the following:

- Security

- Availability and Scalability

- Manageability

- Easy Maintenance

- Data Abstraction

The above mentioned points are some of the key design goals of a successful n-tier application that intends to provide a good Business Solution.

DEFINITION

Simply stated, an n-tier application helps us distribute the overall functionality into various

tiers or layers:

- Presentation Layer

- Business Rules Layer

- Data Access Layer

• Database/Data Store

Each layer can be developed independently of the other provided that it adheres to the standards and communicates with the other layers as per the specifications.

This is the one of the biggest advantages of the n-tier application. Each layer can potentially treat the other layer as a 'Block-Box'.

In other words, each layer does not care how other layer processes the data as long as it sends the right data in a correct format.
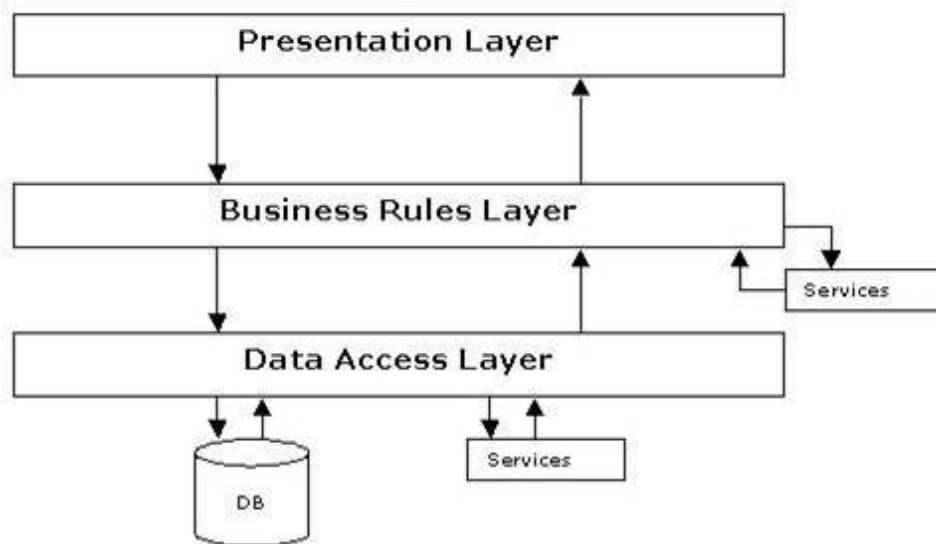

Fig 1.1-N-Tier Architecture

1 THE PRESENTATION LAYER

Also called as the client layer comprises of components that are dedicated to presenting the data to the user. For example: Windows/Web Forms and buttons, edit boxes, Text boxes, labels, grids, etc.

2 THE BUSINESS RULES LAYER

This layer encapsulates the Business rules or the business logic of the encapsulations. To have a separate layer for business logic is of a great advantage. This is because any changes in Business Rules can be easily handled in this layer. As long as the interface between the layers remains the same, any changes to the functionality/processing logic in this layer can be made without impacting the others. A lot of client-server apps failed to implement successfully as changing the business logic was a painful process. **3 THE DATA ACCESS LAYER**

This layer comprises of components that help in accessing the Database. If used in the right way, this layer provides a level of abstraction for the database structures. Simply put changes made to the database, tables, etc do not affect the rest of the application because of the Data Access layer. The different application layers send the data requests to this layer and receive the response from this layer.

4 THE DATABASE LAYER

This layer comprises of the Database Components such as DB Files, Tables, Views, etc. The Actual database could be created using SQL Server, Oracle, Flat files, etc. In an n-tier application, the entire application can be implemented in such a way that it is independent of the actual Database. For instance, you could change the Database Location with minimal changes to Data Access Layer. The rest of the Application should remain unaffected.

MACHINE LEARNING:-

Machine learning (**ML**) is the study of computer algorithms that can improve automatically through experience and by the use of data.[1] It is seen as a part of artificial intelligence.

Machine learning algorithms build a model based on sample data, known as training data, in order to make predictions or decisions without being explicitly programmed to do so.[2] Machine

learning algorithms are used in a wide variety of applications, such as in medicine, email filtering, speech recognition, and computer vision, where it is difficult or unfeasible to develop conventional algorithms to perform the needed tasks.[3]

A subset of machine learning is closely related to computational statistics, which focuses on making predictions using computers; but not all machine learning is statistical learning. The study of mathematical optimization delivers methods, theory and application domains to the field of machine learning. Data mining is a related field of study, focusing on exploratory data analysis through unsupervised learning.[5][6] Some implementations of machine learning use data and neural networks in a way that mimics the working of a biological brain.[7][8] In its application across business problems, machine learning is also referred to as predictive analytics.

Learning algorithms work on the basis that strategies, algorithms, and inferences that worked well in the past are likely to continue working well in the future. These inferences can be obvious, such as "since the sun rose every morning for the last 10,000 days, it will probably rise tomorrow morning as well". They can be nuanced, such as "X% of families have geographically separate species with color variants, so there is a Y% chance that undiscovered black swans exist".[9]

Machine learning programs can perform tasks without being explicitly programmed to do so. It involves computers learning from data provided so that they carry out certain tasks. For simple tasks assigned to computers, it is possible to program algorithms telling the machine how to execute all steps required to solve the problem at hand; on the computer's part, no learning is needed. For more advanced tasks, it can be challenging for a human to manually create the needed algorithms. In practice, it can turn out to be more effective to help the machine develop its own algorithm, rather than having human programmers specify every needed step.[10]

The discipline of machine learning employs various approaches to teach computers to accomplish tasks where no fully satisfactory algorithm is available. In cases where vast numbers of potential answers exist, one approach is to label some of the correct answers as valid. This can then be used as training data for the computer to improve the algorithm(s) it uses to determine correct answers. For example, to train a system for the task of digital character recognition, the MNIST dataset of handwritten digits has often been used.[10]

Association rules

Association rule learning is a rule-based machine learning method for discovering relationships between variables in large databases. It is intended to identify strong rules discovered in databases using some measure of "interestingness".[60]

Rule-based machine learning is a general term for any machine learning method that identifies, learns, or evolves "rules" to store, manipulate or apply knowledge. The defining characteristic of a rule-based machine learning algorithm is the identification and utilization of a set of relational rules that collectively represent the knowledge captured by the system. This is in contrast to other machine learning algorithms that commonly identify a singular model that can be universally applied to any instance in order to make a prediction.[61] Rule-based machine learning approaches include learning classifier systems, association rule learning, and artificial immune systems.

Based on the concept of strong rules, Rakesh Agrawal, Tomasz Imieliński and Arun Swami introduced association rules for discovering regularities between products in large-scale transaction data recorded by point-of-sale (POS) systems in supermarkets.[62] For example, the rule found in the sales data of a supermarket would indicate that if a customer buys onions and potatoes together, they are likely to also buy hamburger meat.

Such information can be used as the basis for decisions about marketing activities such as promotional pricing or product placements. In addition to market basket analysis, association rules are employed today in application areas including Web usage mining, intrusion detection, continuous production, and bioinformatics. In contrast with sequence mining, association rule learning typically does not consider the order of items either within a transaction or across transactions.

Learning classifier systems (LCS) are a family of rule-based machine learning algorithms that combine a discovery component, typically a genetic algorithm, with a learning component, performing either supervised learning, reinforcement learning, or unsupervised learning. They seek to identify a set of context-dependent rules that collectively store and apply knowledge in a piecewise manner in order to make predictions.[63]

Inductive logic programming (ILP) is an approach to rule-learning using logic programming as a uniform representation for input examples, background knowledge, and hypotheses. Given an encoding of the known background knowledge and a set of examples represented as a logical database of facts, an ILP system will derive a hypothesized logic program that entails all positive and no negative examples. Inductive programming is a related field that considers any kind of programming language for representing hypotheses (and not only logic programming), such as functional programs.

Inductive logic programming is particularly useful in bioinformatics and natural language processing. Gordon Plotkin and Ehud Shapiro laid the initial theoretical foundation for inductive machine learning in a logical setting.[64][65][66] Shapiro built their first implementation (Model Inference System) in 1981: a Prolog program that inductively inferred logic programs from positive and negative examples.[67] The term *inductive* here refers to philosophical induction, suggesting a theory to explain observed facts, rather than mathematical induction, proving a property for all members of a well-ordered set.

Models

Performing machine learning involves creating a model, which is trained on some training data and then can process additional data to make predictions. Various types of models have been used and researched for machine learning systems.

Artificial neural networks

An artificial neural network is an interconnected group of nodes, akin to the vast network of neurons in a brain. Here, each circular node represents an artificial neuron and an arrow represents a connection from the output of one artificial neuron to the input of

Artificial neural networks (ANNs), or connectionist systems, are computing systems vaguely inspired by the biological neural networks that constitute animal brains. Such systems "learn" to

perform tasks by considering examples, generally without being programmed with any task- specific rules.

An ANN is a model based on a collection of connected units or nodes called "artificial neurons", which loosely model the neurons in a biological brain. Each connection, like the synapses in a biological brain, can transmit information, a "signal", from one artificial neuron to another. An artificial neuron that receives a signal can process it and then signal additional artificial neurons connected to it. In common ANN implementations, the signal at a connection between artificial neurons is a real number, and the output of each artificial neuron is computed by some non- linear function of the sum of its inputs. The connections between artificial neurons are called "edges". Artificial neurons and edges typically have a weight that adjusts as learning proceeds. The weight increases or decreases the strength of the signal at a connection. Artificial neurons may have a threshold such that the signal is only sent if the aggregate signal crosses that threshold. Typically, artificial neurons are aggregated into layers. Different layers may perform different kinds of transformations on their inputs. Signals travel from the first layer (the input layer) to the last layer (the output layer), possibly after traversing the layers multiple times.

The original goal of the ANN approach was to solve problems in the same way that a human brain would. However, over time, attention moved to performing specific tasks, leading to deviations from biology. Artificial neural networks have been used on a variety of tasks, including computer vision, speech recognition, machine translation, social network filtering, playing board and video games and medical diagnosis.

Deep learning consists of multiple hidden layers in an artificial neural network. This approach tries to model the way the human brain processes light and sound into vision and hearing.
Some successful applications of deep learning are computer vision and speech recognition.[68]

Decision trees[

Decision tree learning uses a decision tree as a predictive model to go from observations about an item (represented in the branches) to conclusions about the item's target value (represented in the leaves). It is one of the predictive modeling approaches used in statistics, data mining, and machine learning. Tree models where the target variable can take a discrete set of values are called classification trees; in these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels. Decision trees where the target variable can take continuous values (typically real numbers) are called regression trees. In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making. In data mining, a decision tree describes data, but the resulting classification tree can be an input for decision making

Support-vector machines

Support-vector machines (SVMs), also known as support-vector networks, are a set of related supervised learning methods used for classification and regression. Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that predicts whether a new example falls into one category or the other.[69] An SVM training algorithm is a non-probabilistic, binary, linear classifier, although methods such as Platt scaling exist to use SVM in a probabilistic classification setting. In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high- dimensional feature spaces.

Illustration of linear regression on a data set.

Regression analysis

Regression analysis encompasses a large variety of statistical methods to estimate the relationship between input variables and their associated features. Its most common form is linear regression, where a single line is drawn to best fit the given data according to a mathematical criterion such as ordinary least squares. The latter is often extended by regularization (mathematics) methods to mitigate overfitting and bias, as in ridge regression. When dealing with non-linear problems, go-to models include polynomial regression (for example, used for trendline fitting in Microsoft Excel[70]), logistic regression (often used in statistical classification) or even kernel regression, which introduces non-linearity by taking advantage of the kernel trick to implicitly map input variables to higherdimensional space.

Bayesian networks

A simple Bayesian network. Rain influences whether the sprinkler is activated, and both rain and the sprinkler influence whether the grass is wet.

A Bayesian network, belief network, or directed acyclic graphical model is a probabilistic graphical model that represents a set of random variables and their

conditional independence with a directed acyclic graph (DAG). For example, a Bayesian network could represent the probabilistic relationships between diseases and symptoms. Given symptoms, the network can be used to compute the probabilities of the presence of various diseases.

Efficient algorithms exist that perform inference and learning. Bayesian networks that model sequences of variables, like speech signals or protein sequences, are called dynamic Bayesian networks. Generalizations of Bayesian networks that can represent and solve decision problems under uncertainty are called influence diagrams.

Genetic algorithms

A genetic algorithm (GA) is a search algorithm and heuristic technique that mimics the process of natural selection, using methods such as mutation and crossover to generate new genotypes in the hope of finding good solutions to a given problem. In machine learning, genetic algorithms were used in the 1980s and 1990s.[71][72] Conversely, machine learning techniques have been used to improve the performance of genetic and evolutionary algorithms.

Training models

Usually, machine learning models require a lot of data in order for them to perform well. Usually, when training a machine learning model, one needs to collect a large, representative sample of data from a training set. Data from the training set can be as varied as a corpus of text, a collection of images, and data collected from individual users of a service. Overfitting is something to watch out for when training a machine learning model. Trained models derived from biased data can result in skewed or undesired predictions. Algorithmic bias is a potential result from data not fully prepared for training.

Artificial intelligence

Part of machine learning as subfield of AI or part of AI as subfield of machine learning[22]

As a scientific endeavor, machine learning grew out of the quest for artificial intelligence. In the early days of AI as an academic discipline, some researchers were interested in having machines learn from data. They attempted to approach the problem with various symbolic methods, as well as what was then termed "neural networks"; these were mostly perceptrons and other models that were later found to be reinventions of the generalized linear models of statistics.[23] Probabilistic reasoning was also employed, especially in automated medical diagnosis.[24]: 488

However, an increasing emphasis on the logical, knowledge-based approach caused a rift between AI and machine learning. Probabilistic systems were plagued by theoretical and practical problems of data acquisition and representation.[24]: 488 By 1980, expert systems had come to dominate AI, and statistics was out of favor.[25] Work on symbolic/knowledge-based learning did continue within AI, leading to inductive logic programming, but the more statistical line of research was now outside the field of AI proper, in pattern recognition and information retrieval.[24]: 708–710, 755 Neural networks research had been abandoned by AI and computer science around the same time. This line, too, was continued outside the AI/CS field, as "connectionism", by researchers from other disciplines

including Hopfield, Rumelhart and Hinton. Their main success came in the mid-1980s with the reinvention of backpropagation.[24]: 25

Machine learning (ML), reorganized as a separate field, started to flourish in the 1990s. The field changed its goal from achieving artificial intelligence to tackling solvable problems of a practical nature. It shifted focus away from the symbolic approaches it had inherited from AI, and toward methods and models borrowed from statistics and probability theory.[25]

The difference between ML and AI is frequently misunderstood. ML learns and predicts based on passive observations, whereas AI implies an agent interacting with the environment to learn and take actions that maximize its chance of successfully achieving its goals.[26]

As of 2020, many sources continue to assert that ML remains a subfield of AI. Others have the view that not all ML is part of AI, but only an 'intelligent subset' of ML should be considered AI.

USE OF SUPERVISED MACHINE LEARNING ALGORITHMS:-

Supervised learning is the types of machine learning in which machines are trained using well "labelled" training data, and on basis of that data, machines predict the output. The labelled data means some input data is already tagged with the correct output.

In supervised learning, the training data provided to the machines work as the supervisor that teaches the machines to predict the output correctly. It applies the same concept as a student learns in the supervision of the teacher.

Supervised learning is a process of providing input data as well as correct output data to the machine learning model. The aim of a supervised learning algorithm is to **find a mapping function to map the input variable(x) with the output variable(y)**.

In the real-world, supervised learning can be used for **Risk Assessment, Image classification, Fraud Detection, spam filtering**, etc.

Supervised learning algorithms build a mathematical model of a set of data that contains both the inputs and the desired outputs.[34] The data is known as training data, and consists of a set of training examples. Each training example has one or more inputs and the desired output, also known as a supervisory signal. In the mathematical model, each training example is represented by an array or vector, sometimes called a feature vector, and the training data is represented by a matrix. Through iterative optimization of an objective function, supervised learning algorithms learn a function that can be used to predict the output associated with new inputs.[35] An optimal function will allow the algorithm to correctly determine the output for inputs that were not a part of the training data. An algorithm that improves the accuracy of its outputs or predictions over time is said to have learned to perform that task.[18]

Types of supervised learning algorithms include active learning, classification and regression.

[26] Classification algorithms are used when the outputs are restricted to a limited set of values, and regression algorithms are used when the outputs may have any numerical value within a range. As an example, for a classification algorithm that filters emails, the input would be an incoming email, and the output would be the name of the folder in which to file the email.

Similarity learning is an area of supervised machine learning closely related to regression and classification, but the goal is to learn from examples using a similarity function that measures how similar or related two objects are. It has applications in ranking, recommendation systems, visual identity tracking, face verification, and speaker verification.

How Supervised Learning Works?

In supervised learning, models are trained using labelled dataset, where the model learns about each type of data. Once the training process is completed, the model is tested on the basis of test data (a subset of the training set), and then it predicts the output.
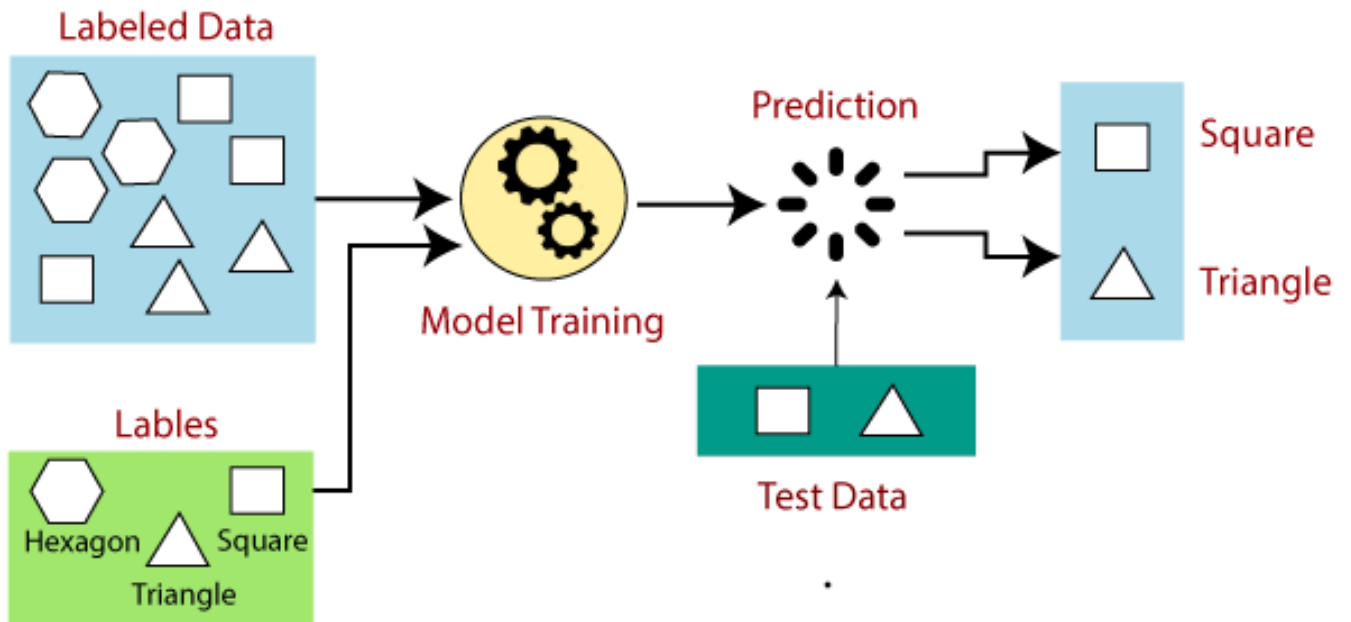
Steps Involved in Supervised Learning:

- ◦ First Determine the type of training dataset ◦
  Collect/Gather the labelled training data.
- ◦ Split the training dataset into training **dataset, test dataset, and validation dataset**.
- ◦ Determine the input features of the training dataset, which should have enough knowledge so that the model can accurately predict the output.
- ◦ Determine the suitable algorithm for the model, such as support vector machine, decision tree, etc.
- ◦ Execute the algorithm on the training dataset. Sometimes we need validation sets as the control parameters, which are the subset of training datasets.
- ◦ Evaluate the accuracy of the model by providing the test set. If the model predicts the correct output, which means our model is accurate.

Types of supervised Machine learning Algorithms:

Supervised learning can be further divided into two types of problems:
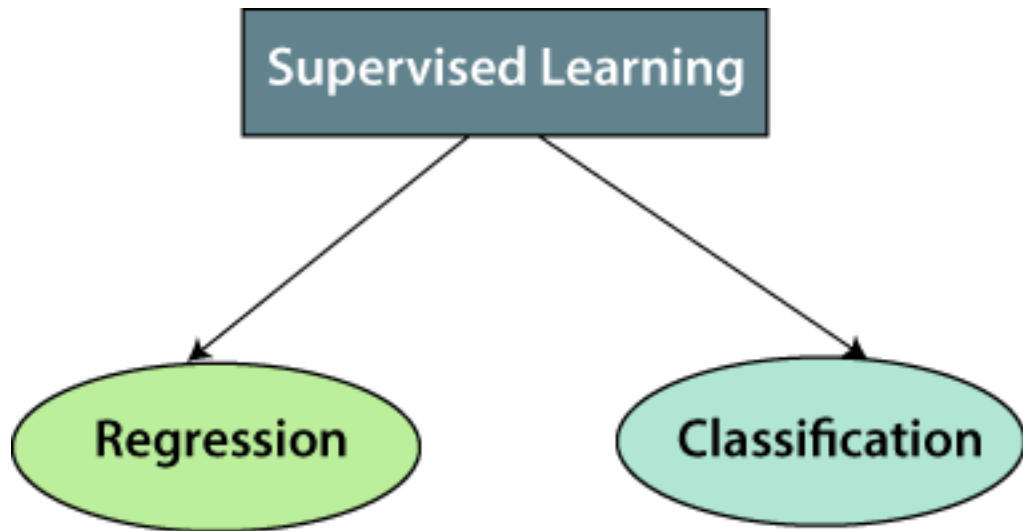
1. Regression

Regression algorithms are used if there is a relationship between the input variable and the output variable. It is used for the prediction of continuous variables, such as Weather forecasting, Market Trends, etc. Below are some popular Regression algorithms which come under supervised learning:

Linear Regression

- Regression Trees
- Non-Linear Regression
- Bayesian Linear Regression
- Polynomial Regression

2.Classification

Classification algorithms are used when the output variable is categorical, which means there are two classes such as Yes-No, Male-Female, True-false, etc.

Spam Filtering,

- Random Forest
- Decision Trees
- Logistic Regression
- Support vector Machines

Advantages of Supervised learning:

◦ With the help of supervised learning, the model can predict the output on the basis of prior experiences.
◦ In supervised learning, we can have an exact idea about the classes of objects.
◦ Supervised learning model helps us to solve various real-world problems such as **fraud detection, spam filtering**, etc.

Disadvantages of supervised learning:

◦ Supervised learning models are not suitable for handling the complex tasks.
◦ Supervised learning cannot predict the correct output if the test data is different from the training dataset.
◦ Training required lots of computation times.
◦ In supervised learning, we need enough knowledge about the classes of object.

CHAPTER-5 (RESULT)

As a result, some of actual skin color was excluded and conversely some of non-skin color was included. More precise skin color detection is expected if the window shape is closer to the actualdistribution, such as triangle. Despite this imperfect windowing, the overall results of skin color detection were very encouraging. In some cases, unnecessary noises were added in the process of edge integration in image segmentation. Though the Roberts cross operator minimized the problem associated with the following small-hole removal process, some black region connected to peripheral face edges were magnified by series of erosions and even divided into separate faceregions that were recognized as several small faces. The Sobel cross filter or the Prewitt filter withpre-rejection of small clutters were proved to work more effectively in some training images, but the threshold from which the area is regarded as a 'small clutter' varies even in a single picture and it discouraged the filter applied in our algorithm. However, the threshold decided in a more interactive way with the original image is able to discriminate face edges from other edge lines effectively and lots of applications have been presented. Based on the color segmentation, skin-colored areas were taken apart into small and squared test images. In some cases, several squared test images were generated within a face due to excessive erosion in the color segmentation process. Before performing eigenimage matching with the test images, these unnecessary squares should be merged into one so that only one test image can be taken Page 17 per face. In order to judge if it corresponds to a face or not, each test image needs to be matched with eigenimage which is simply called from the data base which were already built in.

CHAPTER-6 (CONCLUSION)

The computational models, which were implemented in this project, were chosen after extensive research, and the successful testing results confirm that the choices made by the researcher were reliable.The system with manual face detection and automatic face recognition did not have a recognition accuracy over 90%, due to the limited number of eigenfaces that were used for the PCAtransform. This system was tested under very robust conditions in this experimental study and it is envisaged that real-world performance will be far more accurate.The fully automated frontal view face detection

system displayed virtually perfect accuracy and in the researcher's opinion further work need not be conducted in this area. The fully automated face detection and recognition systemwas not robust enough to achieve a high recognition accuracy. The only reason for this was the facerecognition subsystem did not display even a slight degree of invariance to scale, rotation or shift errors of the segmented face image. This was one of the system requirements identified in section

However, if some sort of further processing, such as an eye detection technique, was implemented to further normalise the segmented face image, performance will increase to levels comparable to the manual face detection and recognition system. Implementing an eye detection technique would be a minor extension to the implemented system and would not require a great dealof additional research.All other implemented systems displayed commendable results and reflect well on the deformable template and Principal Component Analysis strategies.The most suitable real-world applications for face detection and recognition systems are for mugshot matching and surveillance. There are better techniques such as iris or retina recognition and face recognition usingthe thermal spectrum for user access and user verification applications since these need a very high degree of accuracy.The real-time automated pose invariant face detection and recognition system proposed in chapter seven would be ideal for crowd surveillance applications. If such a system werewidely implemented its potential for locating and tracking suspects for law enforcement agencies is immense. The implemented fully automated face detection and recognition system (with an eye detection system) could be used for simple surveillance applications such as ATM user security, while the implemented manual face detection and automated recognition system is ideal of mugshotmatching. Since controlled conditions are present when mugshots are gathered, the frontal view face recognition scheme should display a recognition accuracy far better than the results, which were obtained in this study, which was conducted under adverse conditions. Department of ECE Page 48 Furthermore, many of the test subjects did not present an expressionless, frontal view to thesystem. They would probably be more compliant when a 6'5" policeman is taking their mugshot! In mugshot matching applications, perfect recognition accuracy or an exact match is not a requirement. If a

face recognition system can reduce the number of images that a human operator has to search through for a match from 10000 to even a 100, it would be of incredible practical use in law enforcement. The automated vision systems implemented in this thesis did not even approachthe performance, nor were they as robust as a human's innate face recognition system. However, they give an insight into what the future may hold in computer vision

CHAPTER-7 (REFERENCES)

1. Francis Galton, "Personal identification and description," In Nature,pp. 173-177, June 21, 1888.

2. W. Zaho, "Robust image based 3D face recognition," Ph.D. Thesis, Maryland University, 1999.

3. R. Chellappa, C.L. Wilson and C. Sirohey, "Humain and machine recognition of faces: A survey," Proc. IEEE, vol. 83, no. 5, pp. 705-740, may1995.

4. T. Fromherz, P. Stucki, M. Bichsel, "A survey of face recognition," MML Technical Report, No 97.01, Dept. of Computer Science, University of Zurich, Zurich, 1997.

5. T. Riklin-Raviv and A. Shashua, "The Quotient image: Class based recognition and synthesis under varying illumination conditions," In CVPR,P. II: pp. 566-571,1999.

6. G.j. Edwards, T.f. Cootes and C.J. Taylor, "Face recognition using active appearance models," In ECCV, 1998.

7. T. Sim, R. Sukthankar, M. Mullin and S. Baluja, "Memory-based face recognition for vistor identification," In AFGR, 2000.

8. T. Sim and T. Kanade, "Combing models and exemplars for face recognition: An illuminating example," In Proceeding Of Workshop on Models Versus Exemplars in Computer Vision, CUPR 2001.

9. L. Sirovitch and M. Kirby, "Low-dimensional procedure for the characterization of human faces," Journal of the Optical Society of America A, vol. 2, pp. 519-524, 1987.

10. M. Turk and A. Pentland "Face recognition using eigenfaces," In Proc. IEEE Conference on Computer Vision and Pattern Recognition, pp.586- 591, 1991.

11. P. Belhumeur, P. Hespanha, and D. Kriegman, "Eigenfaces vs fisherfaces: Recognition using class specific linear projection," IEEE Transactions on Pattern Analysis and MachineIntelligence, vol. 19, no. 7, pp. 711-720, 1997.

12. M. Fleming and G. Cottrell, "Categorization of faces using unsupervised feature extraction," In Proc. IEEE IJCNN International Joint Conference on Neural Networks, pp. 65-70, 1990.

13. B. Moghaddam, W. Wahid, and A. Pentland, "Beyond eigenfaces: Probabilistic matching for face recognition," In Proc. IEEE International Conference on Automatic Face and Gesture

    Recognition, pp. 30-35,1998. Malay K. Kundu; Sushmita Mitra; Debasis
14. Mazumdar; Sankar K. Pal, eds. (2012). Perception and Machine Intelligence: First Indo-

Japan Conference, PerMIn 2012, Kolkata, India, January 12–13, 2011, Proceedings. Springer Science & Business Media. p. 29. ISBN 9783642273865.

15. Wechsler, Harry (2009). Malay K. Kundu; Sushmita Mitra (eds.). Reliable Face Recognition Methods: System Design, Implementation and Evaluation. Springer Science & Business Media. pp. 11–12. ISBN 9780387384641

16. Jun Wang; Laiwan Chan; DeLiang Wang, eds. (2012). Neural Information Processing: 13th International Conference, ICONIP 2006, Hong Kong, China, October 3-6, 2006, Proceedings, Part II. Springer Science & Business Media. p. 198. ISBN 9783540464822.

17. Wechsler, Harry (2009). Reliable Face Recognition Methods: System Design, Implementation and Evaluation. Springer Science & Business Media. p. 12. ISBN 9780387384641.

18. Wechsler, Harry (2009). Malay K. Kundu; Sushmita Mitra (eds.). Reliable Face Recognition Methods: System Design, Implementation and Evaluation. Springer Science & Business Media. p. 12. ISBN 9780387384641.

19. Malay K. Kundu; Sushmita Mitra; Debasis Mazumdar; Sankar K. Pal, eds. (2012). Perception and Machine Intelligence: First Indo-Japan Conference, PerMIn 2012, Kolkata, India, January 12–13, 2011, Proceedings. Springer Science & Business Media. p. 29. ISBN 9783642273865.

20. "Mugspot Can Find A Face In The Crowd – Face-Recognition Software Prepares To Go To Work In The Streets". *ScienceDaily*. November 12, 1997. Retrieved November 6, 2007.

21. Malay K. Kundu; Sushmita Mitra; Debasis Mazumdar; Sankar K. Pal, eds. (2012). Perception and Machine Intelligence: First Indo-Japan Conference, PerMIn 2012, Kolkata, India, January 12–13, 2011, Proceedings. Springer Science & Business Media. p. 29. ISBN 9783642273865.

22. Li, Stan Z.; Jain, Anil K. (2005). *Handbook of Face Recognition*. Springer Science & Business Media. pp. 14–15. ISBN 9780387405957.

23. Kumar Datta, Asit; Datta, Madhura; Kumar Banerjee, Pradipta (2015). *Face Detection and Recognition: Theory and Practice*. CRC. p. 123. ISBN 9781482226577.

24. Li, Stan Z.; Jain, Anil K. (2005). *Handbook of Face Recognition*. Springer Science & Business Media. p. 1. ISBN 9780387405957.

25. Li, Stan Z.; Jain, Anil K. (2005). *Handbook of Face Recognition*. Springer Science & Business Media. p. 2. ISBN 9780387405957.

26. "Airport Facial Recognition Passenger Flow Management". *hrsid.com*.

27. Jump up to:

a b c Bonsor, K. (September 4, 2001). "How Facial Recognition Systems Work". Retrieved June 2, 2008.

28. Smith, Kelly. "Face Recognition" (PDF). Retrieved June 4, 2008.

29. R. Brunelli and T. Poggio, "Face Recognition: Features versus Templates", IEEE Trans. on PAMI, 1993, (15)10:1042–1052

30. R. Brunelli, Template Matching Techniques in Computer Vision: Theory and Practice, Wiley, ISBN 978-0-470-51706-2, 2009 ([1] TM book)

31. Zhang, David; Jain, Anil (2006). Advances in Biometrics: International Conference, ICB 2006, Hong Kong, China, January 5–7, 2006, Proceedings. Berlin: Springer Science & Business Media. p. 183. ISBN 9783540311119.

32. "A Study on the Design and Implementation of Facial Recognition Application System". *International Journal of Bio-Science and Bio-Technology*.

33. Harry Wechsler (2009). Reliable Face Recognition Methods: System Design, Implementation and Evaluation. Springer Science & Business Media. p. 196. ISBN 9780387384641.

34. Jump up to:
     a b c d Williams, Mark. "Better Face-Recognition Software". Retrieved June 2, 2008.

35. Crawford, Mark. "Facial recognition progress report". SPIE Newsroom. Retrieved October 6, 2011.