**CLASSIFICATION OF CANCER**

**Abstract**

There are a variety of options available for cancer treatment. The type of treatment recommended for an individual is influenced by various factors such as cancer-type, the severity of a cancer (stage) and most important the genetic heterogeneity. In such a complex environment, the targeted drug treatments are likely to be irresponsive or respond differently. To study anti-cancer drug response we need to understand cancerous profiles. These cancerous profiles carry information which can reveal the underlying factors responsible for cancer growth. Hence, there is need to analyze cancer data for predicting optimal treatment options. Analysis of such profiles can help to predict and discover potential drug targets and drugs. In this paper the main aim is to provide machine learning based classification technique for cancerous profiles.

**EXISTING SYSTEM**

In early days’ Advancement in bioinformatics has raised the patient's life expectancy and boosted the treatment procedure for various chronic diseases. Screening of various diseases like diabetes, cancer and Cancer attack is no more a tedious task. Chip technology in healthcare has provided laboratory on-a-chip devices. These chips help predict the drug responses corresponding to the patient's genetic profile. All these technological advancements in the healthcare industry are helping in the earlier diagnosis and prognosis of stringent diseases like cancer.

**PROPOSED SYSTEM**

The proposed system represents a recent method that improved the algorithm's performance and accuracy in a distributed environment. The research proposes a system with a strong prediction algorithm, which implements powerful classification steps with a comprehensive report generation module. This project aims to implement a self-learning protocol such that the past inputs of the disease outcomes determine the future possibilities of cancer to a particular use. The proposed algorithm is divided into two sections. One is Dataset Pre-processing and Classification using machine learning algorithms.

**SOFTWARE REQUIREMENTS:**

Processor: Pentium IV

Hard Disk: 512GB or more

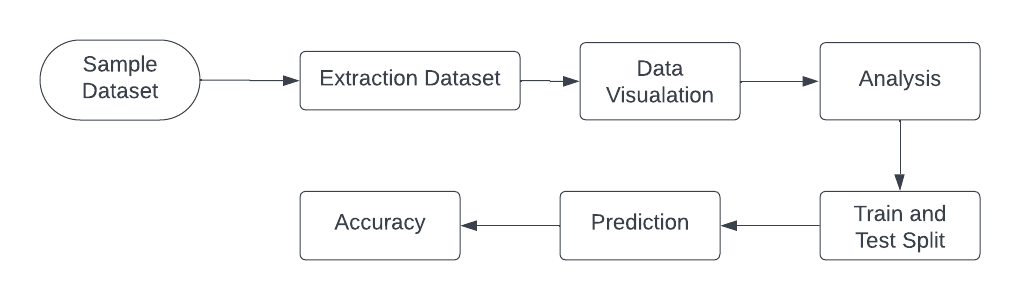
RAM: 8GB or more

Operating System: Windows 7, 10, 11,Linux

Programming Language: Python, HTML, CSS, Js Bootstrap, Django

IDE/Workbench: Pycharm, visual Studio code

**DATAFLOW DIAGRAM**



**PYTHON**

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

**Python Features**

Python has few keywords, simple structure, and a clearly defined syntax. Python code is more clearly defined and visible to the eyes. Python's source code is fairly easy-to-maintaining. Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh. Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.

Portable Python can run on a wide variety of hardware platforms and has the same interface on all platforms.

**Extendable**

It allows to add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.

**Databases**

Python provides interfaces to all major commercial databases.

**GUI Programming**

Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

**Scalable**

Python provides a better structure and support for large programs than shell scripting.

**Object-Oriented Approach**

One of the key aspects of Python is its object-oriented approach. This basically means that Python recognizes the concept of class and object encapsulation thus allowing programs to be efficient in the long run.

**Highly Dynamic**

Python is one of the most dynamic languages available in the industry today. There isno need to specify the type of the variable during coding, thus saving time and increasing efficiency.

**Extensive Array of Libraries**

Python comes inbuilt with many libraries that can be imported at any instance and be used in a specific program.

**Open Source and Free**

Python is an open-source programming language which means that anyone can create and contribute to its development. Python is free to download and use in any operating system, like Windows, Mac or Lin.

**MODULES**

A module allows you to logically organize your Python code. Grouping related code into a module makes

the code easier to understand and use. A module is a Python object with arbitrarily named attributes that

you can bind and reference. Simply, a module is a file consisting of Python code. A module can define

functions, classes and variables. A module can also include runnable code.

Example:

The Python code for a module named aname normally resides in a file named aname.py. Here's an

example of a simple module, support.py

def print\_func( par ):

print "Hello : ", par

return

The importStatement

The import has the following syntax:

import module1[, module2[,... moduleN]

When the interpreter encounters an import statement, it imports the module if the module is present in the

search path. A search path is a list of directories that the interpreter searches before importing a module.

For example, to import the module support.py, you need to put the following command at the top of the

script −

A module is loaded only once, regardless of the number of times it is imported. This prevents the module

execution from happening over and over again if multiple imports occur.

Packages in Python

A package is a hierarchical file directory structure that defines a single Python application environment

that consists of modules and sub packages and sub-sub packages.

Consider a file Pots.py available in Phone directory. This file has following line of source code −

def Pots():

print "I'm Pots Phone"

Similar way, we have another two files having different functions with the same name as above −

• Phone/Isdn.py file having function Isdn()

• Phone/G3.py file having function G3()

Now, create one more file \_\_init\_\_.py in Phone directory −

• Phone/\_\_init\_\_.py

To make all of your functions available when you've imported Phone,to put explicit import statements in

\_\_init\_\_.py as follows −

from Pots import Pots

from Isdn import Isdn

from G3 import G3

After you add these lines to \_\_init\_\_.py, you have all of these classes available when you import the

Phone package.

# Now import your Phone Package.

import Phone

Phone.Pots()

Phone.Isdn()

Phone.G3()

RESULT:

I'm Pots Phone

I'm 3G Phone

I'm ISDN Phone

In the above example, we have taken example of a single functions in each file, but you can keep multiple

functions in your files. You can also define different Python classes in those files and then you can create

your packages out of those classes.

**PYTHON FILES I/O**

This chapter covers all the basic I/O functions available in Python.

PRINTING TO THE SCREEN

The simplest way to produce output is using the print statement where you can pass zero or more

expressions separated by commas. This function converts the expressions you pass into a string and

writes the result to standard output as follows −

print "Python is really a great language,", "isn't it?"

Result:

Python is really a great language, isn't it?

READING KEYBOARD INPUT

Python provides two built-in functions to read a line of text from standard input, which by default comes

from the keyboard. These functions are −

• raw\_input

• input

Theraw\_inputFunction

The raw\_input([prompt]) function reads one line from standard input and returns it as a string (removing

the trailing newline).

str = raw\_input("Enter your input: ");

print "Received input is : ", str

This prompts you to enter any string and it would display same string on the screen. When I typed "Hello

Python!", its output is like this −

Enter your input: Hello Python

Received input is : Hello Python

22 | P a g e

The input Function

The input([prompt]) function is equivalent to raw\_input, except that it assumes the input is a valid Python

expression and returns the evaluated result to you.

str = input("Enter your input: ");

print "Received input is : ", str

This would produce the following result against the entered input −

Enter your input: [x\*5 for x in range(2,10,2)]

Recieved input is : [10, 20, 30, 40]

**Class and Object**

Python has been an object-oriented language since it existed. Because of this, creating and using classes

and objects are downright easy. This chapter helps you become an expert in using Python's objectoriented programming support.

If you do not have any previous experience with object-oriented (OO) programming, you may want to

consult an introductory course on it or at least a tutorial of some sort so that you have a grasp of the basic

concepts.

However, here is small introduction of Object-Oriented Programming (OOP) to bring you at speed −

Overview of OOP Terminology

• Class: A user-defined prototype for an object that defines a set of attributes that characterize

any object of the class. The attributes are data members (class variables and instance

variables) and methods, accessed via dot notation.

• Class variable: A variable that is shared by all instances of a class. Class variables are defined

within a class but outside any of the class's methods. Class variables are not used as frequently

as instance variables are.

• Data member: A class variable or instance variable that holds data associated with a class

and its objects.

• Function overloading: The assignment of more than one behavior to a particular function.

The operation performed varies by the types of objects or argument

• Instance variable: A variable that is defined inside a method and belongs only to the current

instance of a class.

• Inheritance: The transfer of the characteristics of a class to other classes that are derived from

it.

• Instance: An individual object of a certain class. An object obj that belongs to a class Circle,

for example, is an instance of the class Circle.

• Instantiation: The creation of an instance of a class.

• Method : A special kind of function that is defined in a class definition.

Object: A unique instance of a data structure that's defined by its class. An object comprises

both data members (class variables and instance variables) and methods.

• Operator overloading: The assignment of more than one function to a particular operator.

Creating Classes

The class statement creates a new class definition. The name of the class immediately follows the

keyword class followed by a colon as follows −

class ClassName:

'Optional class documentation string'

class\_suite

• The class has a documentation string, which can be accessed via ClassName.\_\_doc\_\_.

• The class\_suite consists of all the component statements defining class members, data

attributes and functions.

Class Inheritance

Instead of starting from scratch, you can create a class by deriving it from a preexisting class by listing

the parent class in parentheses after the new class name.

The child class inherits the attributes of its parent class, and you can use those attributes as if they were

defined in the child class. A child class can also override data members and methods from the parent.

Syntax

Derived classes are declared much like their parent class; however, a list of base classes to inherit from

is given after the class name −

class SubClassName (ParentClass1[, ParentClass2, ...]):

'Optional class documentation string'

class\_suite

Overriding Methods

You can always override your parent class methods. One reason for overriding parent's methods is

because you may want special or different functionality in your subclass.

Example

class Parent: # define parent class

def myMethod(self):

print 'Calling parent method'

class Child(Parent): # define child class

def myMethod(self):

print 'Calling child method'

c = Child() # instance of child

c.myMethod() # child calls overridden method

When the above code is executed, it produces the following result −

Calling child method

Base Overloading Methods

Following table lists some generic functionality that you can override in your own classes −

SN Method, Description & Sample Call

1 \_\_init\_\_ ( self [,args...] )

Constructor (with any optional arguments)

Sample Call : obj = className(args)

2 \_\_del\_\_( self )

Destructor, deletes an object

Sample Call : del obj

3 \_\_repr\_\_( self )

Evaluatable string representation

Sample Call : repr(obj)

4 \_\_str\_\_( self )

Printable string representation

Sample Call : str(obj)

5 \_\_cmp\_\_ ( self, x )

Object comparison

Sample Call : cmp(obj, x)

Overloading Operators

Suppose you have created a Vector class to represent two-dimensional vectors, what happens when you

use the plus operator to add them? Most likely Python will yell at you.

You could, however, define the \_\_add\_\_ method in your class to perform vector addition and then the

plus operator would behave as per expectation −

class Vector:

def \_\_init\_\_(self, a, b):

self.a = a

self.b = b

def \_\_str\_\_(self):

return 'Vector (%d, %d)' % (self.a, self.b)

def \_\_add\_\_(self,other):

return Vector(self.a + other.a, self.b + other.b)

v1 = Vector(2,10)

v2 = Vector(5,-2)

print v1 + v2

When the above code is executed, it produces the following result –

Data Hiding

An object's attributes may or may not be visible outside the class definition. You need to name attributes

with a double underscore prefix, and those attributes then are not be directly visible to outsiders.

lass JustCounter:

\_\_secretCount = 0

def count(self):

self.\_\_secretCount += 1

print self.\_\_secretCount

counter = JustCounter()

counter.count()

counter.count()

print counter.\_\_secretCount

1

2

Traceback (most recent call last):

File "test.py", line 12, in <module>

print counter.\_\_secretCount

AttributeError: JustCounter instance has no attribute '\_\_secretCount'

Python protects those members by internally changing the name to include the class name. You can

access such attributes as object.\_className\_\_attrName. If you would replace your last line as following,

then it works for you –

**NumPy**

”NumPy is a library for the Python programming language, adding support for large, multi- dimensional arrays and matrices, along with a large collection of high-level mathematical func- tions to operate on these arrays”. The previous similar programming of NumPy is Numeric, and this language was originally created by Jim Hugunin with contributions from several other developers. In 2005, Travis Oliphant created NumPy by incorporating features of the competing Numarray into Numeric, with extensive modifications. [12] It is an open source library and free of cost.

**Pandas**

Pandas is also a library or a data analysis tool in python which is written in python program- ming language. It is mostly used for data analysis and data manipulation. It is also used for data structures and time series.

We can see the application of python in many fields such as - Economics, Recommendation Systems - Spotify, Netflix and Amazon, Stock Prediction, Neuro science, Statistics, Advertising, Analytics, Natural Language Processing. Data can be analyzed in pandas in two ways -

Data frames - In this data is two dimensional and consist of multiple series. Data is always represented in rectangular table.

Series - In this data is one dimensional and consist of single list with index.

**Matplotlib**

”Matplotlib is a plotting library for the Python programming language and its numerical math- ematics extension NumPy”[11]. Matlab provides an application that is used in graphical user interface tool kits. Another such libraby is pylab which is almost same as MATLAB.

It is a library for 2D graphics, it finds its application in web application servers, graphical user interface toolkit and shell.Below is the example of a basic plot in python.

**SKLEARN**

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib.

What is Scikit-Learn (Sklearn)

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib.

Installation

If you already installed NumPy and Scipy, following are the two easiest ways to install scikit-learn −

Using pip

Following command can be used to install scikit-learn via pip −

pip install -U scikit-learn

Using conda

Following command can be used to install scikit-learn via conda −

conda install scikit-learn

On the other hand, if NumPy and Scipy is not yet installed on your Python workstation then, you can install them by using either pip or conda.

Another option to use scikit-learn is to use Python distributions like Canopy and Anaconda because they both ship the latest version of scikit-learn.

Features

Rather than focusing on loading, manipulating and summarising data, Scikit-learn library is focused on modeling the data. Some of the most popular groups of models provided by Sklearn are as follows −

Supervised Learning algorithms − Almost all the popular supervised learning algorithms, like Linear Regression, Support Vector Machine (SVM), Decision Tree etc., are the part of scikit-learn.

Unsupervised Learning algorithms − On the other hand, it also has all the popular unsupervised learning algorithms from clustering, factor analysis, PCA (Principal Component Analysis) to unsupervised neural networks.

Clustering − This model is used for grouping unlabeled data.

Cross Validation − It is used to check the accuracy of supervised models on unseen data.

Dimensionality Reduction − It is used for reducing the number of attributes in data which can be further used for summarisation, visualisation and feature selection.

Ensemble methods − As name suggest, it is used for combining the predictions of multiple supervised models.

Feature extraction − It is used to extract the features from data to define the attributes in image and text data.Dataset Loading

A collection of data is called dataset. It is having the following two components −

Features − The variables of data are called its features. They are also known as predictors, inputs or attributes.

Feature matrix − It is the collection of features, in case there are more than one.

Feature Names − It is the list of all the names of the features.

Response − It is the output variable that basically depends upon the feature variables. They are also known as target, label or output.

Response Vector − It is used to represent response column. Generally, we have just one response column.

Target Names − It represent the possible values taken by a response vector.

Scikit-learn have few example datasets like iris and digits for classification and the Boston house prices for regression.

**SYSTEM DESIGN**

The degree of interest in each concept has varied over the year, each has stood the test of time. Each provides the software designer with a foundation from which more sophisticated design methods can be applied. Fundamental design concepts provide the necessary framework for “getting it right”. During the design process the software requirements model is transformed into design models that describe the details of the data structures, system architecture, interface, and components. Each design product is reviewed for quality before moving to the next phase of software development.

**INPUT DESIGN**

The design of input focus on controlling the amount of dataset as input required, avoiding delay and keeping the process simple. The input is designed in such a way to provide security. Input design will consider the following steps:

1. The dataset should be given as input.
2. The dataset should be arranged.
3. Methods for preparing input validations.

**OUTPUT DESIGN**

A quality output is one, which meets the requirement of the user and presents the information clearly. In output design, it is determined how the information is to be displayed for immediate need. Designing computer output should proceed in an organized, well thought out manner the right output must be developed while ensuring that each output element is designed so that the user will find the system can be used easily and effectively.

**DATASET DESIGN**

This phase contains the attributes of the dataset which are maintained in the database table. The dataset collection can be of two types namely train dataset and test dataset.

**FEASIBILITY STUDY**

A feasibility analysis is used to determine the viability of an idea, such as ensuring a project is legally and technically feasible as well as economically justifiable. Feasibility study lets the developer to foresee the project and the usefulness of the system proposal as per its workability. It impacts the organization, ability to meet the user needs and effective use of resource. Thus, when a new application is proposed it normally goes through a feasibility study before it is approved for development.

Three key consideration involved in the feasibility analysis are,

1. TECHNICAL FEASIBILITY
2. OPERATIONAL FEASIBILITY
3. ECONOMIC FEASIBILITY

**TECHNICAL FEASIBILITY**

This phase focuses on the technical resources available to the organization. It helps organizations determine whether the technical resources meet capacity and whether the ideas can be converted into working system model. Technical feasibility also involves the evaluation of the hardware, software, and other technical requirements of

the proposed system.

**OPERATIONAL FEASIBILITY**

This phase involves undertaking a study to analyse and determine how well the organization’s needs can be met by completing the project. Operational feasibility study also examines how a project plan satisfies the requirements that are needed for the phase of system development.

**ECONOMIC FEASIBILITY**

This phase typically involves a cost benefits analysis of the project and help the organization to determine the viability, cost-benefits associated with a project before financial resources are allocated. It also serves as an independent project assessment and enhances project credibility. It helps the decision-makers to determine the positive economic benefits of the organization that the proposed project will provide

**SYSTEM TESTING**

System testing is the stage of implementation that is aimed at ensuring that the system works accurately and efficiently before live operation commences. Testing is vital to the success of the system. System testing makes logical assumption that if all the parts of the system are correct, then the goal will be successfully achieved. System testing involves user training system testing and successful running of the developed proposed system. The user tests the developed system and changes are made per their needs. The testing phase involves the testing of developed system using various kinds of data. While testing, errors are noted and the corrections are made. The corrections are also noted for the future use.

**UNIT TESTING**

Unit testing focuses verification effort on the smallest unit of software design, software component or module. Using the component level design description as a control paths are tested to uncover errors within the boundary of the module. The relative complexity of tests and the errors those uncover is limited by the constrained scope established for unit testing. The unit test focuses on the internal processing logic and data structures within the boundaries of a component. This is normally considered as an adjunct to the coding step. The design of unit tests can be performed before coding begins.

**BLACK BOX TESTING**

Black box testing also called behavioural testing, focuses on the functional requirement of the software. This testing enables to derive set of input conditions of all functional requirements for a program. This technique focuses on the information domain of the software, deriving test cases by partitioning the input and output of a program.

**WHITE BOX TESTING**

White box testing also called as glass box testing, is a test case design that uses the control structures described as part of component level design to derive test cases. This test case is derived to ensure all statements in the program have been executed at least once during the testing and that all logical conditions have been exercised.

**INTEGRATION TESTING**

Integration testing is a systematic technique for constructing the software architecture to conduct errors associated with interfacing. Top-down integration testing is an incremental approach to construction of the software architecture. Modules are integrated by moving ownward through the control hierarchy, beginning with the main control module. Bottom-up integration testing begins the construction and testing with atomic modules. Because components are integrated from the bottom up,

processing required for components subordinate to a given level is always available.

**VALIDATION TESTING**

Validation testing begins at the culmination of integration testing, when individual components have been exercised, the software is completely assembled as a package. The testing focuses on user visible actions and user recognizable output from the system. The testing has been conducted on possible condition such as the function characteristic conforms the specification and a deviation or error is uncovered. The alpha test and beta test is conducted at the developer site by end-users.

**CONCLUSION**

The proposed method, which used to address the very high dimensionality of the initial raw feature space followed by sparse feature learning techniques to construct discriminative and sparse features for the final classification step, provides the potential to overcome problems of traditional approaches with feature dimensionality as well as very limited size data sets. It does this by allowing data from different cancers and other tissue samples to be used during feature learning independently of their applicability to the final classification task. Applying this method to cancer data and comparing it to baseline algorithms, our method not only shows that it can be used to improve the accuracy in cancer classification problems, but also demonstrates that it provides a more general and scalable approach to deal with classification of cancer profile data across different cancer types.

**BIBLIOGRAPHY**

1. \*\*Dataset\*\*: Wisconsin Breast Cancer Dataset (https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic))

2. \*\*Research Papers\*\*:

- Mangasarian, O. L., & Wolberg, W. H. (1990). Cancer diagnosis via linear programming. SIAM News, 23(5), 1-18.

- Wolberg, W. H., Street, W. N., & Mangasarian, O. L. (1995). Machine learning techniques to diagnose breast cancer from fine-needle aspirates. Cancer Letters, 77(2-3), 163-171.

- Cruz-Roa, A., Basavanhally, A., Gonzalez, F., Gilmore, H., Feldman, M., Ganesan, S., ... & Madabhushi, A. (2014). Automatic detection of invasive ductal carcinoma in whole slide images with convolutional neural networks. In SPIE Medical Imaging (pp. 904103-904103). International Society for Optics and Photonics.

3. \*\*Books\*\*:

- Bishop, C. M. (2006). Pattern recognition and machine learning. springer.

- Raschka, S., & Mirjalili, V. (2019). Python machine learning. Packt Publishing Ltd.

4. \*\*Online Courses\*\*:

- Coursera - Machine Learning by Andrew Ng (https://www.coursera.org/learn/machine-learning)

- Udacity - Intro to Machine Learning (https://www.udacity.com/course/intro-to-machine-learning--ud120)

5. \*\*Review Articles\*\*:

- Esteva, A., Kuprel, B., Novoa, R. A., Ko, J., Swetter, S. M., Blau, H. M., & Thrun, S. (2017). Dermatologist-level classification of skin cancer with deep neural networks. Nature, 542(7639), 115-118.

- Litjens, G., Kooi, T., Bejnordi, B. E., Setio, A. A. A., Ciompi, F., Ghafoorian, M., ... & Sanchez, C. I. (2017). A survey on deep learning in medical image analysis. Medical image analysis, 42, 60-88.