**Prediction of time-to-event outcomes in diagnosing lung cancer based on SVM and compare the accuracy of predicted outcome with Deep CNN algorithm**

Abstract:-

This paper demonstrates a computer-aided diagnosis (CAD) system for lung cancer classifification of CT scans with unmarked nodules, a dataset from the Kaggle Data Science Bowl, 2017.

Thresholding was used as an initial segmentation approach to segment out lung tissue from the rest of the CT scan. Thresholding produced the next best lung segmentation. The initial approach was to directly feed the segmented CT scans into 3D CNNs for classifification, but this proved to be inadequate. Instead, a modifified U-Net trained on LUNA16 data (CT scans with labeled nodules) was used to fifirst detect nodule candidates in the Kaggle CT scans.

The U-Net nodule detection produced many false positives, so regions of CTs with segmented lungs where the most likely nodule candidates were located as determined by the U-Net output were fed into 3D Convolutional Neural Networks (CNNs) to ultimately classify the CT scan as positive or negative for lung cancer. The 3D CNNs produced a test set Accuracy of 86.6%. The performance of our CAD system outperforms the current CAD systems in literature which have several training and testing phases that each requires a lot of labeled data, while our CAD system has only three major phases (segmentation, nodule candidate detection, and malignancy classifification), allowing more effificient training and detection and more generalizability to other cancers.

**INTRODUCTION**

Lung cancer is known as a disease that consists of uncontrollable growth of lung cells which may lead to metastasis. Metastasis is the infestation of adjacent and nearby tissue and infiltration beyond the lungs. From epithelial cells Carcinomas are derived which are the vast majority of primary lung cancers. Lung cancer is the most usual cause of cancer-imputed death in men and women. An estimated number of new lung cancer cases, by sex type, are 119,100 for males and 116,660 for females in the US [1].

The early detection of lung cancer can increase overall 5-year survival rates by extracting the lung nodules and distinguishing their location for surgery. Hence, this diagnosis according to nodule location (solitary and juxtapleural) can improve the treatment. In this paper, we aim to classify small lung masses as nodule or non-nodule by using a Computer Aided Diagnosis (CADx) system. Also classify this nodule as a solitary nodule or juxtapleural nodule.

Traditional X-ray and computed tomography (CT scan) is attempted to diagnose lung nodules [2]. The most accurate modality for imaging lung nodules is CT. It allows the detection of a small lung nodule, but a large amount of data leads to a high false negative rate to detect the small nodules. Computer Aided Diagnosis (CADx) system is one of the robust systems which is used in the detection and diagnosis of lung cancer. Because of the nodule’s small size in the lung, it is difficult to distinguish between it and another mass in a 2D slice. Actually, micro-nodules cannot be recognized on single slices: the nodule shape, size, and gray tone are very similar to vessel sections. Therefore, segmentation is a very important step to distinguish between the small nodules and blood vessels. Hence, the type of nodules according to their location (solitary or juxtapleural) will be easily classified.

In this study, two main schemes of supervised learning for classification are proposed in which; two segmentation approaches are achieved (Thresholding ? K-means clustering and Bounding box ? Maximum intensity projection) for both two schemes. For the first scheme, a combination of the first-order and second-order features is extracted. Fisher score ranking is used as a feature selection method. The higher five, ten, and fifteen ranks of features are selected, respectively, from the two sets of features. The first scheme is implemented by using Support Vector Machine (SVM) classifier. The second scheme used Deep Convolutional neural network (DCNN) for deep learning classification. Tenfold cross-validation and random oversampling are used to manage limited and imbalanced data. ROC curve is used to evaluate the DCNN classifier. The general block diagram which represents the methodology and system overview is shown in Fig. 1. Our algorithm outperforms MIP as a 2D segmentation technique achieving good results although it is generally used in a 3D volume rendering. Also, achievement of promising results when using a quite small dataset with DCNN.

**Related work**

To date, many types of research about the detection of the nodule by using CAD systems have been developed. It begins with preprocessing and segmentation followed by the classification step. For instance, Diego et al. [3]

used a method composed of four steps for lung nodule detection. The first employed acquisition of an image and pre-processing. The second phase involved a 2D algorithm to inspire every layer of a scan which eliminates the noninformative structures in the. The final step utilized a support vector machine for separating the candidate masses into nodules and non-nodules according to their features. QingZeng et al. [4]

proposed a stacked autoencoder (SAE), convolution neural network (CNN), and deep neural network (DNN), respectively, to detect pulmonary nodules. The results showed that the CNN network achieved the best performance with an accuracy of 84.15%, sensitivity of 83.96%, and specificity of 84.32%, which induces the best result among the three networks. Yang et al. [5]

described a 3D detection of pulmonary nodule scheme using MSCT images. This method segmented the candidate nodules first and extracted voxels feature based on eigenvalues analysis. Then support vector machine (SVM) and decision rule are applied to reduce the false positives. Sarah et al. [6]

used a Gaussian smoothing kernel for filtration which helps to reduce noise effects. Next, features such as sphericity, mean and variance of the gray level, elongation, and border variation of potential nodules are extracted to classify detected nodules as malignant and benign tumors. Fuzzy KNN is employed to classify potential nodules as non-nodule or nodules with different levels of malignancy. She achieved a sensitivity of 88% for nodule detection with approximately 10.3 False-Positive (FP)/subject. Serhat et al. [7]

used Genetic Cellular Neural Networks (G-CNN) for segmentation. A 3D template was made to find the structures which are like a nodule in the 3D image. The computer-aided diagnosis (CAD) system achieved 100% sensitivity with 13.375 FPs per case when the nodule thickness was greater than or equal to 5.625 mm. To test the system’s efficiency, they used 16 cases with a total of 425 slices, which were taken from the Lung Image Database Consortium (LIDC) dataset. Jin et al. [8]

put forward a kind of lung segmentation method based on morphology and statistic of the size of the image area, while effectively eliminating the influence of the trachea on pulmonary parenchyma image segmentation. He also proposed a method of the region of interest(ROI) extraction based on morphology and circular filter, reducing the number of false positives and trying to retain the integrity of the ROI form. Finally, he has realized reliable lung nodules compute aided diagnosis application on CT image, using convolution neural network. The system achieved 84.6% of accuracy, 86.7% of specificity, and 82.5% of sensitivity. Thomas et al. [9]

developed a CAD system that can localize as many nodules as possible while keeping the number of false positives low. To localize the nodules in the three-dimensional space of a scan, a quantile approach over the cross-sectional slices is applied to retrieve a twodimensional slice. Next, a sliding window method on the slice is used to obtain the (x, y) coordinates of the nodule.

**SOFTWARE ENVIRONMENT**

**PYTHON**

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. An [interpreted language](https://en.wikipedia.org/wiki/Interpreted_language" \o "Interpreted language), Python has a design philosophy that emphasizes code [readability](https://en.wikipedia.org/wiki/Readability" \o "Readability) (notably using [whitespace](https://en.wikipedia.org/wiki/Whitespace_character" \o "Whitespace character) indentation to delimit [code blocks](https://en.wikipedia.org/wiki/Code_block" \o "Code block) rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer [lines of code](https://en.wikipedia.org/wiki/Source_lines_of_code" \o "Source lines of code) than might be used in languages such as [C++](https://en.wikipedia.org/wiki/C%2B%2B" \o "C++)or [Java](https://en.wikipedia.org/wiki/Java_(programming_language)" \o "Java (programming language)). It provides constructs that enable clear programming on both small and large scales. Python interpreters are available for many [operating systems](https://en.wikipedia.org/wiki/Operating_system" \o "Operating system). [CPython](https://en.wikipedia.org/wiki/CPython" \o "CPython), the [reference implementation](https://en.wikipedia.org/wiki/Reference_implementation" \o "Reference implementation) of Python, is [open source](https://en.wikipedia.org/wiki/Open_source" \o "Open source) software and has a community-based development model, as do nearly all of its variant implementations. CPython is managed by the non-profit [Python Software Foundation](https://en.wikipedia.org/wiki/Python_Software_Foundation" \o "Python Software Foundation). Python features a [dynamic type](https://en.wikipedia.org/wiki/Dynamic_type" \o "Dynamic type) system and automatic [memory management](https://en.wikipedia.org/wiki/Memory_management" \o "Memory management). It supports multiple [programming paradigms](https://en.wikipedia.org/wiki/Programming_paradigm" \o "Programming paradigm), including [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming" \o "Object-oriented programming), [imperative](https://en.wikipedia.org/wiki/Imperative_programming" \o "Imperative programming), [functional](https://en.wikipedia.org/wiki/Functional_programming" \o "Functional programming) and [procedural](https://en.wikipedia.org/wiki/Procedural_programming" \o "Procedural programming), and has a large and comprehensive [standard library](https://en.wikipedia.org/wiki/Standard_library" \o "Standard library)

**DJANGO**

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It’s free and open source.

Django's primary goal is to ease the creation of complex, database-driven websites. Django emphasizes [reusability](https://en.wikipedia.org/wiki/Reusability" \o "Reusability)and "pluggability" of components, rapid development, and the principle of [don't repeat yourself](https://en.wikipedia.org/wiki/Don%27t_repeat_yourself" \o "Don't repeat yourself). Python is used throughout, even for settings files and data models.



Django also provides an optional administrative [create, read, update and delete](https://en.wikipedia.org/wiki/Create,_read,_update_and_delete" \o "Create, read, update and delete) interface that is generated dynamically through [introspection](https://en.wikipedia.org/wiki/Introspection_(computer_science)" \o "Introspection (computer science)) and configured via admin models



**SOFTWARE DEVELOPMENT**

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. An [interpreted language](https://en.wikipedia.org/wiki/Interpreted_language" \o "Interpreted language), Python has a design philosophy that emphasizes code [readability](https://en.wikipedia.org/wiki/Readability" \o "Readability) (notably using [whitespace](https://en.wikipedia.org/wiki/Whitespace_character" \o "Whitespace character) indentation to delimit [code blocks](https://en.wikipedia.org/wiki/Code_block" \o "Code block) rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer [lines of code](https://en.wikipedia.org/wiki/Source_lines_of_code" \o "Source lines of code) than might be used in languages such as [C++](https://en.wikipedia.org/wiki/C%2B%2B" \o "C++)or [Java](https://en.wikipedia.org/wiki/Java_(programming_language)" \o "Java (programming language)). It provides constructs that enable clear programming on both small and large scales. Python interpreters are available for many [operating systems](https://en.wikipedia.org/wiki/Operating_system" \o "Operating system). [CPython](https://en.wikipedia.org/wiki/CPython" \o "CPython), the [reference implementation](https://en.wikipedia.org/wiki/Reference_implementation" \o "Reference implementation) of Python, is [open source](https://en.wikipedia.org/wiki/Open_source" \o "Open source) software and has a community-based development model, as do nearly all of its variant implementations. CPython is managed by the non-profit [Python Software Foundation](https://en.wikipedia.org/wiki/Python_Software_Foundation" \o "Python Software Foundation). Python features a [dynamic type](https://en.wikipedia.org/wiki/Dynamic_type" \o "Dynamic type) system and automatic [memory management](https://en.wikipedia.org/wiki/Memory_management" \o "Memory management). It supports multiple [programming paradigms](https://en.wikipedia.org/wiki/Programming_paradigm" \o "Programming paradigm), including [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming" \o "Object-oriented programming), [imperative](https://en.wikipedia.org/wiki/Imperative_programming" \o "Imperative programming), [functional](https://en.wikipedia.org/wiki/Functional_programming" \o "Functional programming) and [procedural](https://en.wikipedia.org/wiki/Procedural_programming" \o "Procedural programming), and has a large and comprehensive [standard library](https://en.wikipedia.org/wiki/Standard_library" \o "Standard library)

## What is Python

## Python is a popular programming language. It was created by Guido van Rossum, and released in 1991.

**It is used for:**

* web development (server-side),
* software development,
* mathematics,
* system scripting.

### What can Python do

* Python can be used on a server to create web applications.
* Python can be used alongside software to create workflows.
* Python can connect to database systems. It can also read and modify files.
* Python can be used to handle big data and perform complex mathematics.
* Python can be used for rapid prototyping, or for production-ready software development.

### Why Python

* Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
* Python has a simple syntax similar to the English language.
* Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
* Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
* Python can be treated in a procedural way, an object-orientated way or a functional way.

### Good to know

* The most recent major version of Python is Python 3, which we shall be using in this tutorial. However, Python 2, although not being updated with anything other than security updates, is still quite popular.
* In this tutorial Python will be written in a text editor. It is possible to write Python in an Integrated Development Environment, such as Thonny, Pycharm, Netbeans or Eclipse which are particularly useful when managing larger collections of Python files.

### Python Syntax compared to other programming languages

* Python was designed for readability, and has some similarities to the English language with influence from mathematics.
* Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.
* Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

**Python Install**

Many PCs and Macs will have python already installed.

To check if you have python installed on a Windows PC, search in the start bar for Python or run the following on the Command Line (cmd.exe):

C:\Users\Your Name>python --version

To check if you have python installed on a Linux or Mac, then on linux open the command line or on Mac open the Terminal and type:

python --version

If you find that you do not have python installed on your computer, then you can download it for free from the following website: https://www.python.org/

Python Quickstart

Python is an interpreted programming language, this means that as a developer you write Python (.py) files in a text editor and then put those files into the python interpreter to be executed.

The way to run a python file is like this on the command line:

C:\Users\Your Name>python helloworld.py

Where "helloworld.py" is the name of your python file.

Let's write our first Python file, called helloworld.py, which can be done in any text editor.

helloworld.py

print("Hello, World!")

Simple as that. Save your file. Open your command line, navigate to the directory where you saved your file, and run:

C:\Users\Your Name>python helloworld.py

The output should read:

Hello, World!

Congratulations, you have written and executed your first Python program.

The Python Command Line

To test a short amount of code in python sometimes it is quickest and easiest not to write the code in a file. This is made possible because Python can be run as a command line itself.

Type the following on the Windows, Mac or Linux command line:

C:\Users\Your Name>python

Or, if the "python" command did not work, you can try "py":

C:\Users\Your Name>py

From there you can write any python, including our hello world example from earlier in the tutorial:

C:\Users\Your Name>python

Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32 bit (Intel)] on win32

Type "help", "copyright", "credits" or "license" for more information.

>>>print("Hello, World!")

Which will write "Hello, World!" in the command line:

C:\Users\Your Name>python

Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32 bit (Intel)] on win32

Type "help", "copyright", "credits" or "license" for more information.

>>>print("Hello, World!")

Hello, World!

Whenever you are done in the python command line, you can simply type the following to quit the python command line interface:

exit()

**Virtual Environments and Packages**

**Introduction**

Python applications will often use packages and modules that don’t come as part of the standard library. Applications will sometimes need a specific version of a library, because the application may require that a particular bug has been fixed or the application may be written using an obsolete version of the library’s interface.

This means it may not be possible for one Python installation to meet the requirements of every application. If application A needs version 1.0 of a particular module but application B needs version 2.0, then the requirements are in conflict and installing either version 1.0 or 2.0 will leave one application unable to run.

The solution for this problem is to create a virtual environment, a self-contained directory tree that contains a Python installation for a particular version of Python, plus a number of additional packages.

Different applications can then use different virtual environments. To resolve the earlier example of conflicting requirements, application A can have its own virtual environment with version 1.0 installed while application B has another virtual environment with version 2.0. If application B requires a library be upgraded to version 3.0, this will not affect application A’s environment.

**Creating Virtual Environments**

The module used to create and manage virtual environments is called venv. venv will usually install the most recent version of Python that you have available. If you have multiple versions of Python on your system, you can select a specific Python version by running python3 or whichever version you want.

To create a virtual environment, decide upon a directory where you want to place it, and run the venv module as a script with the directory path:

python3 -m venv tutorial-env

This will create the tutorial-env directory if it doesn’t exist, and also create directories inside it containing a copy of the Python interpreter, the standard library, and various supporting files.

A common directory location for a virtual environment is .venv. This name keeps the directory typically hidden in your shell and thus out of the way while giving it a name that explains why the directory exists. It also prevents clashing with .env environment variable definition files that some tooling supports.

Once you’ve created a virtual environment, you may activate it.

On Windows, run:

tutorial-env\Scripts\activate.bat

On Unix or MacOS, run:

source tutorial-env/bin/activate

(This script is written for the bash shell. If you use the csh or fish shells, there are alternate activate.csh and activate.fish scripts you should use instead.)

Activating the virtual environment will change your shell’s prompt to show what virtual environment you’re using, and modify the environment so that running python will get you that particular version and installation of Python. For example:

$ source ~/envs/tutorial-env/bin/activate

(tutorial-env) $ python

Python 3.5.1 (default, May 6 2016, 10:59:36)

...

>>> import sys

>>>sys.path

['', '/usr/local/lib/python35.zip', ...,

'~/envs/tutorial-env/lib/python3.5/site-packages']

>>>

12.3. Managing Packages with pip

You can install, upgrade, and remove packages using a program called pip. By default pip will install packages from the Python Package Index, <https://pypi.org>. You can browse the Python Package Index by going to it in your web browser, or you can use pip’s limited search feature:

(tutorial-env) $ pip search astronomy

skyfield - Elegant astronomy for Python

gary - Galactic astronomy and gravitational dynamics.

novas - The United States Naval Observatory NOVAS astronomy library

astroobs - Provides astronomy ephemeris to plan telescope observations

PyAstronomy - A collection of astronomy related tools for Python.

...

pip has a number of subcommands: “search”, “install”, “uninstall”, “freeze”, etc. (Consult the Installing Python Modules guide for complete documentation for pip.)

You can install the latest version of a package by specifying a package’s name:

(tutorial-env) $ pip install novas

Collecting novas

Downloading novas-3.1.1.3.tar.gz (136kB)

Installing collected packages: novas

Running setup.py install for novas

Successfully installed novas-3.1.1.3

You can also install a specific version of a package by giving the package name followed by == and the version number:

(tutorial-env) $ pip install requests==2.6.0

Collecting requests==2.6.0

Using cached requests-2.6.0-py2.py3-none-any.whl

Installing collected packages: requests

Successfully installed requests-2.6.0

If you re-run this command, pip will notice that the requested version is already installed and do nothing. You can supply a different version number to get that version, or you can run pip install --upgrade to upgrade the package to the latest version:

(tutorial-env) $ pip install --upgrade requests

Collecting requests

Installing collected packages: requests

Found existing installation: requests 2.6.0

Uninstalling requests-2.6.0:

Successfully uninstalled requests-2.6.0

Successfully installed requests-2.7.0

pip uninstall followed by one or more package names will remove the packages from the virtual environment.

pip show will display information about a particular package:

(tutorial-env) $ pip show requests

---

Metadata-Version: 2.0

Name: requests

Version: 2.7.0

Summary: Python HTTP for Humans.

Home-page: http://python-requests.org

Author: Kenneth Reitz

Author-email: me@kennethreitz.com

License: Apache 2.0

Location: /Users/akuchling/envs/tutorial-env/lib/python3.4/site-packages

Requires:

pip list will display all of the packages installed in the virtual environment:

(tutorial-env) $ pip list

novas (3.1.1.3)

numpy (1.9.2)

pip (7.0.3)

requests (2.7.0)

setuptools (16.0)

pip freeze will produce a similar list of the installed packages, but the output uses the format that pip install expects. A common convention is to put this list in a requirements.txt file:

(tutorial-env) $ pip freeze > requirements.txt

(tutorial-env) $ cat requirements.txt

novas==3.1.1.3

numpy==1.9.2

requests==2.7.0

The requirements.txt can then be committed to version control and shipped as part of an application. Users can then install all the necessary packages with install -r:

(tutorial-env) $ pip install -r requirements.txt

Collecting novas==3.1.1.3 (from -r requirements.txt (line 1))

...

Collecting numpy==1.9.2 (from -r requirements.txt (line 2))

...

Collecting requests==2.7.0 (from -r requirements.txt (line 3))

...

Installing collected packages: novas, numpy, requests

Running setup.py install for novas

Successfully installed novas-3.1.1.3 numpy-1.9.2 requests-2.7.0

pip has many more options. Consult the Installing Python Modules guide for complete documentation for pip. When you’ve written a package and want to make it available on the Python Package Index, consult the Distributing Python Modules guide.

**Cross Platform**

Platform. Architecture (executable=sys.executable, bits='', linkage='')

Queries the given executable (defaults to the Python interpreter binary) for various architecture information.

Returns a tuple (bits, linkage) which contain information about the bit architecture and the linkage format used for the executable. Both values are returned as strings.

Values that cannot be determined are returned as given by the parameter presets. If bits is given as '', the sizeof(pointer) (or sizeof(long) on Python version < 1.5.2) is used as indicator for the supported pointer size.

The function relies on the system’s file command to do the actual work. This is available on most if not all Unix platforms and some non-Unix platforms and then only if the executable points to the Python interpreter. Reasonable defaults are used when the above needs are not met.

Note On Mac OS X (and perhaps other platforms), executable files may be universal files containing multiple architectures.

To get at the “64-bitness” of the current interpreter, it is more reliable to query the sys.maxsize attribute:

is\_64bits = sys.maxsize> 2\*\*32

platform.machine ()

Returns the machine type, e.g. 'i386'. An empty string is returned if the value cannot be determined.

platform.node ()

Returns the computer’s network name (may not be fully qualified!). An empty string is returned if the value cannot be determined.

platform. Platform(aliased=0, terse=0)

Returns a single string identifying the underlying platform with as much useful information as possible.

The output is intended to be human readable rather than machine parseable. It may look different on different platforms and this is intended.

If aliased is true, the function will use aliases for various platforms that report system names which differ from their common names, for example SunOS will be reported as Solaris. The system\_alias() function is used to implement this.

Setting terse to true causes the function to return only the absolute minimum information needed to identify the platform.

platform.processor()

Returns the (real) processor name, e.g. 'amdk6'.

An empty string is returned if the value cannot be determined. Note that many platforms do not provide this information or simply return the same value as for machine(). NetBSD does this.

platform.python\_build()

Returns a tuple (buildno, builddate) stating the Python build number and date as strings.

platform.python\_compiler()

Returns a string identifying the compiler used for compiling Python.

platform.python\_branch()

Returns a string identifying the Python implementation SCM branch.

New in version 2.6.

platform.python\_implementation()

Returns a string identifying the Python implementation. Possible return values are: ‘CPython’, ‘IronPython’, ‘Jython’, ‘PyPy’.

New in version 2.6.

platform.python\_revision()

Returns a string identifying the Python implementation SCM revision.

New in version 2.6.

platform.python\_version()

Returns the Python version as string 'major.minor.patchlevel'.

Note that unlike the Python sys.version, the returned value will always include the patchlevel (it defaults to 0).

platform.python\_version\_tuple()

Returns the Python version as tuple (major, minor, patchlevel) of strings.

Note that unlike the Python sys.version, the returned value will always include the patchlevel (it defaults to '0').

platform.release()

Returns the system’s release, e.g. '2.2.0' or 'NT' An empty string is returned if the value cannot be determined.

platform.system()

Returns the system/OS name, e.g. 'Linux', 'Windows', or 'Java'. An empty string is returned if the value cannot be determined.

platform.system\_alias(system, release, version)

Returns (system, release, version) aliased to common marketing names used for some systems. It also does some reordering of the information in some cases where it would otherwise cause confusion.

platform.version()

Returns the system’s release version, e.g. '#3 on degas'. An empty string is returned if the value cannot be determined.

platform.uname()

Fairly portable uname interface. Returns a tuple of strings (system, node, release, version, machine, processor) identifying the underlying platform.

Note that unlike the os.uname() function this also returns possible processor information as additional tuple entry.

Entries which cannot be determined are set to ''.

**Java Platform**

platform.java\_ver(release='', vendor='', vminfo=('', '', ''), osinfo=('', '', ''))

Version interface for Jython.

Returns a tuple (release, vendor, vminfo, osinfo) with vminfo being a tuple (vm\_name, vm\_release, vm\_vendor) and osinfo being a tuple (os\_name, os\_version, os\_arch). Values which cannot be determined are set to the defaults given as parameters (which all default to '').

Windows Platform

platform.win32\_ver(release='', version='', csd='', ptype='')

Get additional version information from the Windows Registry and return a tuple (release, version, csd, ptype) referring to OS release, version number, CSD level (service pack) and OS type (multi/single processor).

As a hint: ptype is 'Uniprocessor Free' on single processor NT machines and 'Multiprocessor Free' on multi processor machines. The ‘Free’ refers to the OS version being free of debugging code. It could also state ‘Checked’ which means the OS version uses debugging code, i.e. code that checks arguments, ranges, etc.

Note This function works best with Mark Hammond’s win32all package installed, but also on Python 2.3 and later (support for this was added in Python 2.6). It obviously only runs on Win32 compatible platforms.

**Win95/98 specific**

platform.popen(cmd, mode='r', bufsize=None)

Portable popen() interface. Find a working popen implementation preferring win32pipe.popen(). On Windows NT, win32pipe.popen() should work; on Windows 9x it hangs due to bugs in the MS C library.

**Mac OS Platform**

platform.mac\_ver(release='', versioninfo=('', '', ''), machine='')

Get Mac OS version information and return it as tuple (release, versioninfo, machine) with versioninfo being a tuple (version, dev\_stage, non\_release\_version).

Entries which cannot be determined are set to ''. All tuple entries are strings.

**Unix Platforms**

platform.dist(distname='', version='', id='', supported\_dists=('SuSE', 'debian', 'redhat', 'mandrake', ...))

This is an old version of the functionality now provided by linux\_distribution(). For new code, please use the linux\_distribution().

The only difference between the two is that dist() always returns the short name of the distribution taken from the supported\_dists parameter.

Deprecated since version 2.6.

platform.linux\_distribution(distname='', version='', id='', supported\_dists=('SuSE', 'debian', 'redhat', 'mandrake', ...), full\_distribution\_name=1)

Tries to determine the name of the Linux OS distribution name.

supported\_dists may be given to define the set of Linux distributions to look for. It defaults to a list of currently supported Linux distributions identified by their release file name.

If full\_distribution\_name is true (default), the full distribution read from the OS is returned. Otherwise the short name taken from supported\_dists is used.

Returns a tuple (distname,version,id) which defaults to the args given as parameters. id is the item in parentheses after the version number. It is usually the version codename.

Note This function is deprecated since Python 3.5 and removed in Python 3.8. See alternative like the distro package.

New in version 2.6.

platform.libc\_ver(executable=sys.executable, lib='', version='', chunksize=2048)

Tries to determine the libc version against which the file executable (defaults to the Python interpreter) is linked. Returns a tuple of strings (lib, version) which default to the given parameters in case the lookup fails.

Note that this function has intimate knowledge of how different libc versions add symbols to the executable is probably only usable for executables compiled using gcc. The file is read and scanned in chunks of chunksize bytes.

**2. Using the Python Interpreter**

**2.1. Invoking the Interpreter**

The Python interpreter is usually installed as /usr/local/bin/python3.8 on those machines where it is available; putting /usr/local/bin in your Unix shell’s search path makes it possible to start it by typing the command:

python3.8

to the shell. 1 Since the choice of the directory where the interpreter lives is an installation option, other places are possible; check with your local Python guru or system administrator. (E.g., /usr/local/python is a popular alternative location.)

On Windows machines where you have installed Python from the Microsoft Store, the python3.8 command will be available. If you have the py.exe launcher installed, you can use the py command. See Excursus: Setting environment variables for other ways to launch Python.

Typing an end-of-file character (Control-D on Unix, Control-Z on Windows) at the primary prompt causes the interpreter to exit with a zero exit status. If that doesn’t work, you can exit the interpreter by typing the following command: quit().

The interpreter’s line-editing features include interactive editing, history substitution and code completion on systems that support the GNU Readline library. Perhaps the quickest check to see whether command line editing is supported is typing Control-P to the first Python prompt you get. If it beeps, you have command line editing; see Appendix Interactive Input Editing and History Substitution for an introduction to the keys. If nothing appears to happen, or if ^P is echoed, command line editing isn’t available; you’ll only be able to use backspace to remove characters from the current line.

The interpreter operates somewhat like the Unix shell: when called with standard input connected to a tty device, it reads and executes commands interactively; when called with a file name argument or with a file as standard input, it reads and executes a script from that file.

A second way of starting the interpreter is python -c command [arg] ..., which executes the statement(s) in command, analogous to the shell’s -c option. Since Python statements often contain spaces or other characters that are special to the shell, it is usually advised to quote command in its entirety with single quotes.

Some Python modules are also useful as scripts. These can be invoked using python -m module [arg] ..., which executes the source file for module as if you had spelled out its full name on the command line.

When a script file is used, it is sometimes useful to be able to run the script and enter interactive mode afterwards. This can be done by passing -i before the script.

All command line options are described in Command line and environment.

Argument Passing

When known to the interpreter, the script name and additional arguments thereafter are turned into a list of strings and assigned to the argv variable in the sys module. You can access this list by executing import sys. The length of the list is at least one; when no script and no arguments are given, sys.argv[0] is an empty string. When the script name is given as '-' (meaning standard input), sys.argv[0] is set to '-'. When -c command is used, sys.argv[0] is set to '-c'. When -m module is used, sys.argv[0] is set to the full name of the located module. Options found after -c command or -m module are not consumed by the Python interpreter’s option processing but left in sys.argv for the command or module to handle.

Interactive Mode

When commands are read from a tty, the interpreter is said to be in interactive mode. In this mode it prompts for the next command with the primary prompt, usually three greater-than signs (>>>); for continuation lines it prompts with the secondary prompt, by default three dots (...). The interpreter prints a welcome message stating its version number and a copyright notice before printing the first prompt:

$ python3.8

Python 3.8 (default, Sep 16 2015, 09:25:04)

[GCC 4.8.2] on linux

Type "help", "copyright", "credits" or "license" for more information.

>>>

Continuation lines are needed when entering a multi-line construct. As an example, take a look at this if statement:

>>>

>>>the\_world\_is\_flat = True

>>>ifthe\_world\_is\_flat:

... print("Be careful not to fall off!")

...

Be careful not to fall off!

For more on interactive mode, see Interactive Mode.

**2.2. The Interpreter and Its Environment**

**2.2.1. Source Code Encoding**

By default, Python source files are treated as encoded in UTF-8. In that encoding, characters of most languages in the world can be used simultaneously in string literals, identifiers and comments — although the standard library only uses ASCII characters for identifiers, a convention that any portable code should follow. To display all these characters properly, your editor must recognize that the file is UTF-8, and it must use a font that supports all the characters in the file.

To declare an encoding other than the default one, a special comment line should be added as the first line of the file. The syntax is as follows:

# -\*- coding: encoding -\*-

where encoding is one of the valid codecs supported by Python.

For example, to declare that Windows-1252 encoding is to be used, the first line of your source code file should be:

# -\*- coding: cp1252 -\*-

One exception to the first line rule is when the source code starts with a UNIX “shebang” line. In this case, the encoding declaration should be added as the second line of the file. For example:

#!/usr/bin/env python3

# -\*- coding: cp1252 -\*-

# **Introduction to Artificial Intelligence**

“The science and engineering of making intelligent machines, especially intelligent computer programs”. -John McCarthy-

Artificial Intelligence is an approach to make a computer, a robot, or a product to think how smart human think. AI is a study of how human brain think, learn, decide and work, when it tries to solve problems. And finally this study outputs intelligent software systems.The aim of AI is to improve computer functions which are related to human knowledge, for example, reasoning, learning, and problem-solving.

The intelligence is intangible. It is composed of

* Reasoning
* Learning
* Problem Solving
* Perception
* Linguistic Intelligence

The objectives of AI research are reasoning, knowledge representation, planning, learning, natural language processing, realization, and ability to move and manipulate objects. There are long-term goals in the general intelligence sector.

Approaches include statistical methods, computational intelligence, and traditional coding AI. During the AI research related to search and mathematical optimization, artificial neural networks and methods based on statistics, probability, and economics, we use many tools. Computer science attracts AI in the field of science, mathematics, psychology, linguistics, philosophy and so on.

# Trending AI Articles:

[1. Cheat Sheets for AI, Neural Networks, Machine Learning, Deep Learning & Big Data](https://becominghuman.ai/cheat-sheets-for-ai-neural-networks-machine-learning-deep-learning-big-data-678c51b4b463" \t "_blank)

[2. Data Science Simplified Part 1: Principles and Process](https://becominghuman.ai/data-science-simplified-principles-and-process-b06304d63308" \t "_blank)

[3. Getting Started with Building Realtime API Infrastructure](https://becominghuman.ai/getting-started-with-building-realtime-api-infrastructure-a19601fc794e" \t "_blank)

[4. AI & NLP Workshop](https://becominghuman.ai/ai-nlp-workshop-7bc121986d61" \t "_blank)

**Applications of AI**

· Gaming − AI plays important role for machine to think of large number of possible positions based on deep knowledge in strategic games. for example, chess,river crossing, N-queens problems and etc.

Natural Language Processing − Interact with the computer that understands natural language spoken by humans.

· Expert Systems − Machine or software provide explanation and advice to the users.

· Vision Systems − Systems understand, explain, and describe visual input on the computer.

· Speech Recognition − There are some AI based speech recognition systems have ability to hear and express as sentences and understand their meanings while a person talks to it. For example Siri and Google assistant.

· Handwriting Recognition − The handwriting recognition software reads the text written on paper and recognize the shapes of the letters and convert it into editable text.

· Intelligent Robots − Robots are able to perform the instructions given by a human.

**Major Goals**

* Knowledge reasoning
* Planning
* Machine Learning
* Natural Language Processing
* Computer Vision
* Robotics

**IBM Watson**



“Watson” is an IBM supercomputer that combines Artificial Intelligence (AI) and complex inquisitive programming for ideal execution as a “question answering” machine. The supercomputer is named for IBM’s founder, Thomas J. Watson.

IBM Watson is at the forefront of the new era of computing. At the point when IBM Watson made, IBM communicated that “more than 100 particular techniques are used to inspect perceive sources, find and make theories, find and score affirm, and combination and rank speculations.” recently, the Watson limits have been expanded and the way by which Watson works has been changed to abuse new sending models (Watson on IBM Cloud) and propelled machine learning capacities and upgraded hardware open to architects and authorities. It isn’t any longer completely a request answering figuring system arranged from Q&A joins yet can now ‘see’, ‘hear’, ‘read’, ‘talk’, ‘taste’, ‘translate’, ‘learn’ and ‘endorse’.

# Machine Learning

### Introduction

Machine learning is a subfield of artificial intelligence (AI). The goal of machine learning generally is to understand the structure of data and fit that data into models that can be understood and utilized by people.

Although machine learning is a field within computer science, it differs from traditional computational approaches. In traditional computing, algorithms are sets of explicitly programmed instructions used by computers to calculate or problem solve. Machine learning algorithms instead allow for computers to train on data inputs and use statistical analysis in order to output values that fall within a specific range. Because of this, machine learning facilitates computers in building models from sample data in order to automate decision-making processes based on data inputs.

Any technology user today has benefitted from machine learning. Facial recognition technology allows social media platforms to help users tag and share photos of friends. Optical character recognition (OCR) technology converts images of text into movable type. Recommendation engines, powered by machine learning, suggest what movies or television shows to watch next based on user preferences. Self-driving cars that rely on machine learning to navigate may soon be available to consumers.

Machine learning is a continuously developing field. Because of this, there are some considerations to keep in mind as you work with machine learning methodologies, or analyze the impact of machine learning processes.

In this tutorial, we’ll look into the common machine learning methods of supervised and unsupervised learning, and common algorithmic approaches in machine learning, including the k-nearest neighbor algorithm, decision tree learning, and deep learning. We’ll explore which programming languages are most used in machine learning, providing you with some of the positive and negative attributes of each. Additionally, we’ll discuss biases that are perpetuated by machine learning algorithms, and consider what can be kept in mind to prevent these biases when building algorithms.

## Machine Learning Methods

In machine learning, tasks are generally classified into broad categories. These categories are based on how learning is received or how feedback on the learning is given to the system developed.

Two of the most widely adopted machine learning methods are **supervised learning** which trains algorithms based on example input and output data that is labeled by humans, and **unsupervised learning** which provides the algorithm with no labeled data in order to allow it to find structure within its input data. Let’s explore these methods in more detail.

### Supervised Learning

In supervised learning, the computer is provided with example inputs that are labeled with their desired outputs. The purpose of this method is for the algorithm to be able to “learn” by comparing its actual output with the “taught” outputs to find errors, and modify the model accordingly. Supervised learning therefore uses patterns to predict label values on additional unlabeled data.

For example, with supervised learning, an algorithm may be fed data with images of sharks labeled as fish and images of oceans labeled as water. By being trained on this data, the supervised learning algorithm should be able to later identify unlabeled shark images as fish and unlabeled ocean images as water.

A common use case of supervised learning is to use historical data to predict statistically likely future events. It may use historical stock market information to anticipate upcoming fluctuations, or be employed to filter out spam emails. In supervised learning, tagged photos of dogs can be used as input data to classify untagged photos of dogs.

### Unsupervised Learning

In unsupervised learning, data is unlabeled, so the learning algorithm is left to find commonalities among its input data. As unlabeled data are more abundant than labeled data, machine learning methods that facilitate unsupervised learning are particularly valuable.

The goal of unsupervised learning may be as straightforward as discovering hidden patterns within a dataset, but it may also have a goal of feature learning, which allows the computational machine to automatically discover the representations that are needed to classify raw data.

Unsupervised learning is commonly used for transactional data. You may have a large dataset of customers and their purchases, but as a human you will likely not be able to make sense of what similar attributes can be drawn from customer profiles and their types of purchases. With this data fed into an unsupervised learning algorithm, it may be determined that women of a certain age range who buy unscented soaps are likely to be pregnant, and therefore a marketing campaign related to pregnancy and baby products can be targeted to this audience in order to increase their number of purchases.

Without being told a “correct” answer, unsupervised learning methods can look at complex data that is more expansive and seemingly unrelated in order to organize it in potentially meaningful ways. Unsupervised learning is often used for anomaly detection including for fraudulent credit card purchases, and recommender systems that recommend what products to buy next. In unsupervised learning, untagged photos of dogs can be used as input data for the algorithm to find likenesses and classify dog photos together.

## Approaches

As a field, machine learning is closely related to computational statistics, so having a background knowledge in statistics is useful for understanding and leveraging machine learning algorithms.

For those who may not have studied statistics, it can be helpful to first define correlation and regression, as they are commonly used techniques for investigating the relationship among quantitative variables. **Correlation** is a measure of association between two variables that are not designated as either dependent or independent. **Regression** at a basic level is used to examine the relationship between one dependent and one independent variable. Because regression statistics can be used to anticipate the dependent variable when the independent variable is known, regression enables prediction capabilities.

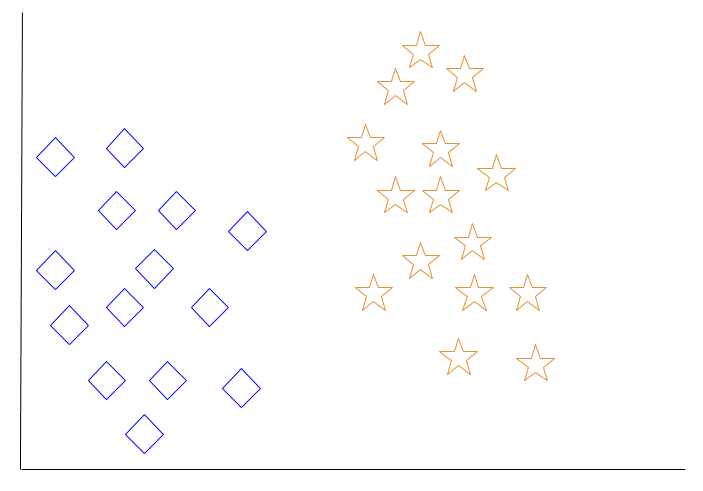
Approaches to machine learning are continuously being developed. For our purposes, we’ll go through a few of the popular approaches that are being used in machine learning at the time of writing.

### k-nearest neighbor

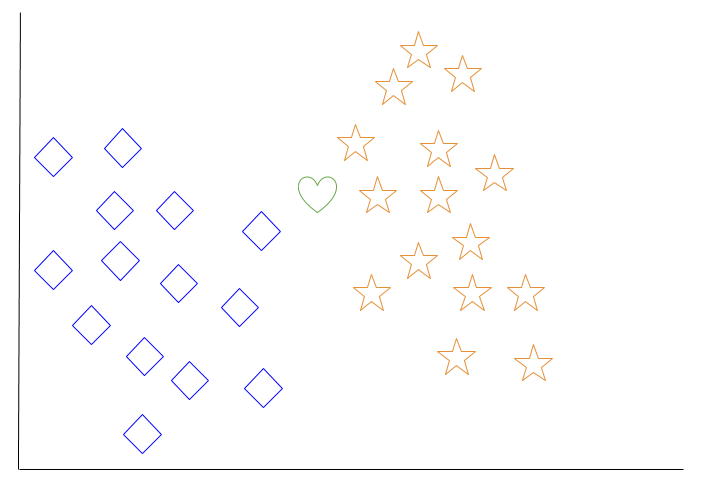
The k-nearest neighbor algorithm is a pattern recognition model that can be used for classification as well as regression. Often abbreviated as k-NN, the **k** in k-nearest neighbor is a positive integer, which is typically small. In either classification or regression, the input will consist of the k closest training examples within a space.

We will focus on k-NN classification. In this method, the output is class membership. This will assign a new object to the class most common among its k nearest neighbors. In the case of k = 1, the object is assigned to the class of the single nearest neighbor.

Let’s look at an example of k-nearest neighbor. In the diagram below, there are blue diamond objects and orange star objects. These belong to two separate classes: the diamond class and the star class.

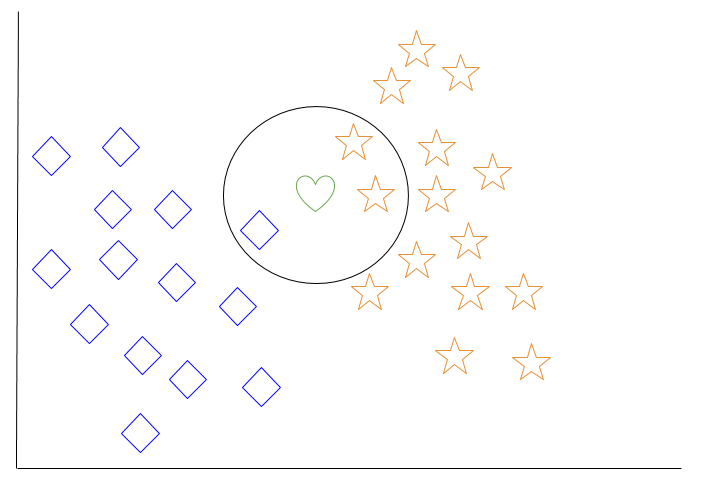


When a new object is added to the space — in this case a green heart — we will want the machine learning algorithm to classify the heart to a certain class.



When we choose k = 3, the algorithm will find the three nearest neighbors of the green heart in order to classify it to either the diamond class or the star class.

In our diagram, the three nearest neighbors of the green heart are one diamond and two stars. Therefore, the algorithm will classify the heart with the star class.



Among the most basic of machine learning algorithms, k-nearest neighbor is considered to be a type of “lazy learning” as generalization beyond the training data does not occur until a query is made to the system.

### Decision Tree Learning

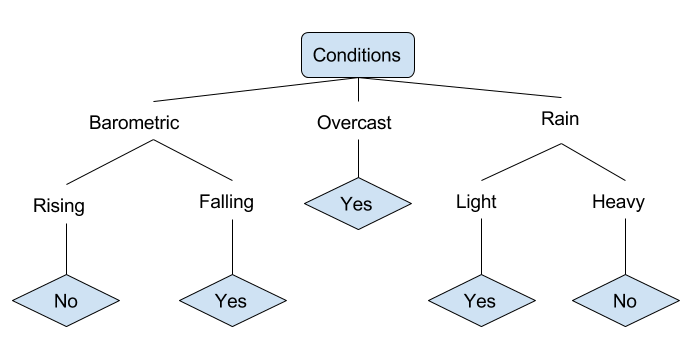
For general use, decision trees are employed to visually represent decisions and show or inform decision making. When working with machine learning and data mining, decision trees are used as a predictive model. These models map observations about data to conclusions about the data’s target value.

The goal of decision tree learning is to create a model that will predict the value of a target based on input variables.

In the predictive model, the data’s attributes that are determined through observation are represented by the branches, while the conclusions about the data’s target value are represented in the leaves.

When “learning” a tree, the source data is divided into subsets based on an attribute value test, which is repeated on each of the derived subsets recursively. Once the subset at a node has the equivalent value as its target value has, the recursion process will be complete.

Let’s look at an example of various conditions that can determine whether or not someone should go fishing. This includes weather conditions as well as barometric pressure conditions.



In the simplified decision tree above, an example is classified by sorting it through the tree to the appropriate leaf node. This then returns the classification associated with the particular leaf, which in this case is either a Yes or a No. The tree classifies a day’s conditions based on whether or not it is suitable for going fishing.

A true classification tree data set would have a lot more features than what is outlined above, but relationships should be straightforward to determine. When working with decision tree learning, several determinations need to be made, including what features to choose, what conditions to use for splitting, and understanding when the decision tree has reached a clear ending.

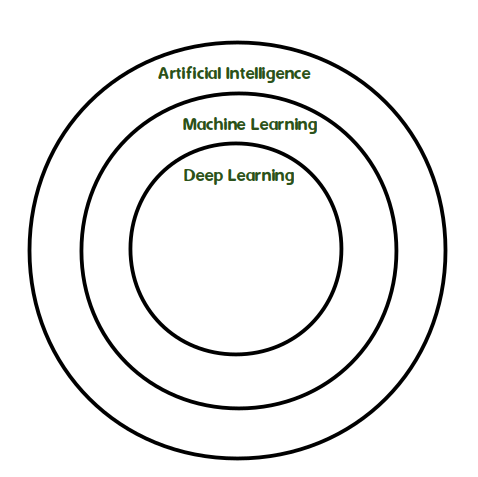
# Introduction to Deep Learning

What is deep learning

Deep learning is a branch of [machine learning](https://www.geeksforgeeks.org/introduction-machine-learning/) which is completely based on [artificial neural networks](https://www.geeksforgeeks.org/tag/neural-network/" \t "_blank), as neural network is going to mimic the human brain so deep learning is also a kind of mimic of human brain. In deep learning, we don’t need to explicitly program everything. The concept of deep learning is not new. It has been around for a couple of years now. It’s on hype nowadays because earlier we did not have that much processing power and a lot of data. As in the last 20 years, the processing power increases exponentially, deep learning and machine learning came in the picture.  
A formal definition of deep learning is- neurons

Deep learning is a particular kind of machine learning that achieves great power and flexibility by learning to represent the world as a nested hierarchy of concepts, with each concept defined in relation to simpler concepts, and more abstract representations computed in terms of less abstract ones.

In human brain approximately 100 billion neurons all together this is a picture of an individual neuron and each neuron is connected through thousand of their neighbours.  
The question here is how do we recreate these neurons in a computer. So, we create an artificial structure called an artificial neural net where we have nodes or neurons. We have some neurons for input value and some for output value and in between, there may be lots of neurons interconnected in the hidden layer.



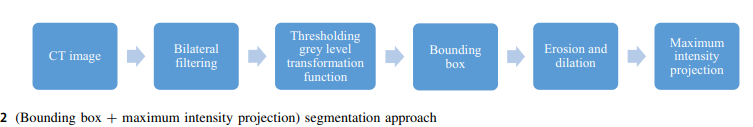
3 Materials and methods

3.1 Dataset

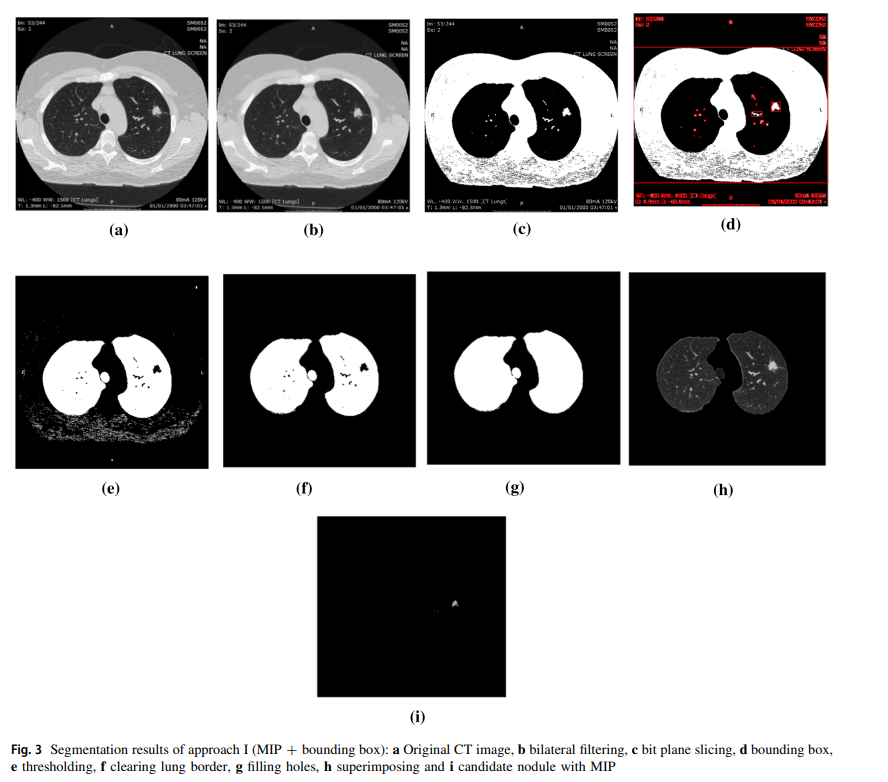
A 14 digital CT consisting of 2991 2D slices which contain 172 nodules (100 solitary nodules and 72 juxtapleural nodules) were downloaded from Cornell University [20] LIDC dataset. Each abnormal image contained a tumor with equivalent diameters of lung nodules ranging from 3 to 30 mm and the total number of normal images is 232 images. For the training phase, (86 abnormal (50 solitary ? 36 juxtapleural), and 116 normal) images are used and for the testing phase, the same number is used. This number of images is quite small to train and test the classifiers, especially the DCNN classifier. We want to compare SVM and DCNN results by using the same dataset so we tackled this point. The in-slice (x, y) resolution is 0.703 9 0.703 mm and the CT slice thickness is 1.25 mm in DICOM format and has 512 9 512 pixels.

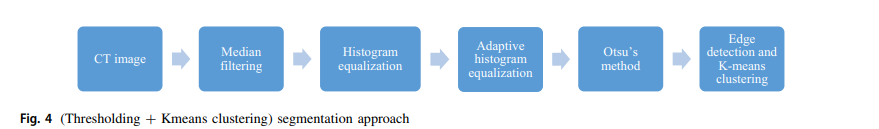
3.2 Segmentation and enhancement for nodule emphasis

Inhomogeneity in the lung region is a very challenging problem as there are similar densities such as veins, arteries, and bronchioles. It is necessary to enhance the quality of the displayed image by rectifying distortions due to media decay or motion artifacts. In this paper, two approaches to segmentation are presented to reach the best segmentation results as follows: 3.2.1 Approach I (bounding box 1 maximum intensity projection) It consists of Bilateral filter [21, 22], Thresholding graylevel transformation function [23], Bounding box [24] 25, erosion and dilation [26] and Maximum intensity projection (MIP) [27] as shown in Fig. 2. The resulted images of approach I are shown in Fig. 3. 3.2.2 Approach II (thresholding 1 K-means clustering) It achieved by using Median filter [28], Histogram equalization [29], Adaptive histogram equalization [30], Otsu’s method [31], Edge detection [32, 33] and K-means clustering [34, 35] as shown in Fig. 4. The resulted images of approach II are shown in Fig. 5. 3.3 Features extraction and selection It is the main step of the Scheme i supervised learning model as the algorithm learns on labeled images; feature extraction; by fitting an answer key that the model can use to evaluate its accuracy on training images. Feature extraction presents all the obtained information in lung CT images to differentiate between the different tissues inside the image. A combination between first-order features (33 shape and texture features (1st set)) and second-order features (80 texture features (2nd set)) are extracted, so the total number of features is 113 (3rd set). Some examples of first-order features are mean [26], standard deviation [26], variance [36], kurtosis [36, 37], entropy [38], nine percentile features [36], 7 invariant moments [36] and Perimeter [39]. Second-order features are the features of gray level co-occurrence matrix (GLCM) [34] such as autocorrelation, contrast, cluster prominence, cluster shade, dissimilarity, and entropy. Because of a large number of features, a filter approach technique is used to rank the strength of these features to achieve high standard accuracy, this technique is called Fisher score ranking [40]. It can differentiate between nodules and non-nodules when calculating the difference



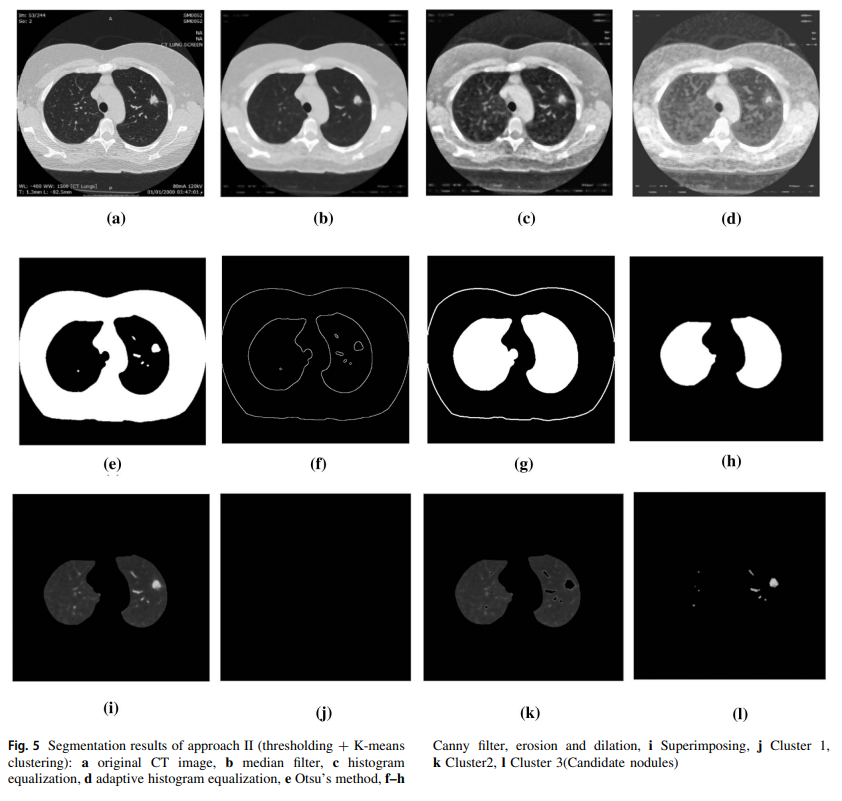
[41] between the mean and the standard deviation of the positive (nodule) and negative (non-nodule) relative to a definite feature. Equation (1) clarifies the Fisher score, in which Ri is the feature rank i, representing the proportion of the substitution of the feature mean i values in the nodule examples (p) and the non-nodule examples (n), and the sum of the standard deviations. The bigger the Ri, the bigger the difference between the values of (p) and (n) examples relative to feature i The highest 5, 10, and 15 ranks of each feature set are obtained to minimize the classification error and decrease the computation time. Ri ¼ li;p li;n ri;pþri;n ð1Þ where li;p and li;n are the means of the positive (abnormal) and negative (normal), ri;p and ri;n are the standard deviations of the positive (abnormal) and negative (normal). 3.4 Fold cross-validation Cross-validation (CV) is a resampling procedure used to evaluate machine learning models on a limited data sample. The procedure has a single parameter called k which refers to the number of groups that a given data sample is to be split into. As such, the procedure is often called k-fold cross-validation. When a specific value for k is chosen, it may be used in place of k in the reference to the model [35, 42]. In this research, k = 10 is used becoming tenfold cross-validation. 3.5 Random over sampling As the abnormal images and normal images are not equal (imbalanced), therefore the classifiers tend to provide a severely imbalanced degree of accuracy. Random oversampling (ROS) is proposed to balance the data by increasing the size of the minority class (abnormal) when replicating randomly a set of samples of it as shown in Eq. (2) [43, 44]. j j S ¼ j j Smin þ Smaj þ j j Emin ð2Þ where Emin represents the set of replicated sample, Smin represents the minority class (abnormal), Smaj represents the majority class (normal).

****

****

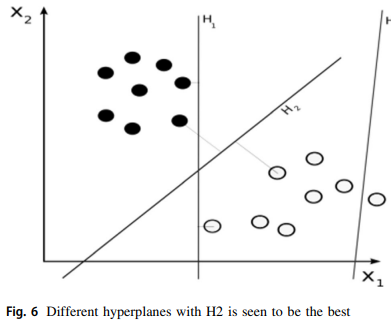
**4 Classification**

In the proposed scheme I, the classifier used the trained dataset to classify the abnormality, and the features of the bigger R are fed into the classifier. On the other hand, in scheme II a deep learning classification is done. The total number of abnormal images is 172 images containing (100 solitary nodules and 72 juxtapleural nodules) and the total number of normal images is 232 images. The training phase has 202 images (116 normal and 86 abnormal (50 solitary and 36 juxtapleural)) and the same number for testing, therefore the total number of images was 404. The dataset was split 50–50 to minimize the computation time.

****

**4.1 Scheme I (supervised learning model)**

In scheme I, Support Vector Machine (SVM) classifier is used. SVM is considered as a linear classifier related to supervised learning algorithms which are applied for classification problems. In a high-dimensional feature space, SVM constructed a hyperplane that separates the data points (samples) of the two classes. But many hyperplanes can be found for this separation (Fig. 6), so it’s the SVM’s second function to find the optimum hyperplane (maximum margin hyperplane). SVM uses a linear functions hypothesis space to achieve the classification easier These kernel functions produce nonlinear separators to map the input space attributes to the feature space [45, 46]. Three basic kernels are used Linear, Quadratic, and Radial Basis Functions as shown in Eqs. (3), (4), and (5). Linear: K xixj ¼ xi; xj where x is a sample data ð3Þ

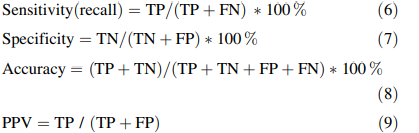
****

Quadratic : Kðxi; xjÞ ¼ cxi; xj þ r d where c [ 0 ð4Þ Radial Basis : Kðxi; xjÞ ¼ e c xixjj 2 where c [ 0 ð5Þ where c, r, and d are kernel parameters. There is no theory about deciding which kernel is the best, so in our system, we tried using the three mentioned kernel functions.

4.2 Scheme II (supervised learning model)

Convolutional neural network (CNN) for deep learning classification is used. It is a successful tool for deep learning classification [47, 48] and developed to suit image recognition as it is a multilayer neural network, which consists of single or more convolution layers followed by one or more fully connected layers. Our convolutional neural network architecture is consisting of a convolution layer, max pooling layer, fully connected layer, and softmax layer as shown in Fig. 7. In the input image layer, we specify the input image size as 512 9 512 and the channel size is 1. The filter size in the convolution layer is (5,5) and number of filters are 20. The max pooling layer returns the maximum values of rectangular regions of inputs, in our study the size of the rectangular region is (2,2). The fully connected layer combines all of the features (local information) that were learned by all previous layers across the image to classify these images. As the output parameter in this layer is equal to the number of classes in the target data, the output size will be 2 classes (solitary and juxtapleural nodules).

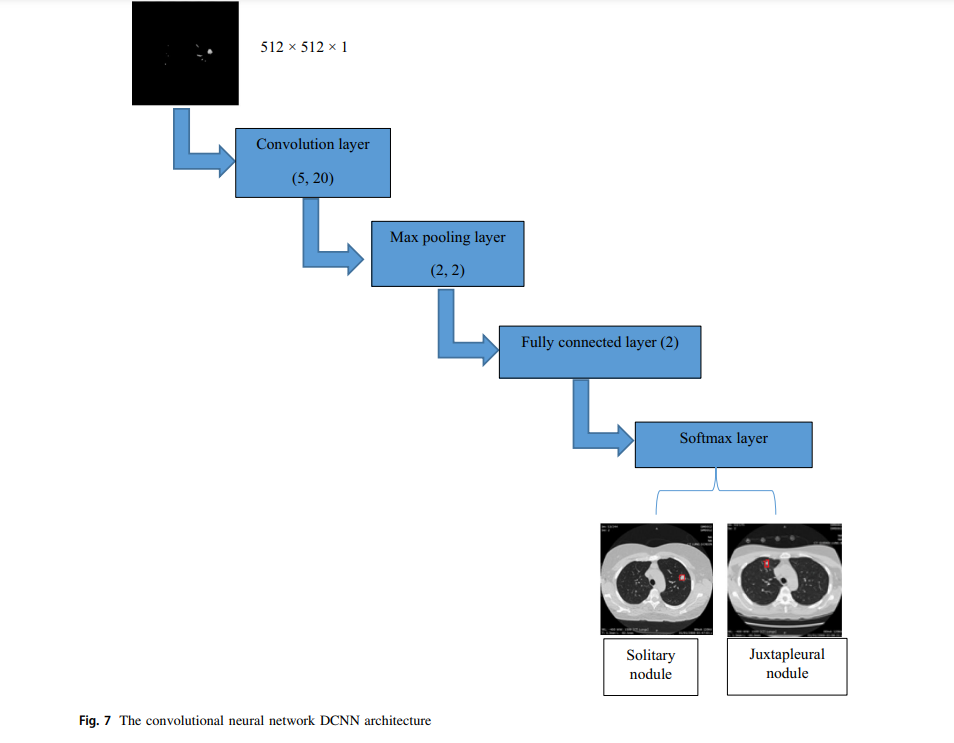
4.3 Performance measures

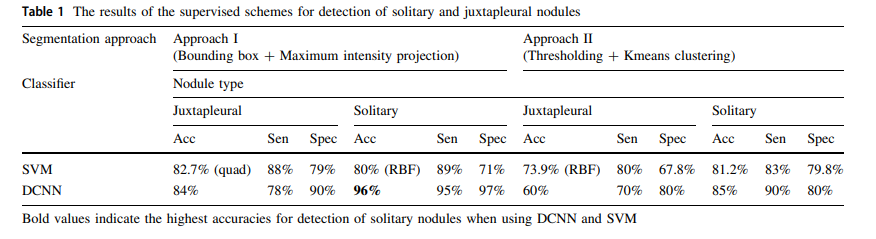
The classification binary test may have an error if the classifier misses to distinguish the abnormality or distinguish an abnormality that is not present. This error can be characterized by True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN) [49]. Where A true positive is an outcome where the model correctly predicts the positive class. Similarly, a true negative is an outcome where the model correctly predicts the negative class. A false positive is an outcome where the model incorrectly predicts the positive class. And a false negative is an outcome where the model incorrectly predicts the negative class. According to these four terms sensitivity (recall), specificity, accuracy, and precision or positive predictive value (PPV) the classifier performance is evaluated as shown in Eqs. (6), (7), (8), and (9), respectively. 

Receiver operating characteristics (ROC) curve is used to evaluate the ANN classifier [50] as it is a two-dimensional graph in which the y-axis represents the TPR and the x-axis represents the FPR. The Area under the ROC curve (AUC) metric is used to estimate the area under this curve. The score of AUC is always confined between zero and one, and there is no factual classifier that has an AUC lower than 0.5.

5 Results and discussions

In this section, the best sensitivity, specificity, and accuracy of our learning schemes are presented in Table 2 after applying CV and ROS. Table 2 shows the results of the SVM classifier of the scheme I with feature set (combination between 1st and 2nd order features) and the result of DCNN classifier of the scheme II which are combined in the two approaches of segmentation (MIP ? Bounding box and Thresholding ? K-means clustering). Regarding approach, I of segmentation (Bounding box ? Maximum Intensity Projection) our proposed CAD system before applying the CV and ROS achieved classification for:

****

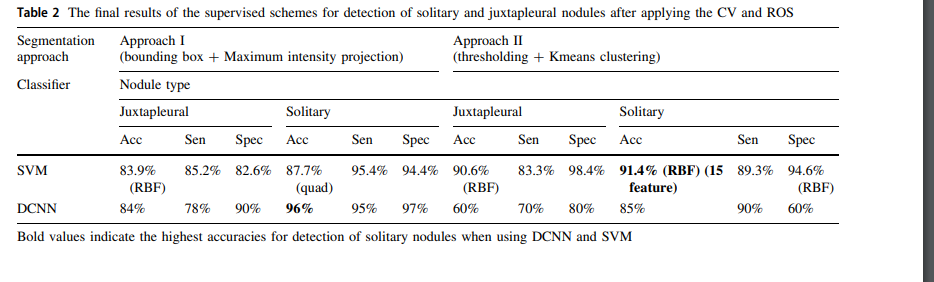
****

a. The solitary nodules:

SVM classifier achieved 80%, 89%, and 71% as an acc., sen., and spec., respectively, when using the radial basis function and extracting the best 5 features of the 1st set, Table 1.

b. The juxtapleural nodules:

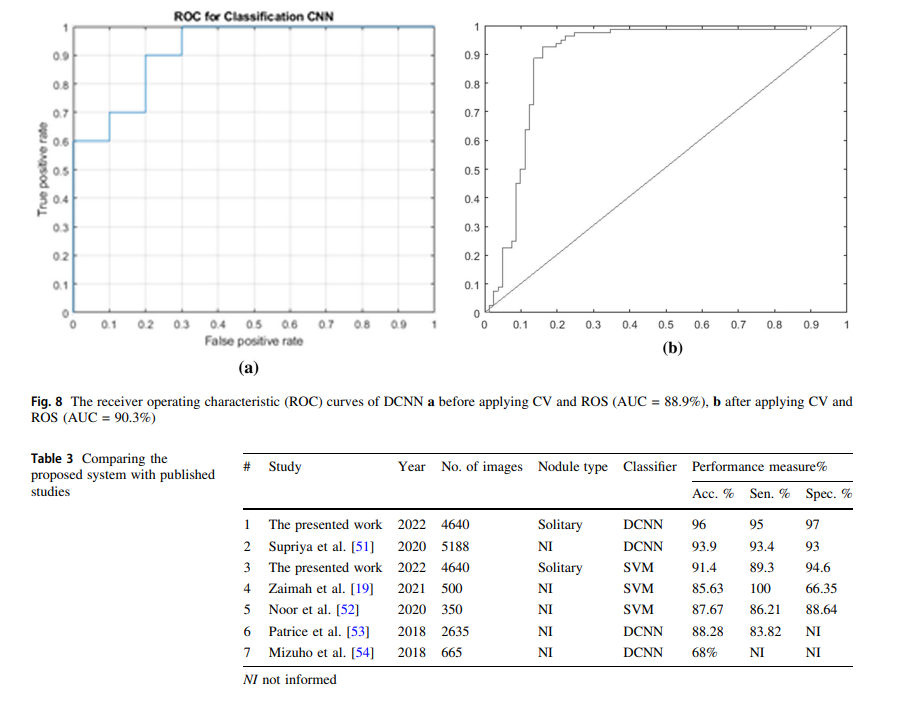
SVM classifier achieved 82.7%, 88%, and 79% as an acc., sen., and spec., respectively, when using quadratic function and extracting the best 15 features of the 1st set, Table 1. DCNN results are overfitted before applying CV and ROS. Regarding approach, II of segmentation (Thresholding ? Kmeans clustering) the proposed CAD system before applying the CV and ROS achieved classification for: c. The solitary nodules: SVM classifier achieved 81.2%, 83%, and 79.8% as an acc., sen., and spec., respectively, when using a quadratic function with the best 10 features of the 3rd set, Table 1. d. The juxtapleural nodules: SVM classifier achieved 73.9%, 80%, and 67.8% as an acc., sen., and spec., respectively, when using the RBF function with the best 5 features of the 3rd set, Table 1. Regarding approach, I of segmentation (Bounding box ? Maximum Intensity Projection) our proposed CAD system after applying the CV and ROS achieved classification for: e. The solitary nodules: SVM classifier achieved 87.7%, 95.4%, and 94.4% as an acc., sen., and spec., respectively, when using radial basis function and extracting the best 15 features of the 3rd set, Table 2. DCNN achieved the highest accuracy, sensitivity, and specificity 96 %, 95 %, and 97 %, respectively, to detect the solitary nodule, Table 2. f. The juxtapleural nodules: SVM classifier achieved 83.9%, 85.2%, and 82.6% as an acc., Sen., and spec., respectively, when using the radial basis function with the best 10 features of the 2nd set, Table 2.

****

DCNN achieved accuracy, sensitivity, and specificity of 84%, 78%, and 90%, respectively, Table 2. Regarding approach, II of segmentation (Thresholding? Kmeans clustering) the proposed CAD system after applying the CV and ROS achieved classification for: g. The solitary nodules: SVM classifier achieved 91.4%, 89.3%, and 94.6% as accuracy, sensitivity, and specificity, respectively, when using RBF function with the best 15 features of the 3rd set, Table 2. DCNN achieved the highest accuracy, sensitivity, and specificity 85 %, 90 %, and 80 %, respectively, to detect the solitary nodule, Table 2. h. The juxtapleural nodules: SVM classifier achieved 90.6%, 83.3%, and 98.4% as an acc., Sen., and spec., respectively, when using the RBF function with the best 15 features of the 1st set, Table 2. DCNN achieves accuracy, sensitivity, and specificity of 60%, 70%, and 80%, respectively, Table 2. The highest Area Under the Curve (AUC) of the receiver operating characteristic (ROC) curve for the CNN classifier is 90.3%, which is shown in Fig. 7b. The results of sensitivity, specificity, and accuracy of the Scheme I (SVM) and scheme II (DCNN) are shown in the results section; Tables 1 and 2 for distinguishing between solitary and juxtapleural nodules. At first, the performance measures of the classifiers are estimated as follows (Table 1). For the SVM, it achieved 82.6%, 87.9%, and 78.9% as accuracy, sensitivity, and specificity, respectively, to classify the juxtapleural nodules when applying approach I of segmentation (Bounding Box ? Maximum intensity projection) before applying the CV and ROS. There were some challenges, but we tackled with it to improve the overall classification performance by upgrading the overall performance measures as shown in Table 2

and also increasing the area under the curve of ROC curves. At first, the proposed system is applied to a total number of abnormal images which is 172 images containing (100 solitary nodules and 72 juxtapleural nodules) and a total number of normal images which is 232 images. For the training phase (86 abnormal (50 solitary ? 36 juxtapleural) images, 116 normal images) and for the testing phase, the same number is used (50–50). This number of images achieved results which are shown in Table 1. To enhance these results by increasing the images number, tenfold cross-validation is used. The total number is divided by 10. For example, the abnormal images are 172/10 & 17 (for testing) and 172–17 & 155 (for training), and then this step is repeated 10 times. Therefore, the abnormal images increases to 1550 images for the training phase and 170 for testing (the total number of abnormal images became (1550 ? 170 = 1720)). Also, tenfold cross-validation was applied to the normal images in the same manner. It increases to 2320 images for the training phase and testing phase, so the limited data sample becomes 1720 abnormal images and 2320 normal images. At first, for the DCNN only 20, 30, and 40 images for training and the same for testing to decrease the time of computation as it consumes averaging time of 5 min only, but the results are overfitted, so tenfold CV was applied to increase the number of images to 2320 normal images and 1720 abnormal images as mentioned previously. Hence, the computation time increases to around 20 min based on our hardware and software specifications. DCNN achieves good results of accuracy, sensitivity, and specificity 96%, 95%, and 97%, respectively, when classifying solitary nodules, so it does not need an additional samples of training data as the results of CNN model fulfilled the requirements. Most models that use imageNet trained for five to six days such as network which is called AlexNet which has 15 million training images [39]. There is another challenge we tackled with, the imbalanced data, it reduced the overall classification performance as the normal and abnormal images are not equal and hence the images of training and testing phases are not equal. After applying CV, the normal images became 2320 and the abnormal images became 1720 which are not equal. After applying ROS the abnormal images (minority class) increased to 2320 (the same number as the majority class). Therefore, the total number of images increased from 404 (172 ? 232) images to 4640 (2320 ? 2320) images, so the classifiers provide optimum results as shown in Table 2 and the ROC curve result will be near the best.

• Before applying CV and ROS: Table 1 SVM classifier achieved 82.7%, 88%, 79%, and 74.5% as the highest accuracy, sensitivity (recall), specificity, and PPV, respectively, in the diagnosis of juxtapleural nodule when using BB ? MIP segmentation. This result is achieved with FP = 50, TN = 188, TP = 146, and FN = 20. Also, the SVM classifier achieved 81.2%, 83%, 79.8%, and 76% as the highest accuracy, sensitivity (recall), specificity, and PPV, respectively, in the diagnosis of solitary nodule when using k-means clustering segmentation. This result is achieved with FP = 46, TN = 182, TP = 146, and FN = 30. DCNN results are overfitted before applying CV and ROS. The AUC of the ROC curve was 88.9% as shown in Fig. 8a. • After applying CV and ROS: Table 2 SVM classifier achieved 90.6%, 83.3%, 98.4%, and 98.3% as the highest accuracy, sensitivity, specificity, and PPV, respectively, in the diagnosis of juxtapleural nodule when using k-means clustering segmentation. This result is achieved with FP = 35, TN = 2200, TP = 2005, and FN = 400. Also, the SVM classifier achieved 91.4%, 89.3%, 94.6%, and 96.2% as the highest accuracy, sensitivity, specificity, and PPV, respectively, in the diagnosis of solitary nodule when using k-means clustering segmentation. This result is achieved with FP = 100, TN = 1740, TP = 2500, and FN = 300. The DCNN achieved 96%, 95%, and 97% as accuracy, sensitivity, and specificity, respectively, in the diagnosis of the solitary nodule. The AUC of the ROC curve was increased to 90.3% as shown in Fig. 8b. Based on literature research shown in Table 3, sensitivity, accuracy, nodule type, classifier and database are observed. The systems compared to our system are Supriya et al. [51], Noor et al. [52], Patrice et al. [53], and Mizuho et al. [54]. The first used a CNN classifier with 5188 images which achieved accuracy and sensitivity of 93.9% and 93.4%, respectively. The second used an SVM classifier using 350 images from the LIDC dataset. She achieved an accuracy of 87.67% and a sensitivity of 86.21%. However, validation of the systems was not tested with all nodule types; it showed promising results. The third tested his method with 2635 nodules and achieved 88.28% accuracy and a sensitivity of 83.82% when using Deep Convolutional Neural Network (DCNN) for nodule detection knowing that the nodule type was not informed.

****

The fourth used CNN classifier with 665 nodules and obtained an accuracy of 68%. The presented system showed the best result when using DCNN achieving 96% accuracy for detection of solitary nodules when using CNN for deep learning classification with a sensitivity of 95% when compared to Supriya et al. [51]. We registered the highest results with SVM compared to Noor et al. [52] with accuracy and sensitivity of 91.4% and 89.3%, respectively

**SYSTEM STUDY**

**FEASIBILITY STUDY**

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

**Three key considerations involved in the feasibility analysis are,**

* **ECONOMICAL FEASIBILITY**
* **TECHNICAL FEASIBILITY**
* **SOCIAL FEASIBILITY**

**ECONOMICAL FEASIBILITY**

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

### TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

**SOCIAL FEASIBILITY**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

**SYSTEM TEST**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

### TYPES OF TESTS

**Unit testing**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**Integration testing**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

**Functional test**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

**System Test**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

**White Box Testing**

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

**Black Box Testing**

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

**Unit Testing**

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

**Test strategy and approach**

Field testing will be performed manually and functional tests will be written in detail.

**Test objectives**

* All field entries must work properly.
* Pages must be activated from the identified link.
* The entry screen, messages and responses must not be delayed.

**Features to be tested**

* Verify that the entries are of the correct format
* No duplicate entries should be allowed
* All links should take the user to the correct page.

# Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

**Acceptance Testing**

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

**INPUT AND OUTPUT DESIGN**

**INPUT DESIGN**

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

* What data should be given as input?
* How the data should be arranged or coded?
* The dialog to guide the operating personnel in providing input.
* Methods for preparing input validations and steps to follow when error occur.

**OBJECTIVES**

1.Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

3.When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow

**OUTPUT DESIGN**

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system’s relationship to help user decision-making.

1. 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 people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

2.Select methods for presenting information.

3.Create document, report, or other formats that contain information produced by the system.

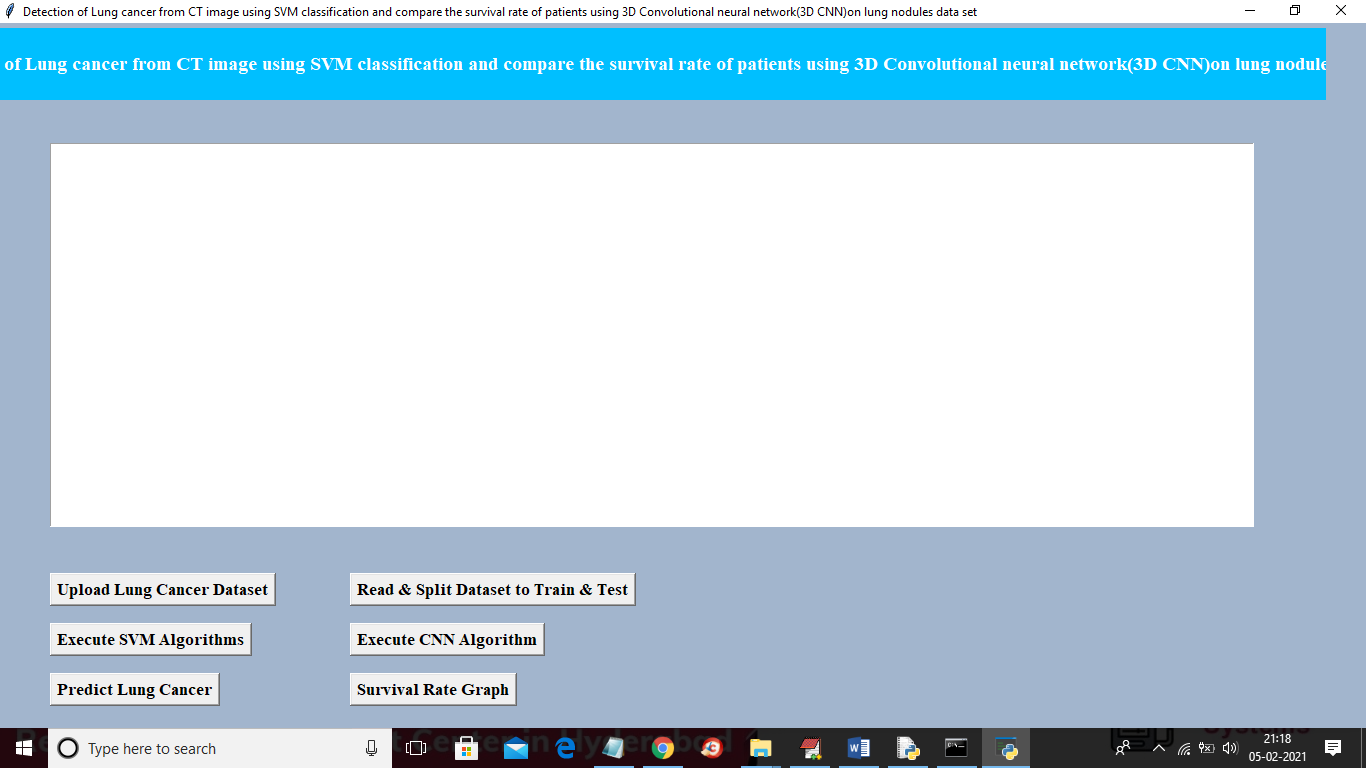
The output form of an information system should accomplish one or more of the following objectives.

* Convey information about past activities, current status or projections of the
* Future.
* Signal important events, opportunities, problems, or warnings.
* Trigger an action.
* Confirm an action.

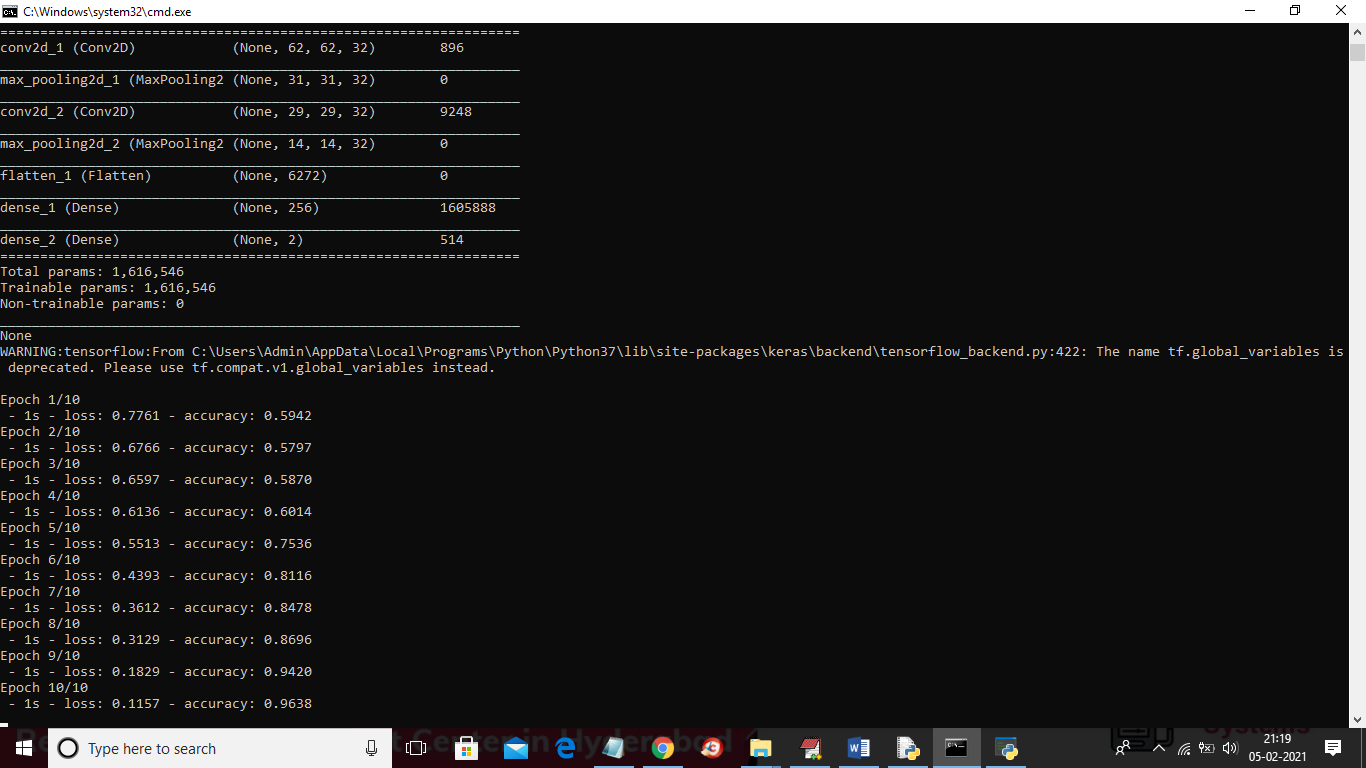
OUTPUT SCREENS

SCREEN SHOTS

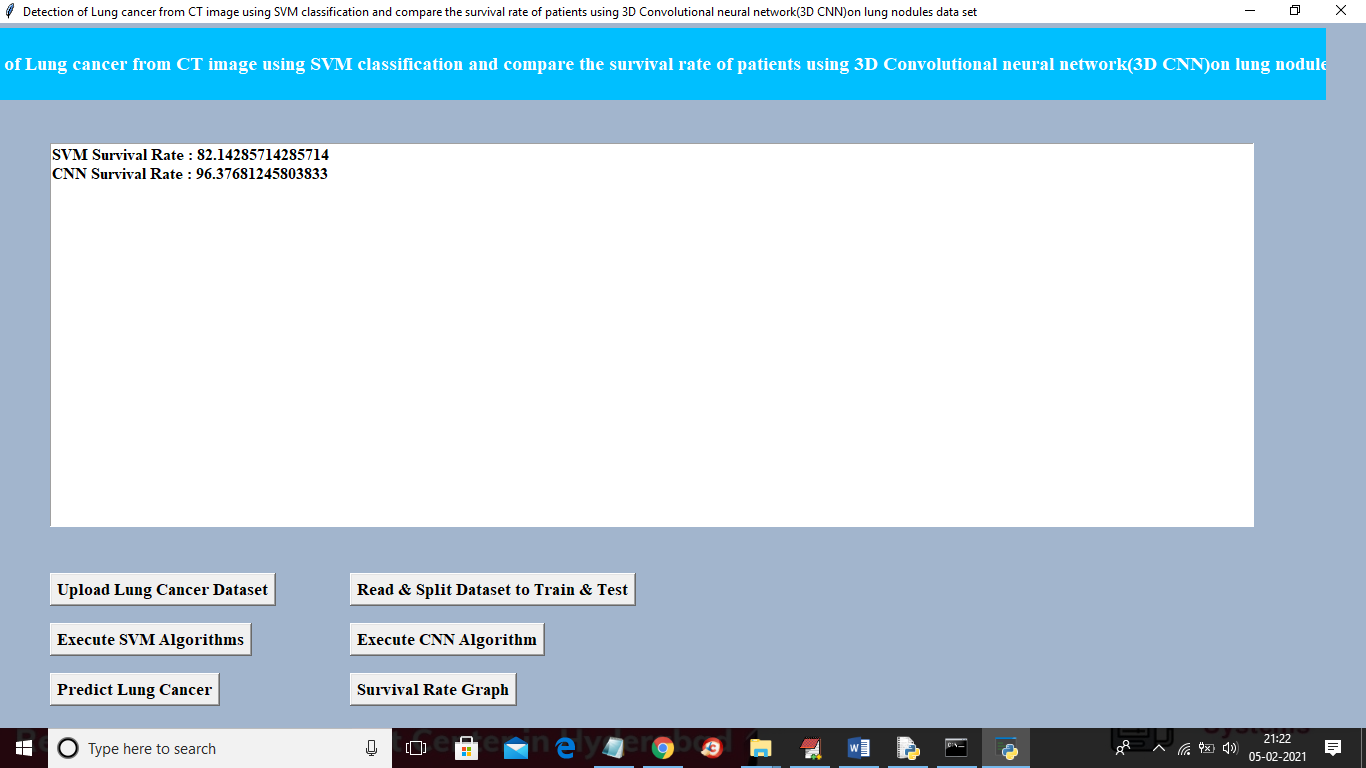
To run project double click on ‘run.bat’ file from ‘Title3\_SVM\_CNN’ folder to get below screen



Similar to first two projects here also you upload ‘Dataset’ folder and then click on “read & split” button and then execute SVM and CNN and then predict cancer and go for survival rate graph. For CNN results you can refer to black console below



In above screen you can see for CNN we use multiple filters to filter dataset for better prediction result and in above screen in first layer CNN use 62 X 62 image size with 32 filters and in second layer for 31 X 31 image size also it uses 32 filters and for each filter we will have best image features and prediction accuracy will be better. In above screen to run CNN I used 10 epoch/iteration and for each increase iteration accuracy get better and better and for last epoch we got 0.96% accuracy and below is the final accuracy result for both SVM and CNN



In above screen SVM survival rate is 82% and CNN survival rate is 96% and similarly you can go for predict button and graph button.

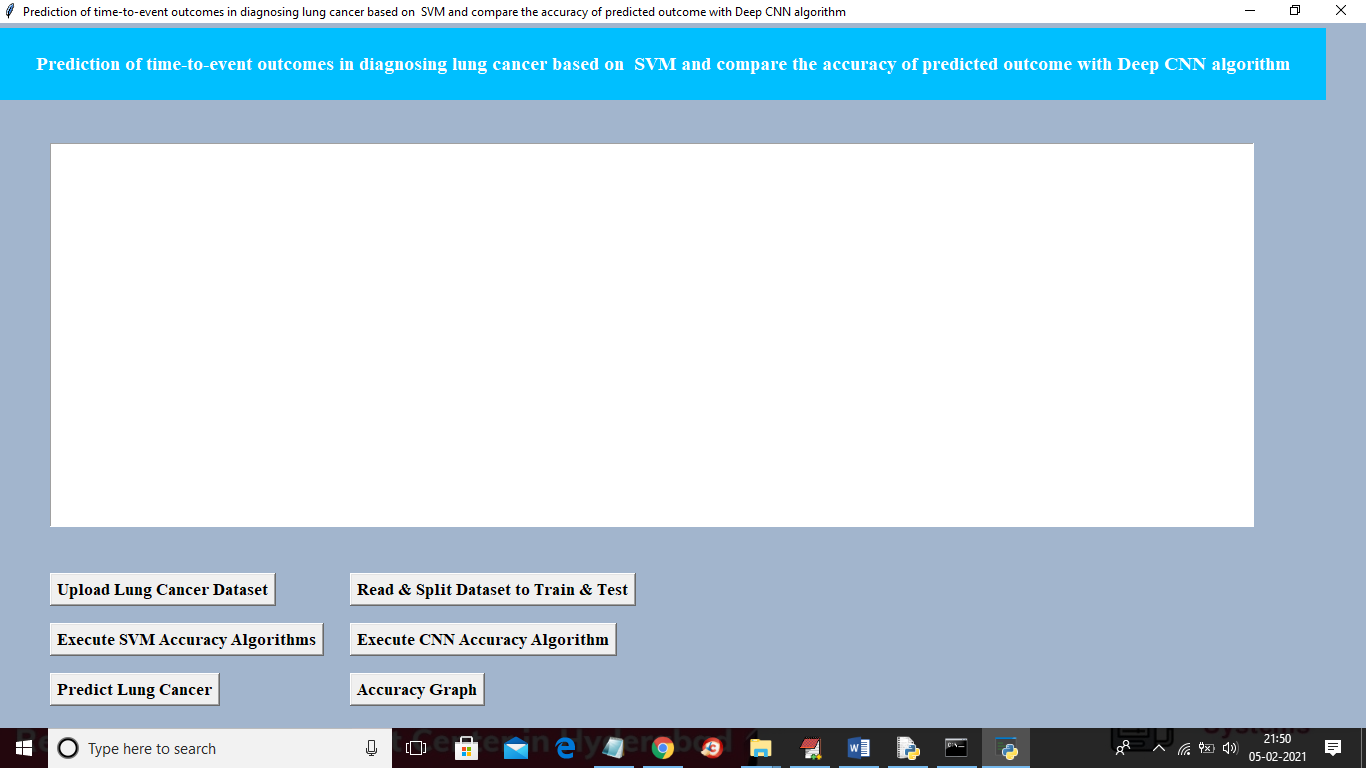
Title 4

**Prediction of time-to-event outcomes in diagnosing lung cancer based on SVM and compare the accuracy of predicted outcome with Deep CNN algorithm**

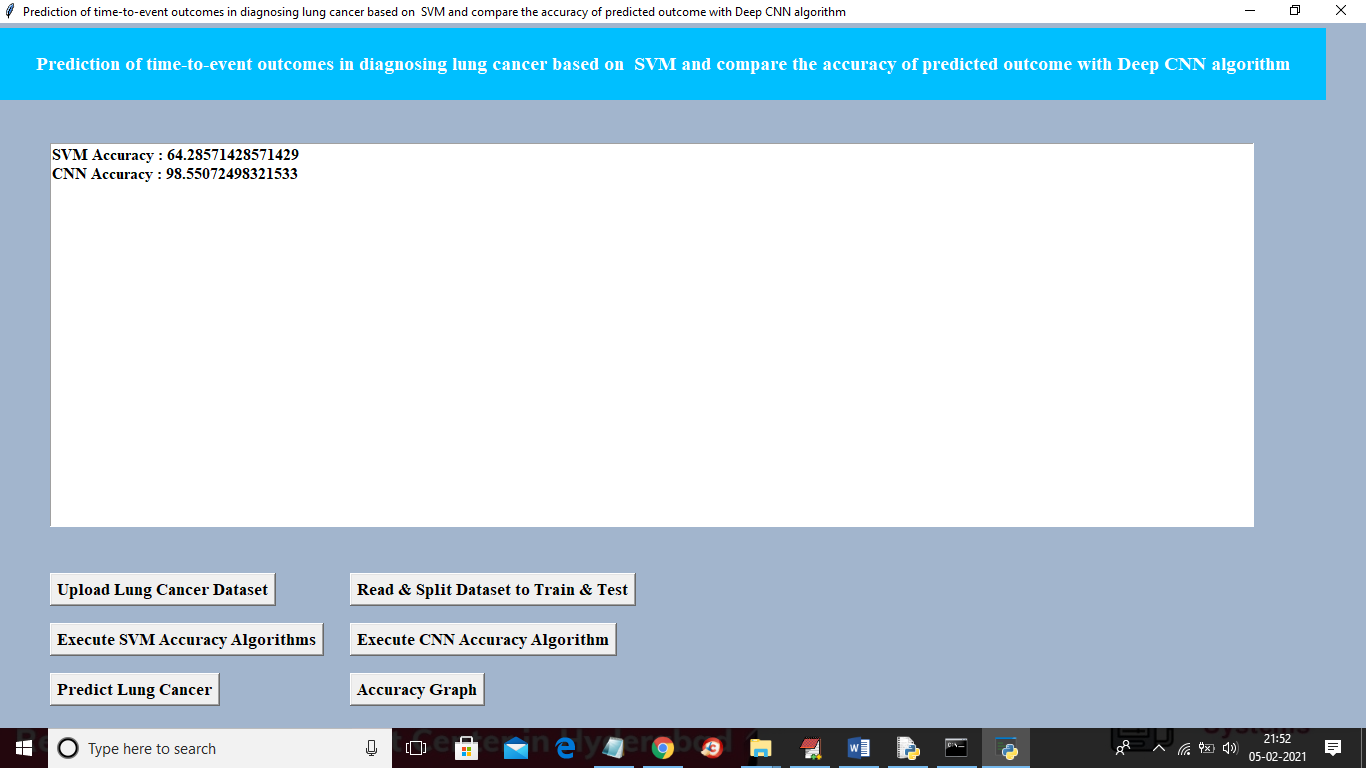
In this project we are training SVM and CNN with same LUNG dataset and then calculating and comparing accuracy of both algorithms

SCREEN SHOT

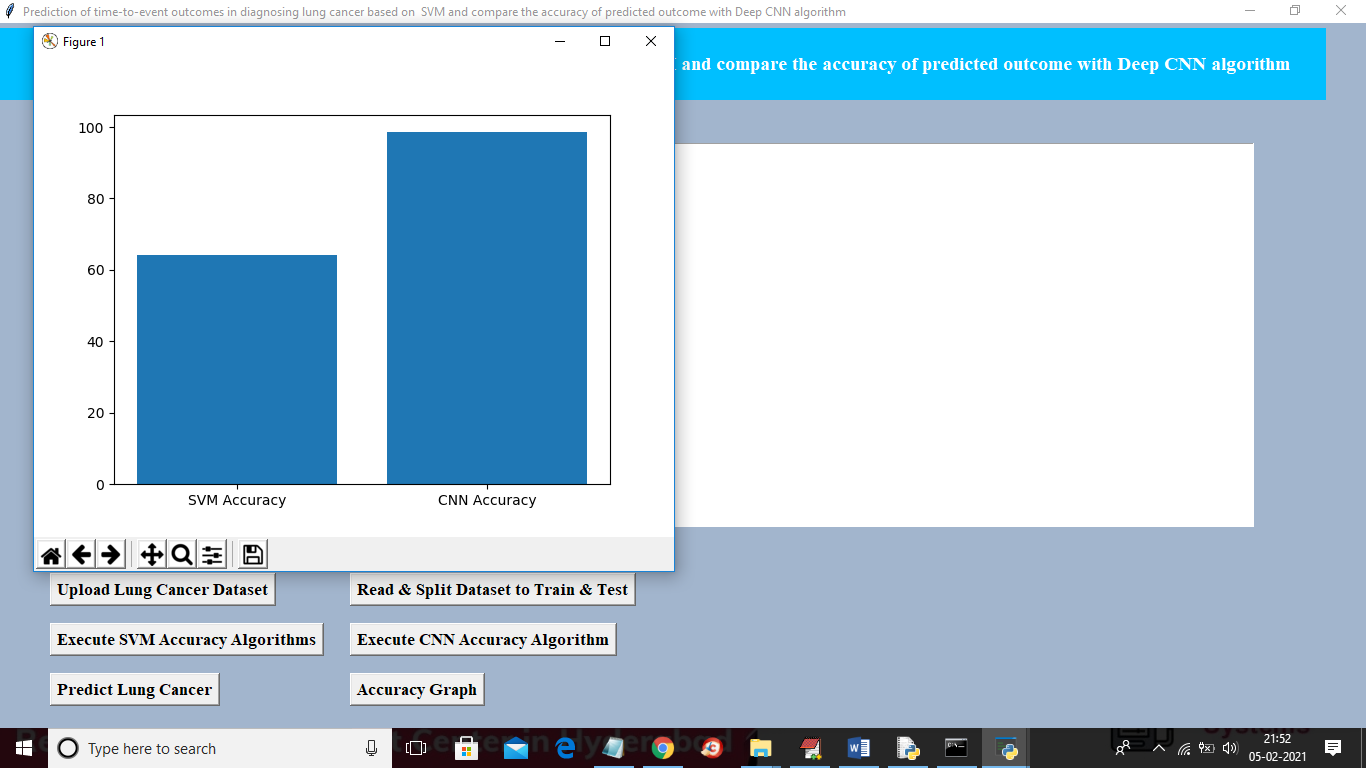
To run this project double click on ‘run.bat’ file from ‘Title4\_SVM\_CNN\_Accuracy’ folder to get below screen



In above screen similar to first two projects upload dataset and then click on ‘read and split dataset’ button and then execute SVM with accuracy and CNN with accuracy and then you can go for predict lung cancer and accuracy graph



In above screen SVM accuracy is 64% and CNN accuracy is 98% and below is the comparison graph for title 4



TEST CASES

**USER REQUIREMENTS:**

1. Home

Home:

|  |  |
| --- | --- |
| Use case ID | TWITTER SENTIMENT ANALYSIS BASED ON ORDINAL REGRESSION |
| Use case Name | Home button |
| Description | Display home page of application |
| Primary actor | User |
| Precondition | User must open application |
| Post condition | Display the Home Page of an application |
| Frequency of Use case | Many times |
| Alternative use case | N/A |
| Use case Diagrams |  |
| Attachments | N/A |

**FUNCTIONAL REQUIREMENTS**

In [software engineering](https://en.wikipedia.org/wiki/Software_engineering" \o "Software engineering) and [systems engineering](https://en.wikipedia.org/wiki/Systems_engineering" \o "Systems engineering), a **functional requirement** defines a function of a [system](https://en.wikipedia.org/wiki/System" \o "System) or its component, where a function is described as a specification of behavior between outputs and inputs.[[1]](https://en.wikipedia.org/wiki/Functional_requirement" \l "cite_note-FultonAirborne17-1)

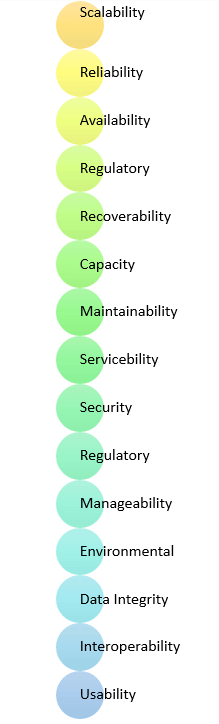
Functional requirements may involve calculations, technical details, data manipulation and processing, and other specific functionality that define what a system is supposed to accomplish.[[2]](https://en.wikipedia.org/wiki/Functional_requirement" \l "cite_note-2) Behavioral requirements describe all the cases where the system uses the functional requirements, these are captured in [use cases](https://en.wikipedia.org/wiki/Use_case" \o "Use case). Functional requirements are supported by [non-functional requirements](https://en.wikipedia.org/wiki/Non-functional_requirement" \o "Non-functional requirement) (also known as "quality requirements"), which impose constraints on the design or implementation (such as performance requirements, security, or reliability). Generally, functional requirements are expressed in the form "system must do <requirement>," while non-functional requirements take the form "system shall be <requirement>." The plan for implementing functional requirements is detailed in the system design, whereas *non-functional* requirements are detailed in the system architecture.[[4]](https://en.wikipedia.org/wiki/Functional_requirement" \l "cite_note-AdamsNon15-4)[[5]](https://en.wikipedia.org/wiki/Functional_requirement" \l "cite_note-J%C3%B6nssonImpact06-5)

As defined in [requirements engineering](https://en.wikipedia.org/wiki/Requirements_analysis" \o "Requirements analysis), functional requirements specify particular results of a system. This should be contrasted with non-functional requirements, which specify overall characteristics such as cost and [reliability](https://en.wikipedia.org/wiki/Reliability_engineering" \o "Reliability engineering). Functional requirements drive the application architecture of a system, while non-functional requirements drive the technical architecture of a system.[[4]](https://en.wikipedia.org/wiki/Functional_requirement" \l "cite_note-AdamsNon15-4)

In some cases a requirements analyst generates use cases after gathering and validating a set of functional requirements. The hierarchy of functional requirements collection and change, broadly speaking, is: user/[stakeholder](https://en.wikipedia.org/wiki/Project_stakeholder" \o "Project stakeholder) request → analyze → use case → incorporate. Stakeholders make a request; systems engineers attempt to discuss, observe, and understand the aspects of the requirement; use cases, entity relationship diagrams, and other models are built to validate the requirement; and, if documented and approved, the requirement is implemented/incorporated.[[6]](https://en.wikipedia.org/wiki/Functional_requirement" \l "cite_note-MITRESys14-6) Each use case illustrates behavioral scenarios through one or more functional requirements. Often, though, an analyst will begin by eliciting a set of use cases, from which the analyst can derive the functional requirements that must be implemented to allow a user to perform each use case.

**NON-FUNCTIONAL REQUIREMENT** (NFR) specifies the quality attribute of a software system. They judge the software system based on Responsiveness, Usability, Security, Portability and other non-functional standards that are critical to the success of the software system. Example of nonfunctional requirement, *“how fast does the website load?”* Failing to meet non-functional requirements can result in systems that fail to satisfy user needs.

Non-functional Requirements allows you to impose constraints or restrictions on the design of the system across the various agile backlogs. Example, the site should load in 3 seconds when the number of simultaneous users are > 10000. Description of non-functional requirements is just as critical as a functional requirement.



* Usability requirement
* Serviceability requirement
* Manageability requirement
* Recoverability requirement
* Security requirement
* Data Integrity requirement
* Capacity requirement
* Availability requirement
* Scalability requirement
* Interoperability requirement
* Reliability requirement
* Maintainability requirement
* Regulatory requirement
* Environmental requirement

**Advantages of Non-Functional Requirement**

Benefits/pros of Non-functional testing are:

* The nonfunctional requirements ensure the software system follow legal and compliance rules.
* They ensure the reliability, availability, and performance of the software system
* They ensure good user experience and ease of operating the software.
* They help in formulating security policy of the software system.

**Disadvantages of Non-functional requirement**

Cons/drawbacks of Non-function requirement are:

* None functional requirement may affect the various high-level software subsystem
* They require special consideration during the software architecture/high-level design phase which increases costs.
* Their implementation does not usually map to the specific software sub-system,
* It is tough to modify non-functional once you pass the architecture phase.

**KEY LEARNING**

* A non-functional requirement defines the performance attribute of a software system.
* Types of Non-functional requirement are Scalability Capacity, Availability, Reliability, Recoverability, Data Integrity, etc.
* Example of Non Functional Requirement is Employees never allowed to update their salary information. Such attempt should be reported to the security administrator.
* Functional Requirement is a verb while Non-Functional Requirement is an attribute
* The advantage of Non-functional requirement is that it helps you to ensure good user experience and ease of operating the software
* The biggest disadvantage of Non-functional requirement is that it may affect the various high-level software subsystems.

6 Conclusions In this research, a Computer Aided Diagnosis (CADx) system of candidate lung nodules either solitary or juxtapleural nodules is proposed regarding its location. This CAD system acts as a second opinion for the radiologist to help in the early diagnosis of lung cancer. A database consisting of 14 digital CT consisting of 2991 2D slices containing 172 nodules with equivalent diameters of lung nodules (solitary and juxtapleural) ranging from 3 to 30 mm is used. For classification, SVM is used in scheme I, and Convolutional Neural Network is used in scheme II. A Segmentation and enhancement approaches (bounding box ? MIP) and (Thresholding ? K-means clustering) are proposed. A combination of two sets of features is extracted. Fisher score ranking is used as a feature selection method. Selected features are input to SVM classifiers in the scheme I. K-fold cross-validation is used as a resampling procedure by which our supervised learning algorithm has higher average accuracy, we used Fig. 8 The receiver operating characteristic (ROC) curves of DCNN a before applying CV and ROS (AUC = 88.9%), b after applying CV and ROS (AUC = 90.3%) Table 3 Comparing the proposed system with published studies # Study Year No. of images Nodule type Classifier Performance measure% Acc. % Sen. % Spec. % 1 The presented work 2022 4640 Solitary DCNN 96 95 97 2 Supriya et al. [51] 2020 5188 NI DCNN 93.9 93.4 93 3 The presented work 2022 4640 Solitary SVM 91.4 89.3 94.6 4 Zaimah et al. [19] 2021 500 NI SVM 85.63 100 66.35 5 Noor et al. [52] 2020 350 NI SVM 87.67 86.21 88.64 6 Patrice et al. [53] 2018 2635 NI DCNN 88.28 83.82 NI 7 Mizuho et al. [54] 2018 665 NI DCNN 68% NI NI NI not informed 1656 Neural Computing and Applications (2023) 35:1645–1659 123 k = 10. Random oversampling provides a balanced distribution by increasing the number of examples of the two classes through the random replication of examples of this class therefore the overall classification performance is improved. As it is known to all, deep learning architectures require a large amount of training data. Besides, most deep learning techniques, for example, CNN-based methods, require labeled data for supervised learning, which is difficult and time-consuming clinically. How to take the best advantage of limited data for training and how to train deeper networks effectively remains to be addressed. As the dataset was quite small, the DCNN classifier is overfitted. To tackle the overfitting problem we increased the number of images from 404 to 4640 images. Therefore, the processing time of the DCNN model is considered a limitation as it increased from 5 to around 20 min based on our hardware and software specifications. In the second phase of our CAD system, we intend to use a bigger dataset to generalize the proposed techniques. We demonstrated that our algorithm outperforms the previous state-of-the-art techniques in terms of segmentation (bounding box ? MIP) and (thresholding ? K-means clustering). The segmentation techniques are significantly robust for the two nodule types (solitary and juxtapleural), respectively. The results suggest that MIP can be used as a 2D segmentation technique achieving good results although it is generally in many researches used in a 3D volume rendering. Suggesting the suitability of our CAD system to minimize the radiologist’s effort through the early diagnosis of the lung nodules. Also, it can achieve diagnostic and therapeutic quality assurance. Although the model showed some promising results in the early diagnosis of lung cancer, the Juxtavascular nodule can be considered in the future, which would require a modification of the current segmentation algorithm. Also, the processor family of the hardware can be upgraded so that the computation time of the DCNN classifier will decrease and we can use a large database. Lung cancer diagnosis either benign or malignant according to its equivalent diameter can be done which requires a database with malignant and benign nodules. Acknowledgements We express our gratitude to the anonymous referees for their constructive reviews of the manuscript and for helpful comments. Funding Open access funding provided by The Science, Technology & Innovation Funding Authority (STDF) in cooperation with The Egyptian Knowledge Bank (EKB).

Declarations Conflict of interest The authors declare that they have no competing interests. Availability of data and material All data analyzed during this study are included in this published article. Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article’s Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article’s Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons. org/licenses/by/4.0/.

References

1. Siegel RL, Miller KD, Fuchs HE, Jemal A (2021) Cancer statistics, 2021. CA Cancer J Clin 71:7–3
2. Minna JD, Schiller JH (2008) Harrison’s principles of internal medicine, 17th edn. McGraw-Hill, New York, pp 551–562
3. Pen˜a DM, Luo S, Abdelgader AMS (2016) Auto diagnostics of lung nodules using minimal characteristics extraction technique. Diagnostics 6:13
4. Song QZ, Zhao L, Luo XK, Dou XC (2017) Using deep learning for classification of lung nodules on computed tomography images. J Healthc Eng 217:8314740
5. Liu Y, Yang J, Zhao D, Liu J (2009) Computer aided detection of lung nodules based on voxel analysis utilizing support vector machines. In: International conference on future biomedical information engineering
6. Namin ST, Abrishami H, Esmaeil-Zadeh M (2010) Automated detection and classification of pulmonary nodules in 3D thoracic CT images. In: 2010 IEEE international conference on systems, man and cybernetics, pp 4244–6588
7. Ozekes S, Osman O, Ucan ON (2008) Nodule detection in a lung region that’s segmented with using genetic cellular neural networks and 3D template matching with fuzzy rule based thresholding. Korean J Radiol 9:1–9
8. Jin X, Zhang Y, Jin Q (2016) Pulmonary nodule detection based on ct images using convolution neural network. In: 9th International symposium on computational intelligence and design
9. Heeneman T, Hoogendoorn M (2018) Lung nodule detection by using Deep Learning. VRIJE Universiteit Amsterdam, research paper. https://beta.vu.nl/nl/Images/werkstuk-heeneman\_tcm235- 876475.pdf
10. Wu S, Wang J (2012) Pulmonary nodules 3D detection on serial CT scans. In: Third global congress on intelligent systems
11. Yang M, Periaswam S, Wu Y (2007) False positive reduction in lung GGO nodule detection with 3D volume shape descriptor. In: 2007 IEEE international conference on acoustics, speech and signal processing—ICASSP ’07
12. Ye X, Lin X, Dehmeshki J, Beddoe G (2009) Shape-based computer-aided detection of lung nodules in thoracic CT images. IEEE Trans Biomed Eng 56:7
13. Antonelli M, Lazzerini B, Marcelloni F (2005) Segmentation and reconstruction of the lung volume in CT images. In: ACM symposium on applied computing (SAC), March (13–17)
14. Sasidhar B, Ramesh Babu DR, Bhaskarao N, Jan B (2013) Automated segmentation of lung regions and detection of lung cancer in CT scan. Int J Eng Adv Technol (IJEAT) 2:4
15. Zhu Q, Xiong H, Jiang X (2012) Pulmonary blood vessels and nodules segmentation via vessel energy function and radiusvariable sphere model. In: IEEE second conference on healthcare informatics, imaging and systems biology
16. Wang Q-z, Wang K, Guo Y, Wang X-z (2010) Automatic detection of pulmonary nodules in multi-slice CT based on 3D neural networks with adaptive initial weights. In: International conference on intelligent computation technology and automation
17. El-Regaily SA, Salem MAM, Abdel Aziz MH, Roushdy MI (2020) Multi-view convolutional neural network for lung nodule false positive reduction. Expert Syst Appl 162:113017
18. Blanc D, Racine V, Khalil A, Deloche M et al (2020) Artificial intelligence solution to classify pulmonary nodules on CT. Diagn Interv Imaging 101:803–810
19. Permatasar Z, Purnomo MH, Ketut Eddy Purnama I (2021) Lung nodule detection of CT and image-based GLCM and RLM CT scan using the support vector machine (SVM) method. J Adv Res Electr Eng 5:2
20. Lung Cancer Database (2020) https://www.cancerimagingarc hive.net/. Accessed June 2020
21. Banterle F, Corsini M, Cignoni P, Scopigno R (2012) A lowmemory, straightforward and fast bilateral filter through subsampling in spatial domain. Comput Graph Forum 31:19–32
22. Deswal S, Gupta S, Bhushan B (2015) A survey of various bilateral filtering techniques. Int J Signal Process Image Process Pattern Recognit 8:105–120
23. Mabrouk M, Karrar A, Sharawy A (2013) Support vector machine based computer aided diagnosis system for large lung nodules classification. J Med Imaging Health Inf 3:214–220
24. Grady L, Jolly MP (2011) Segmentation from a box. In: IEEE international conference on computer vision, ICCV, November 6–13
25. Abhinav K, Chauhan JS, Sarkar D (2018) Image segmentation of multi-shaped overlapping objects. In: International conference on computer vision theory and applications
26. Pandey RK, Mathurkar SS (2017) A review on morphological filter and its implementation. Int J Sci Res (IJSR) 6:1
27. Antropova N, Abe H, Giger ML (2018) Use of clinical MRI maximum intensity projections for improved breast lesion classification with deep convolutional neural networks. J Med Imaging (Bellingham) 5:1
28. Bae J, Yoo H (2018) Fast median filtering by use of fast localization of median value. Int J Appl Eng Res 13:10882–10885
29. Dorothy R, Joany RM, Joseph Rathish R, Santhana Prabha S, Rajendran S (2015) Image enhancement by histogram equalization. Int J Nano Corr Sci Eng. 2:21–30
30. Sasi NM, Jayasree VK (2013) Contrast limited adaptive histogram equalization for qualitative enhancement of myocardial perfusion images. Engineering 5:326–331
31. Miss HJ, Vala P, Baxi A (2013) A review on otsu image segmentation algorithm. Int J Adv Res Comput Eng Technol (IJARCET) 2:2
32. Rashmi MK, Saxena R (2013) Algorithm and technique on various edge detection: a survey. Signal Image Process Int J (Sipij) 4:3
33. Stosic Z, Rutesic P (2018) An improved canny edge detection algorithm for detecting brain tumors in MRI images. Int J Signal Process 3
34. Zheng X, Lei Q, Yao R, Gong Y, Yin Q (2018) Image segmentation based on adaptive K-means algorithm. EURASIP J Image Video Process 2018:1
35. Javaid M, Ali Shah SI, Ur Rehman Z, Javid M (2016) A novel approach to CAD system for the detection of lung nodules in CT images. Comput Methods Progr Biomed 135:125–139
36. Mabrouk M, Karrar A, Sharawy A (2012) Computer aided detection of large lung nodules using chest computer tomography images. Int J Appl Inf Syst (IJAIS) 3:9
37. Mondal A, Banerjee P, Tang H (2018) A novel feature extraction technique for pulmonary sound analysis based on EMD. Comput Methods Programs Biomed 159:199–209
38. Shan P (2018) Image segmentation method based on K-mean algorithm. EURASIP J Image Video Process 2018:81
39. He L, Chao Y, Zhao X, Yao B, Kasuya H, Ohta A (2017) An algorithm for calculating objects’ shape features in binary images. In: 2nd international conference on artificial intelligence and engineering applications (AIEA 2017)
40. Zawbaa HM, Emary E, CrinaGrosan VS (2018) Large-dimensionality small-instance set feature selection: a hybrid bio-inspired heuristic approach. Swarm Evol Comput 42:29–42
41. Stopel D, Boger Z, Moskovitch R, Shahar Y, Elovici Y (2006) Improving worm detection with artificial neural networks through feature selection and temporal analysis. Int J Appl Math Comput Sci 1:1
42. Karrar A, Mabrouk MS, Wahed MA (2020) Diagnosis of lung nodules from 2d computer tomography scans. Biomed Eng Appl Basis Commun 32:2
43. Ren R, Yang Y, Sun L (2020) Oversampling technique based on fuzzy representativeness difference for classifying imbalanced data. Appl Intell 50(8):2465–2487
44. Guodong Du, Zhang J, Luo Z, Ma F, Ma L, Li S (2020) Joint imbalanced classification and feature selection for hospital readmissions. Knowl Based Syst 200:106020
45. Dang Y, Jiang N, Hu H, Ji Z, Zhang W (2018) Image classification based on quantum KNN algorithm. arXiv:1805.06260v1 [cs.CV]. https://arxiv.org/abs/1805.06260
46. Abduh Z, Wahed MA, Kadah YM (2016) Robust computer-aided detection of pulmonary nodules from chest computed tomography. J Med Imaging Health Inf 6:1–7
47. Mehdy MM, Ng PY, Shair EF, Md Saleh NI, Gomes C (2017) Artificial neural networks in image processing for early detection of breast cancer. Hindawi Comput Math Methods Med 2017:1
48. Sehgal R, Gupta S (2016) Lung cancer detection using neural networks. Int J Adv Res Comput Sci Softw Eng 6:10
49. Kohad R, Ahire V (2014) Diagnosis of lung cancer using support vector machine with ant colony optimization technique. Int J Adv Comput Sci Technol (IJACST) 3:19–25
50. Yu Gu, Xiaoqi Lu, Zhang B, Zhao Y, Dahua Yu, Gao L, Cui G, Liang Wu, Zhou T (2019) Automatic lung nodule detection using multiscale dot nodule-enhancement filter and weighted support vector machines in chest computed tomography. PLoS ONE 14:1
51. Suresh S, Mohan S (2020) ROI-based feature learning for efficient true positive prediction using convolutional neural network for lung cancer diagnosis. Neural Comput Appl 32:15989–16009
52. Khehrah N, Farid MS, Bilal S, Khan MH (2020) Lung nodule detection in ct images using statistical and shape-based features. J Imaging 2020(6):6
53. Monkam P, Qi S, Mingjie Xu, Han F, Zhao X, Qian W (2018) CNN models discriminating between pulmonary micro nodules and non-nodules from CT images. BioMed Eng OnLine 17:17–96

54. Nishio M, Sugiyama O, Yakami M (2018) Computer-aided diagnosis of lung nodule classification between benign nodule, primary lung cancer, and metastatic lung cancer at different image size using deep convolutional neural network with transfer learning. PLoS ONE 13:7