BREAST CANCER PREDICTION

A PROJECT REPORT

In partial fulfilment of the requirements for the award of the degree

B.TECH

IN

COMPUTER SCIENCE AND ENGINEERING

Under the guidance of

GURUDEV ADHIKARY

And Technical Guidance by

Souvik Ganguly

BY

AVRADEEP NAYAK

Vellore Institute of Technology,

Bhopal

In association with

(ISO9001:2015)

(Note: All entries of the proforma of approval should be filled up with appropriate and complete information. Incomplete proforma of approval in any

respect will be summarily rejected.)

1. Title of the Project: **BREAST CANCER PREDICTION**
2. Project Made by: **AVRADEEP NAYAK**
3. Name of the guide: **Mr. GURUDEV ADHIKARY**
4. Name of the Technical Guide: Mr. Souvik Ganguly

<u>Project Version Control History</u>

Version	**Member**	**Description of Version**	**Date Completed**
Final	AVRADEEP NAYAK	Project Report	31/07/2022

Signature of Team Member

Date:

Signature of Approver

Date:

For Office Use Only

Approved

Not Approved

MR. GURUDEV ADHIKARY

Project Proposal Evaluator

DECLARATION

We hereby declare that the project work being presented in the project proposal entitled **"BREAST CANCER PREDICTION"** in partial fulfilment of the requirements for the award of the degree of **B.TECH** At **ANALYTICENS, INDIA,** is an authentic work carried out under the guidance of **MR. GURUDEV ADHIKARY** and technical guidance by **MR. Souvik Ganguly**. The matter embodied in this project work has not been submitted elsewhere for the award of any degree of our knowledge and belief.

Date:

Name of the Student : AVRADEEP NAYAK

Signature of the students:

ANALYTICENS (An ISO 9001:2015 Certified)

CERTIFICATE

This is to certify that this proposal of the minor project entitled **"BREAST CANCER PREDICTION"** is a record of bonafide work, carried out by **AVRADEEP NAYAK** under my guidance at **ANALYTICENS**. In my opinion, the report in its present form is in partial fulfilment of the requirements for the award of the degree of **B.TECH** and as per regulations of the **ANALYTICENS.** To the best of my knowledge, the results embodied in this report, are original in nature and worthy of incorporation in the present version of the report.

Guide / Supervisor

MR. GURUDEV ADHIKARY

Project Engineer

Analyticens (An ISO 9001:2015 Certified)

ACKNOWLEDGEMENT

Success of any project depends largely on the encouragement and guidelines of many others. I take this sincere opportunity to express my gratitude to the people who have been instrumental in the successful completion of this project work.

I would like to show our greatest appreciation to **MR. GURUDEV ADHIKARY**, Project Engineer at Analyticens, India. I always feel motivated and encouraged every time by his valuable advice and constant inspiration; without his encouragement and guidance this project would not have materialized.

Words are inadequate in offering our thanks to the other trainees, project assistants and other members at Analyticens for their encouragement and cooperation in carrying out this project work. The guidance and support received from all the members and who are contributing to this project, was vital for the success of this project.

CONTENTS

OVERVIEW

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and has fewer syntactical constructions than other languages.

Python is interpreted: Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to Perl and PHP.

Python is Interactive: You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python is Object-Oriented: Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

Python is a Beginner's Language: Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

HISTORY OF PYTHON

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands. Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, Small Talk, UNIX shell, and other scripting languages. Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL). Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

FEATURES OF PYTHON

Easy-to-learn: Python has few Keywords, simple structure and clearly defined syntax. This allows a student to pick up the language quickly.

Easy-to-Read: Python code is more clearly defined and visible to the eyes.

Easy -to-Maintain: Python's source code is fairly easy-to-maintain.

A broad standard library: Python's bulk of the library is very portable and cross platform compatible on UNIX, Windows, and Macintosh.

Interactive Mode: Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.

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

Extendable: You can add low level modules to the python interpreter. These modules enables programmers to add to or customize their tools to be more efficient.

Databases: Python provides interfaces to all major commercial databases.

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

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

Apart from the above-mentioned features, Python has a big list of good features, few are listed below:

- It support functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte code for building large applications.
- It provides very high level dynamic data types and supports dynamic type checking.
- It supports automatic garbage collections.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA and JAVA.

ENVIRONMENT SETUP

Open a terminal window and type "python" to find out if it is already installed and which version is installed.

- UNIX (Solaris, Linux, FreeBSD, AIX, HP/UX, SunOS, IRIX, etc.)
- Win 9x/NT/2000
- Macintosh (Intel, PPC, 68K)
- OS/2
- DOS (multiple versions)
- PalmOS
- Nokia mobile phones
- Windows CE
- Acorn/RISC OS

BASIC SYNTAX OF PYTHON PROGRAM

Type the following text at the Python prompt and press the Enter –

```
>>> print "Hello, Python!"
```

If you are running new version of Python, then you would need to use print statement with parenthesis as in **print ("Hello, Python!");**.
However in Python version 2.4.3, this produces the following result –

```
Hello, Python!
```

Python Identifiers

A Python identifier is a name used to identify a variable, function, class, module or other object. An identifier starts with a letter A to Z or a to z or an underscore (_) followed by zero or more letters, underscores and digits (0 to 9).
Python does not allow punctuation characters such as @, $, and % within identifiers. Python is a case sensitive programming language.

Python Keywords

The following list shows the Python keywords. These are reserved words and you cannot use them as constant or variable or any other identifier names. All the Python keywords contain lowercase letters only.

And, exec, not
Assert, finally, or
Break, for, pass
Class, from, print
continue, global, raise
def, if, return
del, import, try
elif, in, while
else, is, with
except, lambda, yield

Lines & Indentation

Python provides no braces to indicate blocks of code for class and function definitions or flow control. Blocks of code are denoted by line indentation, which is rigidly enforced.
The number of spaces in the indentation is variable, but all statements within the block must be indented the same amount. For example –

```
if True:
print "True"
```

```
else:
print "False"
```

Command Line Arguments

Many programs can be run to provide you with some basic information about how they should be run. Python enables you to do this with -h −

```
$ python-h
usage: python [option]...[-c cmd|-m mod | file |-][arg]...
```

Options and arguments (and corresponding environment variables):

-c cmd: program passed in as string(terminates option list)

-d   : debug output from parser (also PYTHONDEBUG=x)

-E   : ignore environment variables (such as PYTHONPATH)

-h   : print this help message and exit [etc.]

MODULES

A module allows you to logically organize your Python code. Grouping related code into a module makes the code easier to understand and use. A module is a Python object with arbitrarily named attributes that you can bind and reference .

The Python code for a module named *aname* normally resides in a file named *aname.py*. Here's an example of a simple module, support.py

```
def print_func( par ):
    print"Hello : ", par
    return
```

The *import* Statement

You can use any Python source file as a module by executing an import statement in some other Python source file. The *import* has the following syntax –

```
import module1[, module2[,… moduleN]
```

PACKAGES

A package is a hierarchical file directory structure that defines a single Python application environment that consists of modules and sub packages and sub-subpackages, and so on.

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

```
def Pots():
        print "I'm Pots Phone"
```

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

- *Phone/Isdn.py* file having function Isdn()
- *Phone/G3.py* file having function G3()

Now, create one more file __init__.py in *Phone* directory −

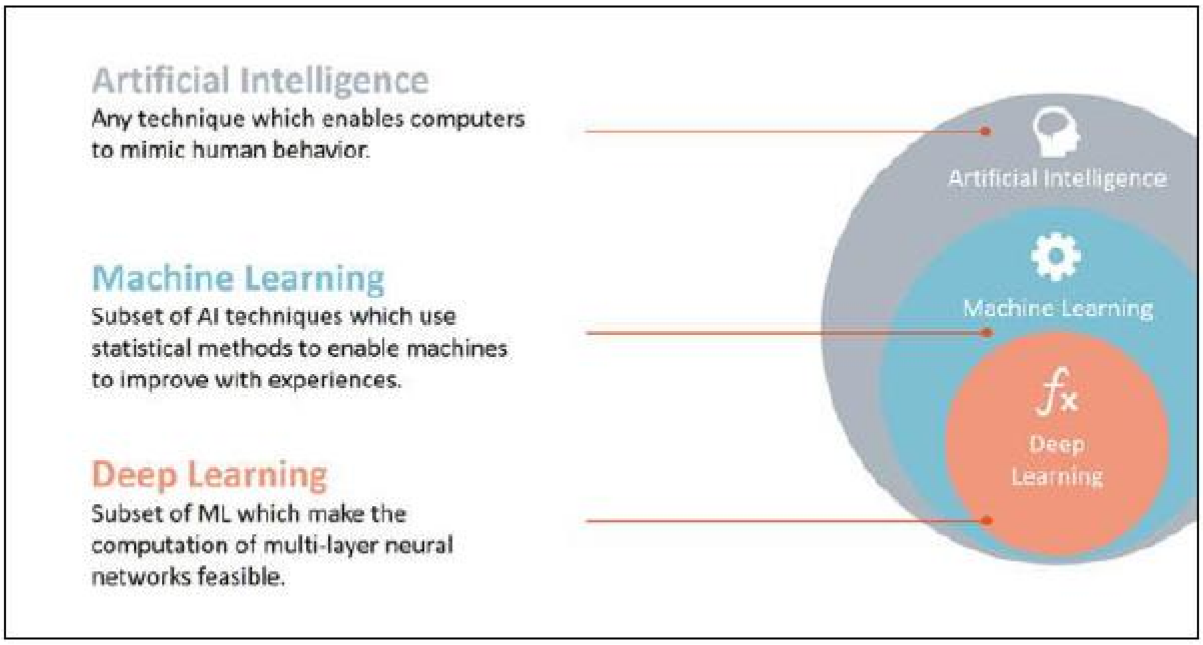
- Phone/__init__.py

To make all of your functions available when you've imported Phone, you need to put explicit import statements in __init__.py as follows −

```
from Pots import Pots
from Isdn import Isdn
from G3 import
```

ARTIFICIAL INTELLIGENCE

Introduction


Artificial Intelligence
Any technique which enables computers to mimic human behavior.
Machine Learning
Subset of AI techniques which use statistical methods to enable machines to improve with experiences.
Deep Learning
Subset of ML which make the computation of multi-layer neural networks feasible.
Artificial Intelligence
Machine Learning
Deep Learning

According to the father of Artificial Intelligence, John McCarthy, it is *"The science and engineering of making intelligent machines, especially intelligent computer programs".*

Artificial Intelligence is a way of **making a computer, a computer-controlled robot, or a software think intelligently**, in the similar manner the intelligent humans think.

AI is accomplished by studying how human brain thinks, and how humans learn, decide, and work while trying to solve a problem, and then using the outcomes of this study as a basis of developing intelligent software and systems.

The development of AI started with the intention of creating similar intelligence in machines that we find and regard high in humans.

Goals of AI

To Create Expert Systems − The systems which exhibit intelligent behaviour, learn, demonstrate, explain, and advice its users.
To Implement Human Intelligence in Machines − Creating systems that understand, think, learn, and behave like humans.

Applications of AI

AI has been dominant in various fields such as :-

Gaming − AI plays crucial role in strategic games such as chess, poker, tic-tac-toe, etc., where machine can think of large number of possible positions based on heuristic knowledge.

Natural Language Processing − It is possible to interact with the computer that understands natural language spoken by humans.

Expert Systems − There are some applications which integrate machine, software, and special information to impart reasoning and advising. They provide explanation and advice to the users.

Vision Systems − These systems understand, interpret, and comprehend visual input on the computer.

For example: A spying aeroplane takes photographs, which are used to figure out spatial information
or map of the areas.

Doctors use clinical expert system to diagnose the patient.

Police use computer software that can recognize the face of criminal with the stored
portrait made by forensic artist.

Speech Recognition − Some intelligent systems are capable of hearing and comprehending the language in terms of sentences and their meanings while a human talks to it. It can handle different accents, slang words, noise in the background, change in human's noise due to cold, etc.

Handwriting Recognition − The handwriting recognition software reads the text written on paper by a pen or on screen by a stylus. It can recognize the shapes of the letters and convert it into editable text.

Intelligent Robots − Robots are able to perform the tasks given by a human. They have sensors to detect physical data from the real world such as light, heat, temperature, movement, sound, bump, and pressure. They have efficient processors, multiple sensors and huge memory, to exhibit intelligence. In addition, they are capable of learning from their mistakes and they can adapt to the new environment.

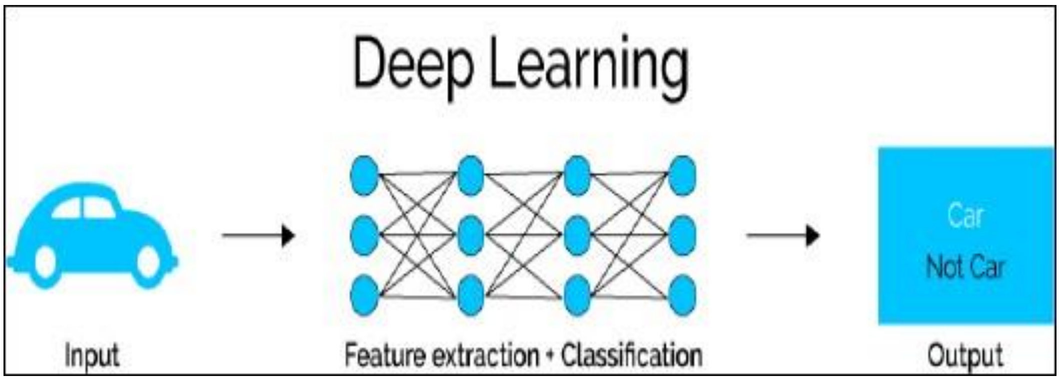
Application of AI

Deep Learning

Deep learning is a subset of machine learning. Usually, when people use the term deep learning, they are referring to deep artificial neural networks, and somewhat less frequently to deep reinforcement learning.

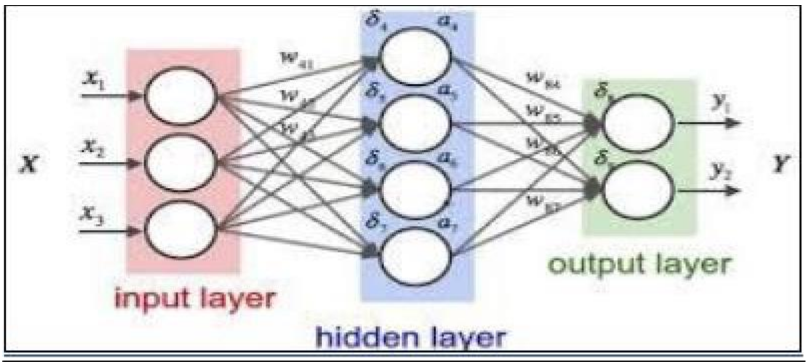
Deep learning is a class of machine learning algorithms that:

- use a cascade of multiple layers of nonlinear processing units for feature extraction and transformation. Each successive layer uses the output from the previous layer as input.
- learn in supervised (e.g., classification) and/or unsupervised (e.g., pattern analysis) manners.
- learn multiple levels of representations that correspond to different levels of abstraction; the levels form a hierarchy of concepts.
- use some form of gradient descent for training via backpropagation.



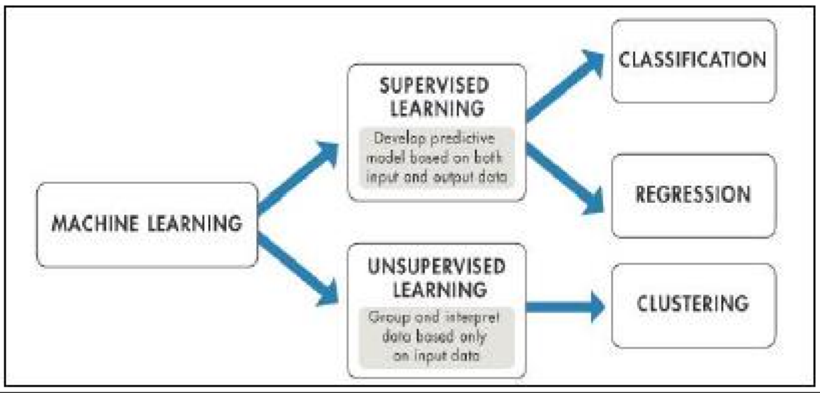
NEURAL NETWORKING

Artificial neural networks (**ANNs**) or **connectionist systems** are computing systems inspired by the biological neural networks that constitute animal brains. Such systems learn (progressively improve performance on) tasks by considering examples, generally without task-specific programming



An ANN is based on a collection of connected units or nodes called artificial neurons (analogous to biological neurons in an animal brain). Each connection between artificial neurons can transmit a signal from one to another.

MACHINE LEARNING


MACHINE LEARNING
SUPERVISED LEARNING
Develop predictive model based on both input and output data
UNSUPERVISED LEARNING
Group and interpret data based only on input data
CLASSIFICATION
REGRESSION
CLUSTERING

Machine learning is a field of computer science that gives computers the ability to learn without being explicitly programmed.
Evolved from the study of pattern recognition and computational learning theory in artificial intelligence, machine learning explores the study and construction of algorithms that can learn from and make predictions on data.

INTRODUCTION TO MACHINE LEARNING

Machine learning is a field of computer science that gives computers the ability to learn without being explicitly programmed.

Arthur Samuel, an American pioneer in the field of computer gaming and artificial intelligence, coined the term "Machine Learning" in 1959 while at IBM. Evolved from the study of pattern recognition and computational learning theory in artificial intelligence, machine learning explores the study and construction of algorithms that can learn from and make predictions on data

Machine learning tasks are typically classified into two broad categories, depending on whether there is a learning "signal" or "feedback" available to a learning system:-

SUPERVISED LEARNING

Supervised learning is the machine learning task of inferring a function from *labeled training data*.[1] The training data consist of a set of *training examples*. In supervised learning, each example is a *pair* consisting of an input object (typically a vector) and a desired output value.

A supervised learning algorithm analyses the training data and produces an inferred function, which can be used for mapping new examples. An optimal scenario will allow for the algorithm to correctly determine the class labels for unseen instances. This requires the learning algorithm to generalize from the training data to unseen situations in a "reasonable" way.

UNSUPERVISED LEARNING

Unsupervised learning is the machine learning task of inferring a function to describe hidden structure from "unlabelled" data (a classification or categorization is not included in the observations). Since the examples given to the learner are unlabelled, there is no evaluation of the accuracy of the structure that is output by the relevant algorithm—which is one way of distinguishing unsupervised learning from supervised learning and reinforcement learning.

A central case of unsupervised learning is the problem of density estimation in statistics, though unsupervised learning encompasses many other problems (and solutions) involving summarizing and explaining key features of the data.

NUMPY

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. The ancestor of NumPy, Numeric, was originally created by Jim Hugunin.

NumPy targets the CPython reference implementation of Python, which is a non-optimizing bytecode interpreter. Mathematical algorithms written for this version of Python often run much slower than compiled equivalents.

Using NumPy in Python gives functionality comparable to MATLAB since they are both interpreted, and they both allow the user to write fast programs as long as most operations work on arrays or matrices instead of scalars.

NUMPY ARRAY

NumPy’s main object is the homogeneous multidimensional array. It is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers. In NumPy dimensions are called *axes*. The number of axes is *rank*.

For example, the coordinates of a point in 3D space [1, 2, 1] is an array of rank 1, because it has one axis. That axis has a length of 3. In the example pictured below, the array has rank 2 (it is 2-dimensional). The first dimension (axis) has a length of 2, the second dimension has a length of 3.

```
[[ 1., 0., 0.],
[ 0., 1., 2.]]
```

NumPy’s array class is called *ndarray*. It is also known by the alias.

SLICING NUMPY ARRAY

```
importnumpy as np

a = np.array([[1,2,3],[3,4,5],[4,5,6]])
print 'Our array is:'
print a
print '\n'

print 'The items in the second column are:'
print a[...,1]
print '\n'

print 'The items in the second row are:'
print a[1,...]
print '\n'

print 'The items column 1 onwards are:'
print a[...,1:]
```

OUTPUT

```
Our array is:
[[1 2 3]
[3 4 5]
[4 5 6]]

The items in the second column are:
[2 4 5]

The items in the second row are:
[3 4 5]

The items column 1 onwards are:
[[2 3]
[4 5]
[5 6]]
```

SCIPY

modules for optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing, ODE solvers and other tasks common in science and engineering.

SciPy builds on the NumPy array object and is part of the NumPy stack which includes tools like Matplotlib, pandas and SymPy, and an expanding set of scientific computing libraries. This NumPy stack has similar users to other applications such as MATLAB, GNU Octave, and Scilab. The NumPy stack is also sometimes referred to as the SciPy stack.

The SciPy Library/Package

The SciPy package of key algorithms and functions core to Python's scientific computing capabilities. Available sub-packages include:

- **constants:** physical constants and conversion factors (since version 0.7.0)
- **cluster:** hierarchical clustering, vector quantization, K-means
- **fftpack:** Discrete Fourier Transform algorithms
- **integrate:** numerical integration routines
- **interpolate:** interpolation tools
- **io:** data input and output
- **lib:** Python wrappers to external libraries
- **linalg:** linear algebra routines
- **misc:** miscellaneous utilities (e.g. image reading/writing)
- **ndimage:** various functions for multi-dimensional image processing
- **optimize:** optimization algorithms including linear programming
- **signal:** signal processing tools
- **sparse:** sparse matrix and related algorithms
- **spatial:** KD-trees, nearest neighbours, distance functions
- **special:** special functions
- **stats:** statistical functions
- **weave:** tool for writing C/C++ code as Python multiline strings

Data Structures

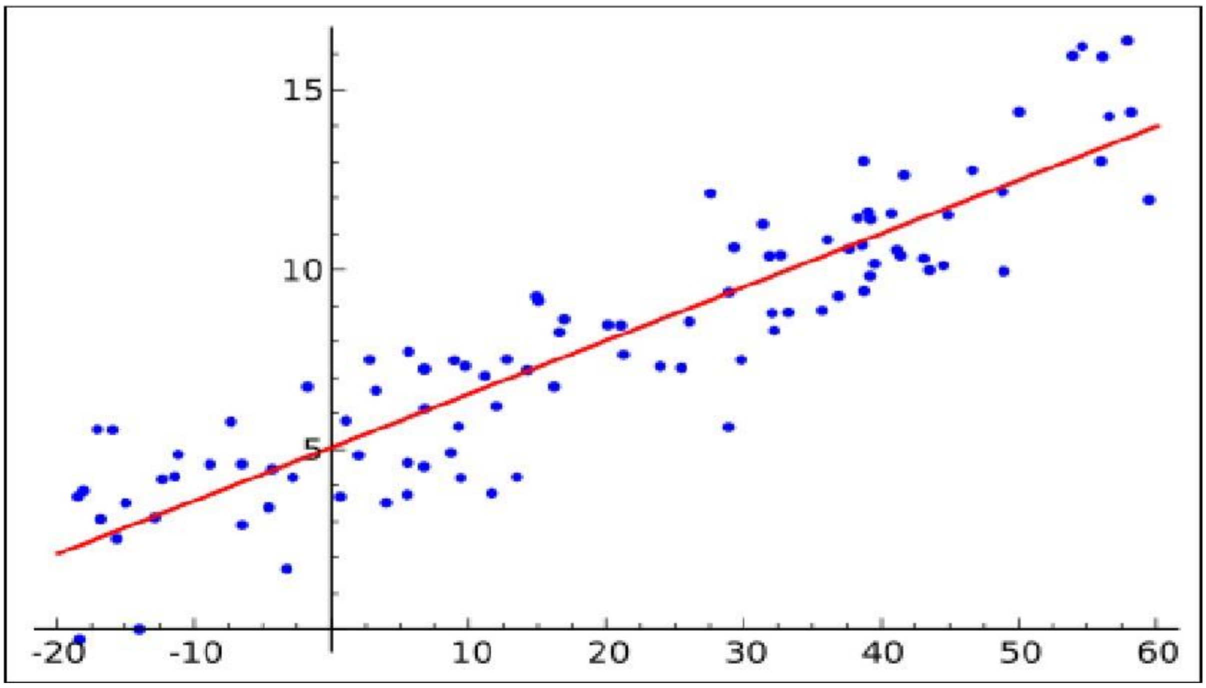
The basic data structure used by SciPy is a multidimensional array provided by the NumPy module. NumPy provides some functions for linear algebra, Fourier transforms and random number generation, but not with the generality of the equivalent functions in SciPy. NumPy can also be used as an efficient multi-dimensional container of data with arbitrary data-types. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases. Older versions of SciPy used Numeric as an array type, which is now deprecated in favour of the newer NumPy array code.

SCIKIT-LEARN

Scikit-learn is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, *k*-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

The scikit-learn project started as scikits.learn, a Google Summer of Code project by David Cournapeau. Its name stems from the notion that it is a "SciKit" (SciPy Toolkit), a separately-developed and distributed third-party extension to SciPy.[4] The original codebase was later rewritten by other developers. In 2010 Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort and Vincent Michel, all from INRIA took leadership of the project and made the first public release on February the 1st 2010[5]. Of the various scikits, scikit-learn as well as scikit-image were described as "well-maintained and popular" in November 2012.

REGRESSION ANALYSIS



In statistical modelling, **regression analysis** is a set of statistical processes for estimating the relationships among variables. It includes many techniques for modelling and analysing several variables, when the focus is on the relationship between a dependent variable and one or more independent variables (or 'predictors'). More specifically, regression analysis helps one understand how the typical value of the dependent variable (or 'criterion variable') changes when any one of the independent variables is varied, while the other independent variables are held fixed.

Regression analysis is widely used for prediction and forecasting, where its use has substantial overlap with the field of machine learning. Regression analysis is also used to understand which among the independent variables are related to the dependent variable, and to explore the forms of these relationships. In restricted circumstances, regression analysis can be used to infer casual relationships between the independent and dependent variables. However this can lead to illusions or false relationships, so caution is advisable

LINEAR REGRESSION

Linear regression is a linear approach for modelling the relationship between a scalar dependent variable y and one or more explanatory variables (or independent variables) denoted X. The case of one explanatory variable is called *simple linear regression*. For more than one explanatory variable, the process is called *multiple linear regression*.

In linear regression, the relationships are modelled using linear predictor functions whose unknown model parameters are estimated from the data. Such models are called *linear models*.

LOGISTIC REGRESSION

Logistic regression, or logit regression, or logit model [1] is a regression model where the dependent variable (DV) is categorical. This article covers the case of a binary dependent variable—that is, where the output can take only two values, "0" and "1", which represent outcomes such as pass/fail, win/lose, alive/dead or healthy/sick. Cases where the dependent variable has more than two outcome categories may be analysed in multinomial logistic regression, or, if the multiple categories are ordered, in ordinal logistic regression. In the terminology of economics, logistic regression is an example of a qualitative response/discrete choice model.

POLYNOMIAL REGRESSION

Polynomial regression is a form of regression analysis in which the relationship between the independent variable x and the dependent variable y is modelled as an n^{th} degree polynomial in x.

Polynomial regression fits a nonlinear relationship between the value of x and the corresponding conditional mean of y , denoted E(y | x), and has been used to describe nonlinear phenomena such as the growth rate of tissues, the distribution of carbon isotopes in lake sediments, and the progression of disease epidemics.

Although *polynomial regression* fits a nonlinear model to the data, as a statistical estimation problem it is linear, in the sense that the regression function E(y | x) is linear in the unknown parameters that are estimated from the data.

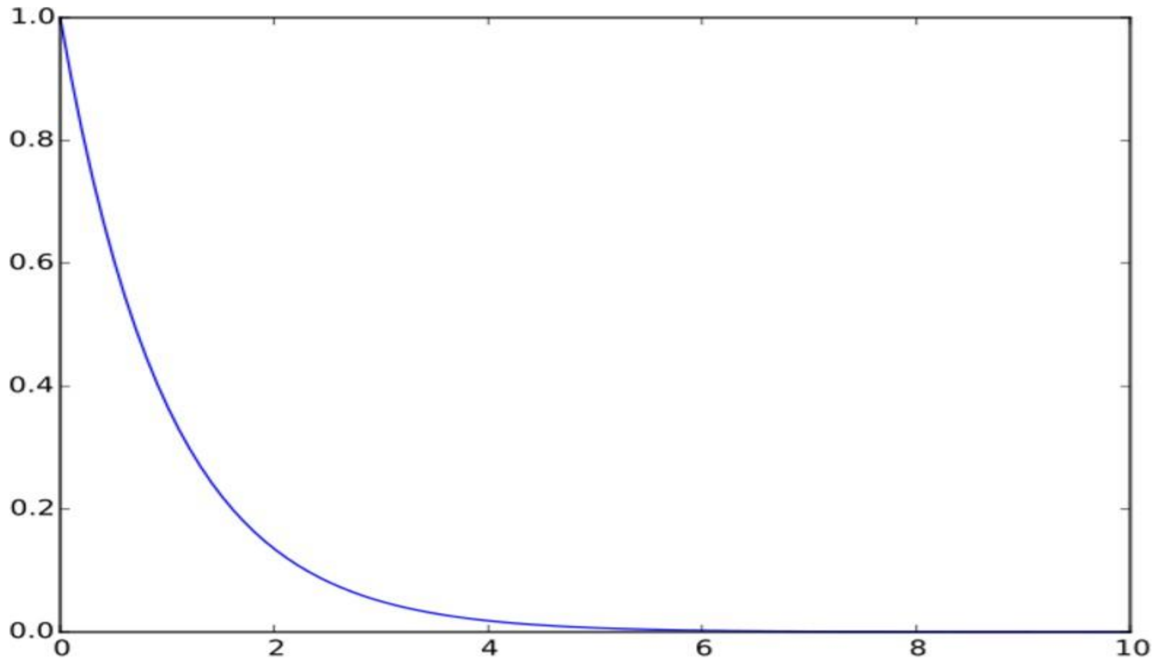
MATPLOTLIB

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter,wxPython, Qt, or GTK+. There is also a procedural "pylab" interface based on a state machine (like OpenGL), designed to closely resemble that of MATLAB, though its use is discouraged .SciPy makes use of matplotlib.

EXAMPLE

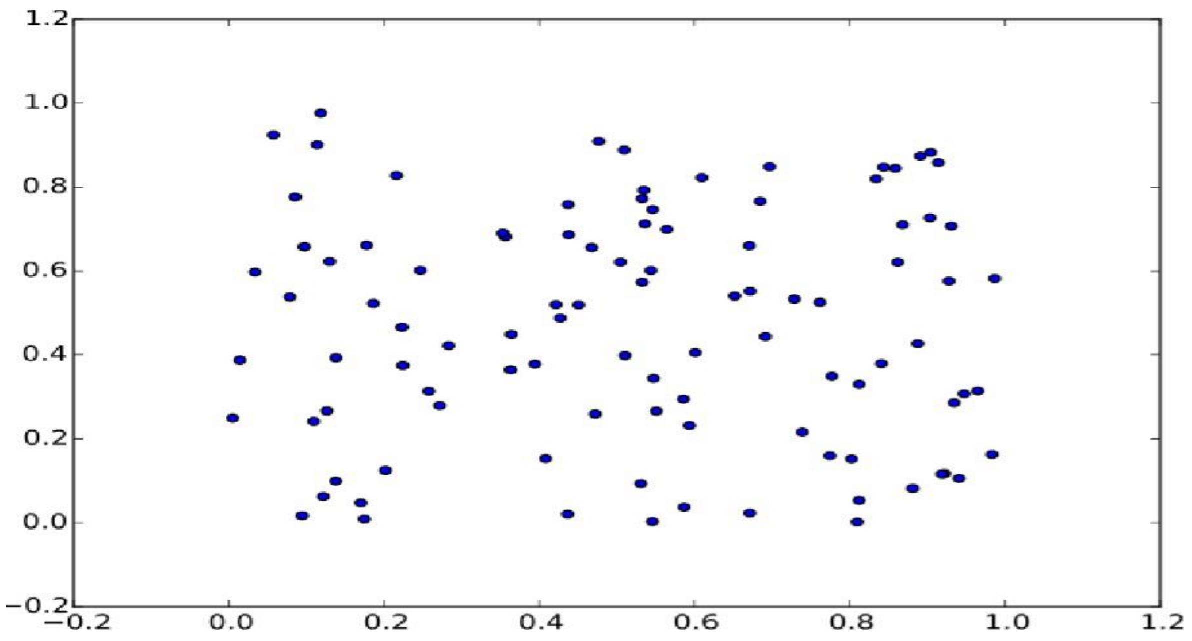
- **LINE PLOT**

```
>>>importmatplotlib.pyplotasplt
>>>importnumpyasnp
>>> a =np.linspace(0,10,100)
>>> b =np.exp(-a)
>>>plt.plot(a,b)
>>>plt.show()
```



- **SCATTER PLOT**

```
>>>importmatplotlib.pyplotasplt
>>>fromnumpy.randomimport rand
>>> a =rand(100)
>>> b =rand(100)
>>>plt.scatter(a,b)
>>>plt.show()
```



PANDAS

In computer programming, **pandas** is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license. "Panel data", an econometrics term for multidimensional, structured data sets.

LIBRARY FEATURES

- Data Frame object for data manipulation with integrated indexing.
- Tools for reading and writing data between in-memory data structures and different file formats.
- Data alignment and integrated handling of missing data.
- Reshaping and pivoting of data sets.
- Label-based slicing, fancy indexing, and sub setting of large data sets.
- Data structure column insertion and deletion.
- Group by engine allowing split-apply-combine operations on data sets.
- Data set merging and joining.
- Hierarchical axis indexing to work with high-dimensional data in a lower-dimensional data structure.
- Time series-functionality: Date range generation.

CLUSTERING

Cluster analysis or **clustering** is the task of grouping a set of objects in such a way that objects in the same group (called a **cluster**) are more similar (in some sense or another) to each other than to those in other groups (clusters). It is a main task of exploratory data mining, and a common technique for statistical data analysis, used in many fields, including machine learning, pattern recognition, image analysis, information retrieval, bioinformatics, data compression, and computer graphics.

Cluster analysis itself is not one specific algorithm, but the general task to be solved. It can be achieved by various algorithms that differ significantly in their notion of what constitutes a cluster and how to efficiently find them. Popular notions of clusters include groups with small distances among the cluster members, dense areas of the data space, intervals or particular statistical distributions. Clustering can therefore be formulated as a multi-objective optimization problem.

The appropriate clustering algorithm and parameter settings (including values such as the distance function to use, a density threshold or the number of expected clusters) depend on the individual data set and intended use of the results. Cluster analysis as such is not an automatic task, but an iterative process of knowledge discovery or interactive multi-objective optimization that involves trial and failure. It is often necessary to modify data pre-processing and model parameters until the result achieves the desired properties.

ALGORITHM

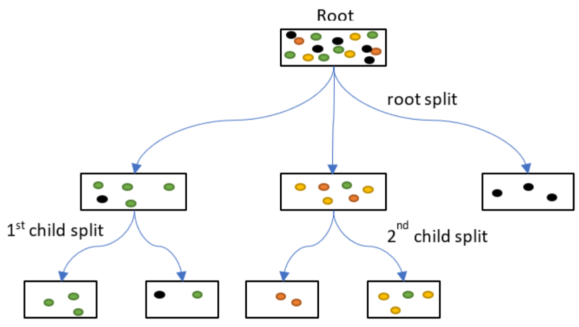
- Data Collection
- Data Formatting
- Model Selection
- Training
- Testing

Data Collection: We have collected data sets of weather from online website. We have downloaded the .csv files in which information was present.

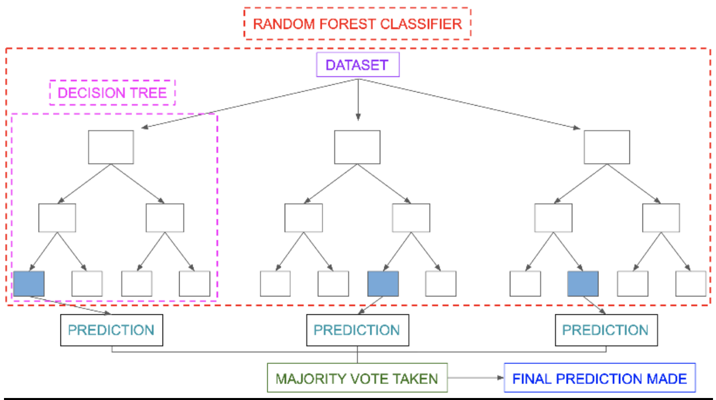
Data Formatting: The collected data is formatted into suitable data sets. We check the collinearity with mean temperature. The data sets which have collinearity nearer to 1.0 has been selected.

Model Selection: We have selected different models to minimize the error of the predicted value. The different models used are Decision Tree model, Random Forest model and SVM(Support Vector Machine) model.

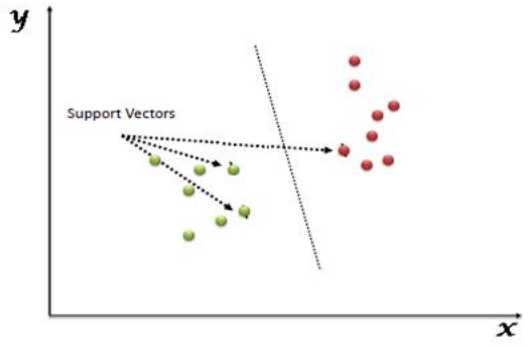
- Decision Tree model: In computational complexity the **decision tree model** is the model of computation in which an algorithm is considered to be basically a decision tree, i.e., a sequence of *queries* or *tests* that are done adaptively, so the outcome of the previous tests can influence the test is performed next.



- Random Forest Model: Random forest is a Supervised Machine Learning Algorithm that is used widely in Classification and Regression problems. It builds decision trees on different samples and takes their majority vote for classification and average in case of regression.



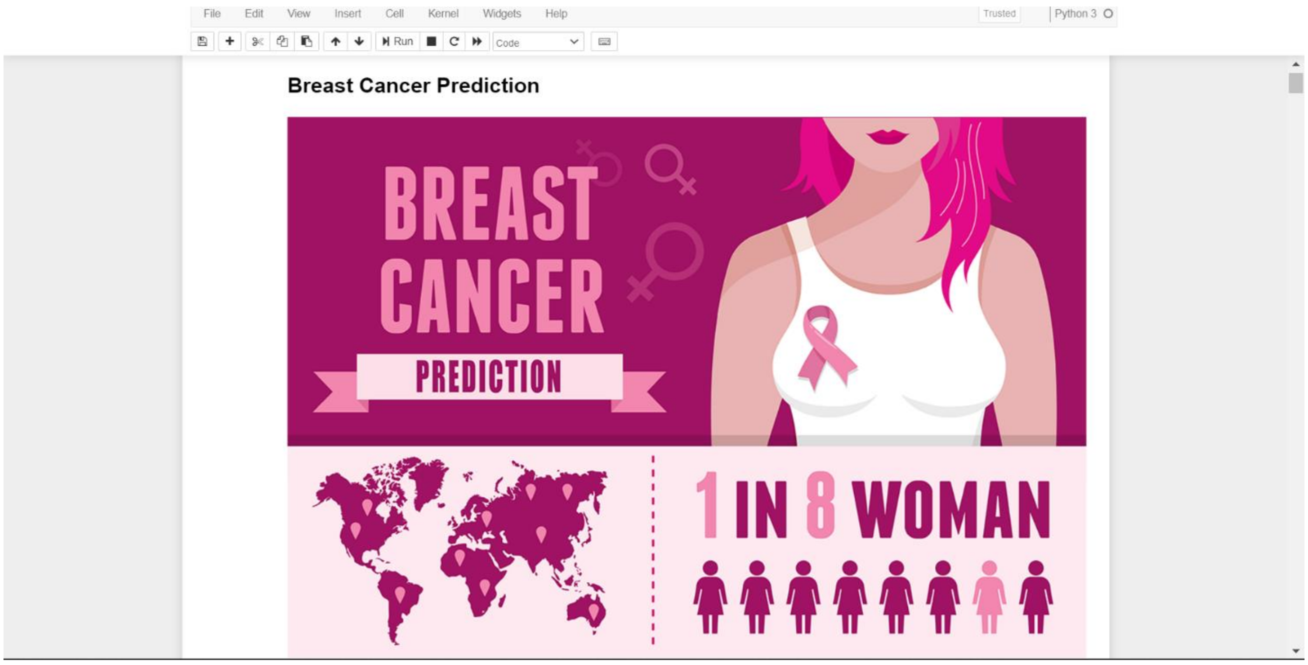
- SVM(Support Vector Machine): "Support Vector Machine" (SVM) is a supervised machine learning algorithm that can be used for both classification or regression challenges. However, it is mostly used in classification problems. In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is a number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the two classes very well (look at the below snapshot).



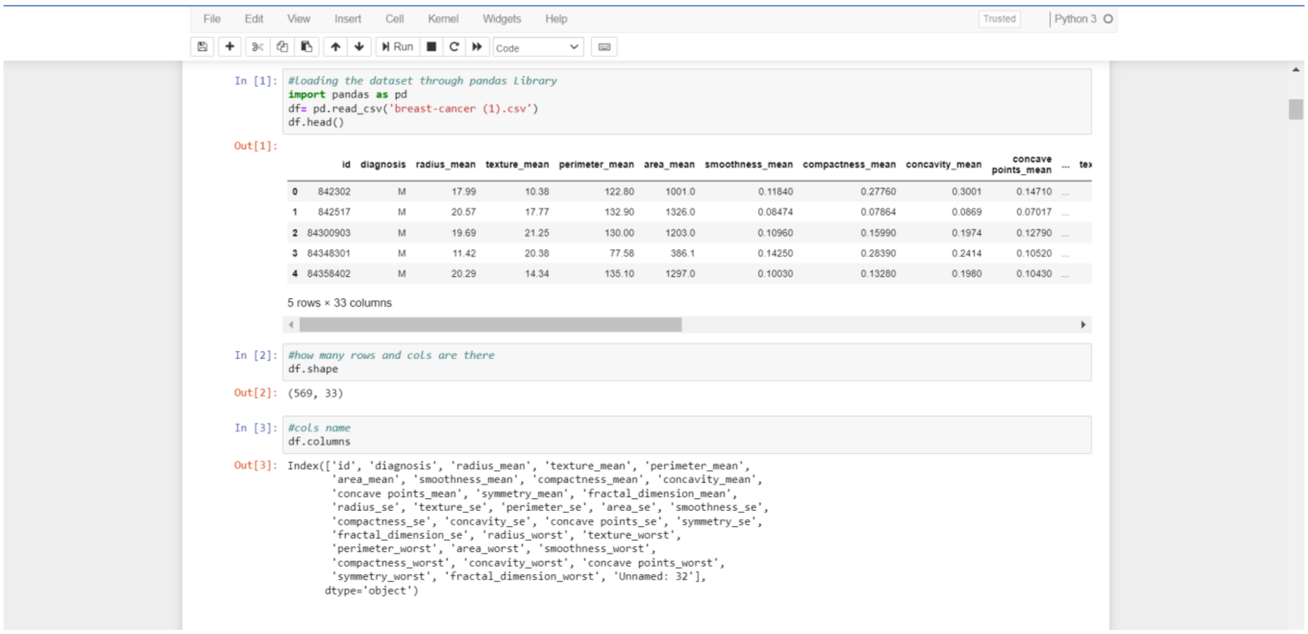
Training: The data sets was divided such that x_train is used to train the model with corresponding x_test values and some y_train kept reserved for testing.

Testing: The model was tested with y_train and stored in y_predict . Both y_train and y_predict was compared.

ACTUAL SCREENSHOTS FOR BREAST CANCER PREDICTION



Breast Cancer Prediction

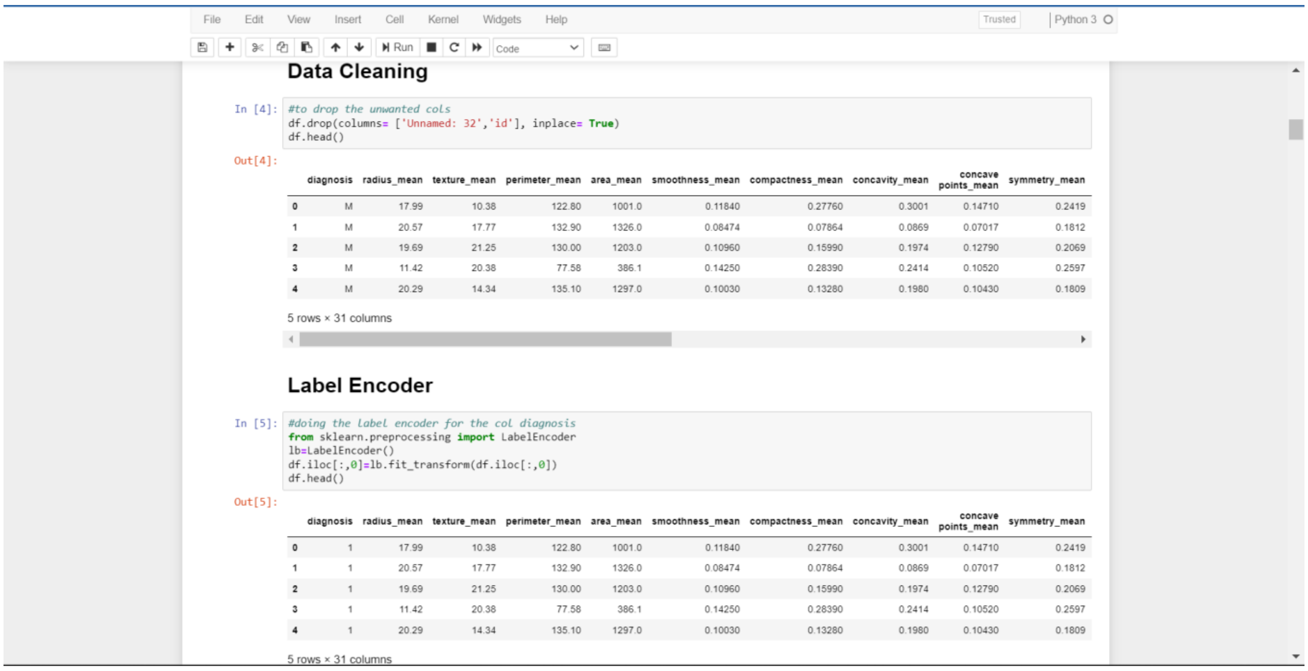


```
In [1]: #loading the dataset through pandas Library
        import pandas as pd
        df= pd.read_csv('breast-cancer (1).csv')
        df.head()

Out[1]:
```

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	...	tex
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	...	
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	...	
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	...	
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	...	
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	...	

5 rows × 33 columns

```
In [2]: #how many rows and cols are there
        df.shape

Out[2]: (569, 33)

In [3]: #cols name
        df.columns

Out[3]: Index(['id', 'diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean',
               'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean',
               'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean',
               'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_se',
               'compactness_se', 'concavity_se', 'concave points_se', 'symmetry_se',
               'fractal_dimension_se', 'radius_worst', 'texture_worst',
               'perimeter_worst', 'area_worst', 'smoothness_worst',
               'compactness_worst', 'concavity_worst', 'concave points_worst',
               'symmetry_worst', 'fractal_dimension_worst', 'Unnamed: 32'],
              dtype='object')
```



Data Cleaning

In [4]:
```
#to drop the unwanted cols
df.drop(columns= ['Unnamed: 32','id'], inplace= True)
df.head()
```

Out[4]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_mean
0	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419
1	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812
2	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069
3	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597
4	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809

5 rows × 31 columns

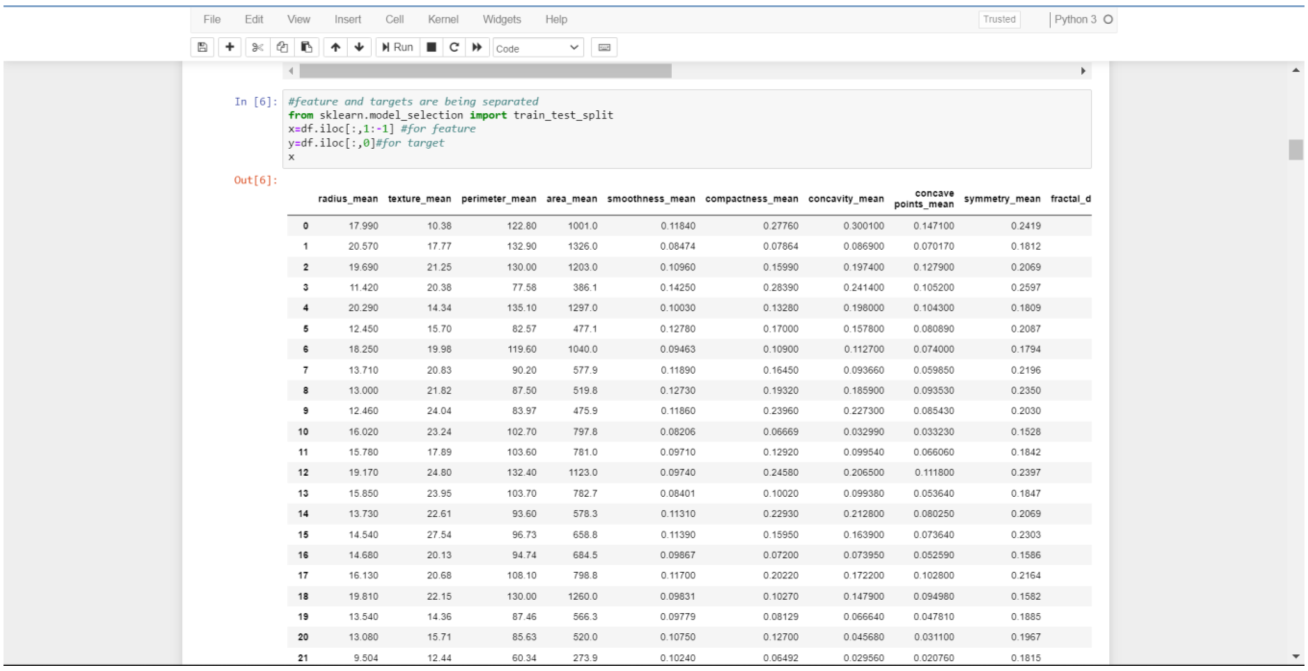
Label Encoder

In [5]:
```
#doing the label encoder for the col diagnosis
from sklearn.preprocessing import LabelEncoder
lb=LabelEncoder()
df.iloc[:,0]=lb.fit_transform(df.iloc[:,0])
df.head()
```

Out[5]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_mean
0	1	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419
1	1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812
2	1	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069
3	1	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597
4	1	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809

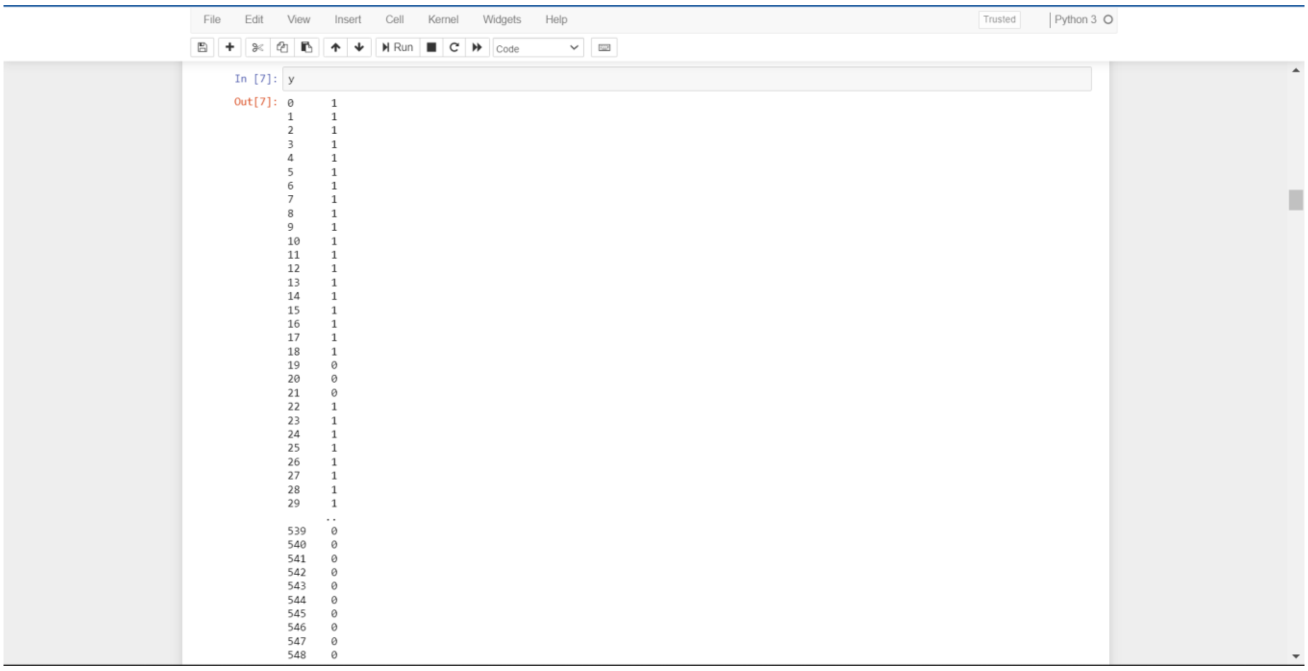
5 rows × 31 columns



In [6]:
```
#feature and targets are being separated
from sklearn.model_selection import train_test_split
x=df.iloc[:,1:-1] #for feature
y=df.iloc[:,0]#for target
x
```

Out[6]:

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_mean	fractal_d
0	17.990	10.38	122.80	1001.0	0.11840	0.27760	0.300100	0.147100	0.2419	
1	20.570	17.77	132.90	1326.0	0.08474	0.07864	0.086900	0.070170	0.1812	
2	19.690	21.25	130.00	1203.0	0.10960	0.15990	0.197400	0.127900	0.2069	
3	11.420	20.38	77.58	386.1	0.14250	0.28390	0.241400	0.105200	0.2597	
4	20.290	14.34	135.10	1297.0	0.10030	0.13280	0.198000	0.104300	0.1809	
5	12.450	15.70	82.57	477.1	0.12780	0.17000	0.157800	0.080890	0.2087	
6	18.250	19.98	119.60	1040.0	0.09463	0.10900	0.112700	0.074000	0.1794	
7	13.710	20.83	90.20	577.9	0.11890	0.16450	0.093660	0.059850	0.2196	
8	13.000	21.82	87.50	519.8	0.12730	0.19320	0.185900	0.093530	0.2350	
9	12.460	24.04	83.97	475.9	0.11860	0.23960	0.227300	0.085430	0.2030	
10	16.020	23.24	102.70	797.8	0.08206	0.06669	0.032990	0.033230	0.1528	
11	15.780	17.89	103.60	781.0	0.09710	0.12920	0.099540	0.066060	0.1842	
12	19.170	24.80	132.40	1123.0	0.09740	0.24580	0.206500	0.111800	0.2397	
13	15.850	23.95	103.70	782.7	0.08401	0.10020	0.099380	0.053640	0.1847	
14	13.730	22.61	93.60	578.3	0.11310	0.22930	0.212800	0.080250	0.2069	
15	14.540	27.54	96.73	658.8	0.11390	0.15950	0.163900	0.073640	0.2303	
16	14.680	20.13	94.74	684.5	0.09867	0.07200	0.073950	0.052590	0.1586	
17	16.130	20.68	108.10	798.8	0.11700	0.20220	0.172200	0.102800	0.2164	
18	19.810	22.15	130.00	1260.0	0.09831	0.10270	0.147900	0.094980	0.1582	
19	13.540	14.36	87.46	566.3	0.09779	0.08129	0.066640	0.047810	0.1885	
20	13.080	15.71	85.63	520.0	0.10750	0.12700	0.045680	0.031100	0.1967	
21	9.504	12.44	60.34	273.9	0.10240	0.06492	0.029560	0.020760	0.1815	



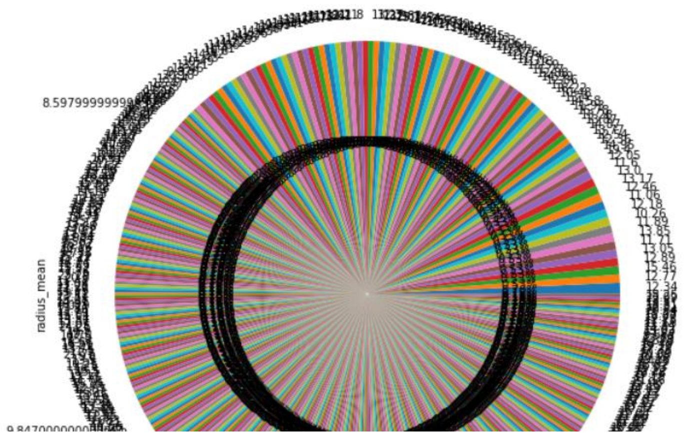
```
In [7]: y

Out[7]: 0      1
        1      1
        2      1
        3      1
        4      1
        5      1
        6      1
        7      1
        8      1
        9      1
        10     1
        11     1
        12     1
        13     1
        14     1
        15     1
        16     1
        17     1
        18     1
        19     0
        20     0
        21     0
        22     1
        23     1
        24     1
        25     1
        26     1
        27     1
        28     1
        29     1
              ..
        539    0
        540    0
        541    0
        542    0
        543    0
        544    0
        545    0
        546    0
        547    0
        548    0
```

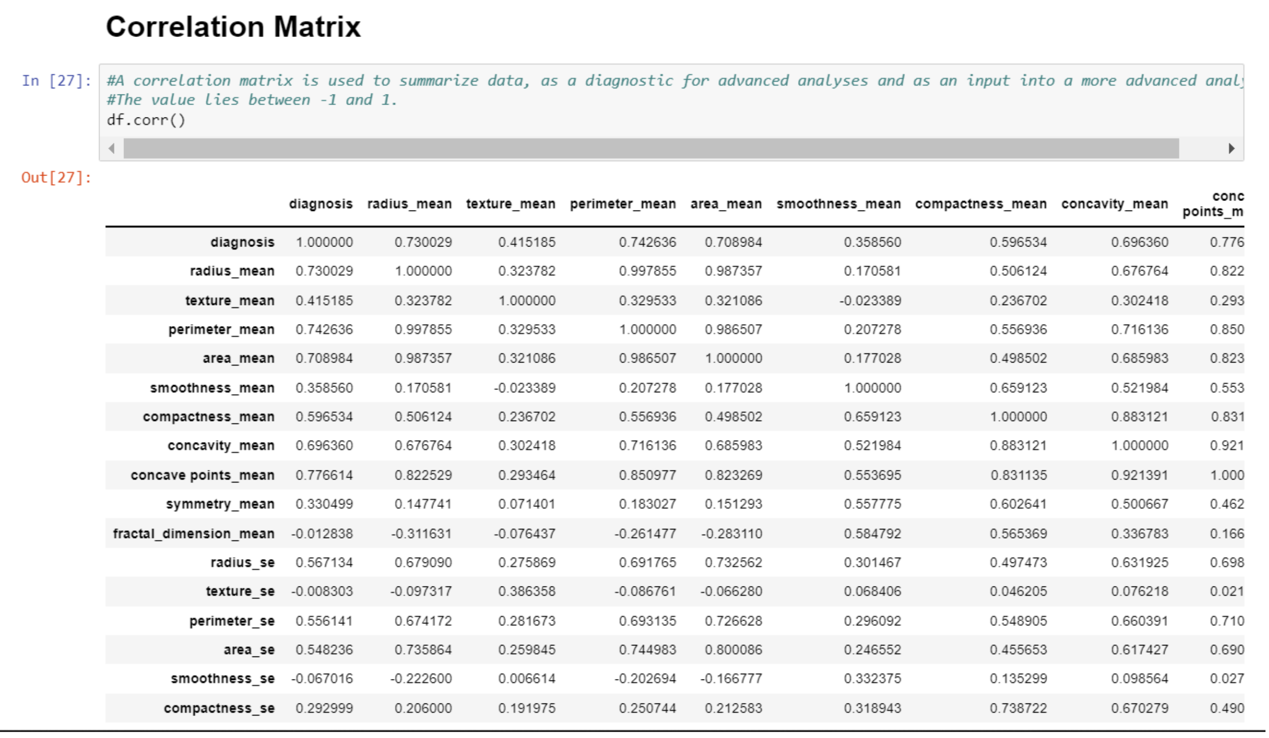
```
In [10]: #x_train data being normalized into 0 to 1
         x_train_norm

Out[10]: array([[0.1452506 , 0.32448133, 0.14249188, ..., 0.08426518, 0.22387186,
                 0.26197516],
                [0.18074684, 0.50912863, 0.17275931, ..., 0.15391374, 0.25783672,
                 0.27597083],
                [0.43348005, 0.21369295, 0.41814664, ..., 0.18450479, 0.38890803,
                 0.23910901],
                ...,
                [0.11619102, 0.35726141, 0.11077327, ..., 0.0913738 , 0.17402687,
                 0.17524147],
                [0.12963226, 0.35311203, 0.11706171, ..., 0.        , 0.        ,
                 0.06780997],
                [0.21434995, 0.59004149, 0.21235575, ..., 0.2899361 , 0.33251808,
                 0.10782574]])
```

```
In [11]: #x_test data being normalized into 0 to 1
         x_test_norm

Out[11]: array([[0.39000069, 0.3387381 , 0.38790307, ..., 0.43641026, 0.704811  ,
                 0.51654939],
                [0.37693419, 0.50546352, 0.34459073, ..., 0.11880342, 0.20635739,
                 0.21918165],
                [0.43263875, 0.16743038, 0.39696623, ..., 0.05350427, 0.28233677,
                 0.13891061],
                ...,
                [0.85626848, 0.30419457, 0.81873688, ..., 0.45367521, 0.7467354 ,
                 0.37242637],
                [0.7276666 , 0.34085301, 0.69471475, ..., 0.32128205, 0.51890034,
                 0.38337243],
                [0.50278523, 0.20549877, 0.48416333, ..., 0.15863248, 0.34982818,
                 0.14959604]])
```

```
In [13]: #x_train data being standardized
         x_train_sc

Out[13]: array([[-1.15036482, -0.39064196, -1.12855021, ..., -0.81232053,
                 -0.75798367, -0.01614761],
                [-0.93798972,  0.68051405, -0.94820146, ..., -0.37504806,
                 -0.60687023,  0.09669004],
                [ 0.574121  , -1.03333557,  0.51394098, ..., -0.18298917,
                 -0.02371948, -0.20050207],
                ...,
                [-1.32422924, -0.20048168, -1.31754581, ..., -0.76769066,
                 -0.97974953, -0.71542314],
                [-1.24380987, -0.2245526 , -1.28007609, ..., -1.34136004,
                 -1.75401433, -1.58157125],
                [-0.73694129,  1.14989702, -0.71226578, ...,  0.47893704,
                 -0.27460457, -1.25895095]])

In [14]: #x_test data being standardized
         x_test_sc

Out[14]: array([[-0.22609091,  0.14299357, -0.16219992, ...,  0.91450514,
                  1.33438591,  1.22101459],
                [-0.28072076,  1.13113906, -0.34954245, ..., -0.63253907,
                 -0.81952682, -0.77541863],
                [-0.04782508, -0.87231025, -0.12299829, ..., -0.95060736,
                 -0.49120548, -1.31433312],
                ...,
                [ 1.7233322 , -0.06173848,  1.70132185, ...,  0.99860173,
                  1.51554921,  0.25341812],
                [ 1.18565945,  0.15552818,  1.16487847, ...,  0.35372243,
                  0.53103066,  0.32690646],
                [ 0.24545096, -0.64668718,  0.25416267, ..., -0.43853407,
                 -0.19956228, -1.2425945 ]])
```

```
In [20]: df['radius_mean'].mean()

Out[20]: 14.127291739894563

In [21]: df['radius_mean'].median()

Out[21]: 13.37

In [22]: df['radius_mean'].std()

Out[22]: 3.524048826212078

In [23]: #pie plot for radius mean value counts
         plt.figure(figsize=(15,10))
         df['radius_mean'].value_counts().plot.pie(autopct='%1.2f%%')
         plt.show()
```



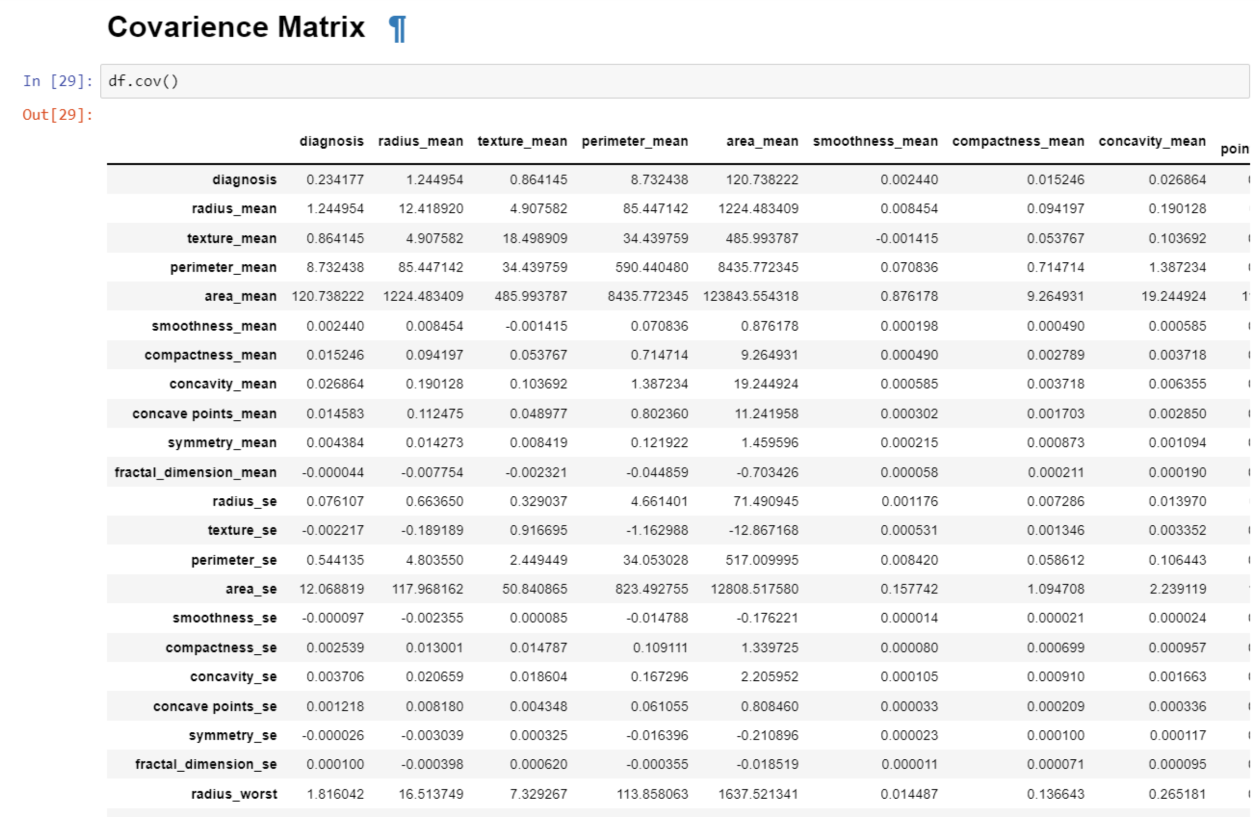


Correlation Matrix

In [27]:
```
#A correlation matrix is used to summarize data, as a diagnostic for advanced analyses and as an input into a more advanced analy
#The value lies between -1 and 1.
df.corr()
```

Out[27]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	conc points_m
diagnosis	1.000000	0.730029	0.415185	0.742636	0.708984	0.358560	0.596534	0.696360	0.776
radius_mean	0.730029	1.000000	0.323782	0.997855	0.987357	0.170581	0.506124	0.676764	0.822
texture_mean	0.415185	0.323782	1.000000	0.329533	0.321086	-0.023389	0.236702	0.302418	0.293
perimeter_mean	0.742636	0.997855	0.329533	1.000000	0.986507	0.207278	0.556936	0.716136	0.850
area_mean	0.708984	0.987357	0.321086	0.986507	1.000000	0.177028	0.498502	0.685983	0.823
smoothness_mean	0.358560	0.170581	-0.023389	0.207278	0.177028	1.000000	0.659123	0.521984	0.553
compactness_mean	0.596534	0.506124	0.236702	0.556936	0.498502	0.659123	1.000000	0.883121	0.831
concavity_mean	0.696360	0.676764	0.302418	0.716136	0.685983	0.521984	0.883121	1.000000	0.921
concave points_mean	0.776614	0.822529	0.293464	0.850977	0.823269	0.553695	0.831135	0.921391	1.000
symmetry_mean	0.330499	0.147741	0.071401	0.183027	0.151293	0.557775	0.602641	0.500667	0.462
fractal_dimension_mean	-0.012838	-0.311631	-0.076437	-0.261477	-0.283110	0.584792	0.565369	0.336783	0.166
radius_se	0.567134	0.679090	0.275869	0.691765	0.732562	0.301467	0.497473	0.631925	0.698
texture_se	-0.008303	-0.097317	0.386358	-0.086761	-0.066280	0.068406	0.046205	0.076218	0.021
perimeter_se	0.556141	0.674172	0.281673	0.693135	0.726628	0.296092	0.548905	0.660391	0.710
area_se	0.548236	0.735864	0.259845	0.744983	0.800086	0.246552	0.455653	0.617427	0.690
smoothness_se	-0.067016	-0.222600	0.006614	-0.202694	-0.166777	0.332375	0.135299	0.098564	0.027
compactness_se	0.292999	0.206000	0.191975	0.250744	0.212583	0.318943	0.738722	0.670279	0.490

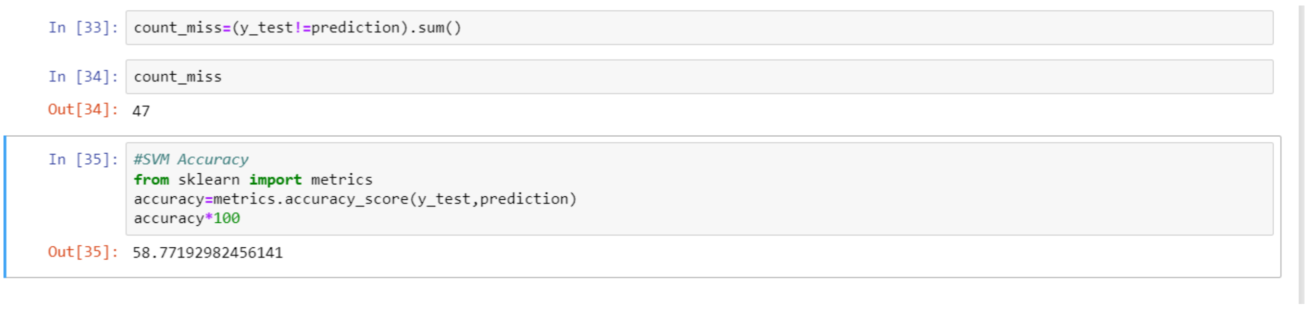


Covarience Matrix ¶

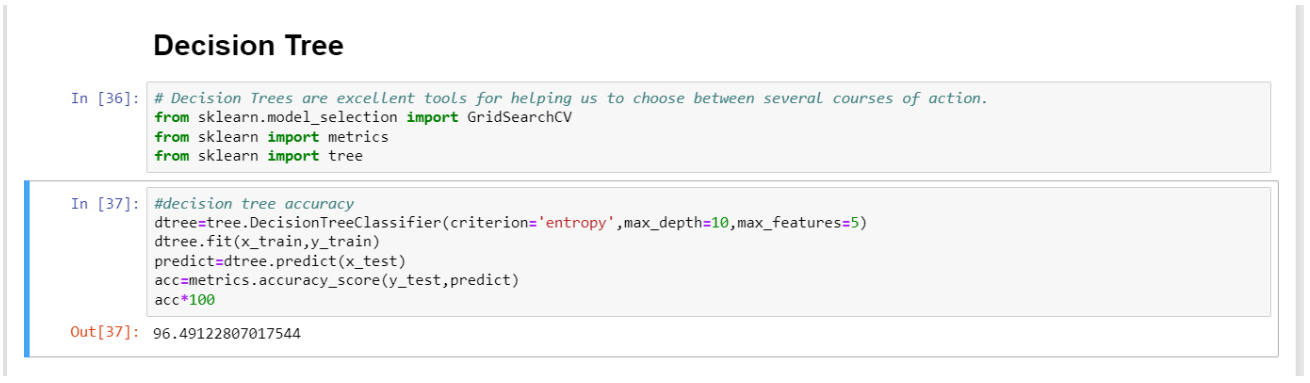
In [29]:
```
df.cov()
```

Out[29]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	poin
diagnosis	0.234177	1.244954	0.864145	8.732438	120.738222	0.002440	0.015246	0.026864	
radius_mean	1.244954	12.418920	4.907582	85.447142	1224.483409	0.008454	0.094197	0.190128	
texture_mean	0.864145	4.907582	18.498909	34.439759	485.993787	-0.001415	0.053767	0.103692	
perimeter_mean	8.732438	85.447142	34.439759	590.440480	8435.772345	0.070836	0.714714	1.387234	
area_mean	120.738222	1224.483409	485.993787	8435.772345	123843.554318	0.876178	9.264931	19.244924	1
smoothness_mean	0.002440	0.008454	-0.001415	0.070836	0.876178	0.000198	0.000490	0.000585	
compactness_mean	0.015246	0.094197	0.053767	0.714714	9.264931	0.000490	0.002789	0.003718	
concavity_mean	0.026864	0.190128	0.103692	1.387234	19.244924	0.000585	0.003718	0.006355	
concave points_mean	0.014583	0.112475	0.048977	0.802360	11.241958	0.000302	0.001703	0.002850	
symmetry_mean	0.004384	0.014273	0.008419	0.121922	1.459596	0.000215	0.000873	0.001094	
fractal_dimension_mean	-0.000044	-0.007754	-0.002321	-0.044859	-0.703426	0.000058	0.000211	0.000190	
radius_se	0.076107	0.663650	0.329037	4.661401	71.490945	0.001176	0.007286	0.013970	
texture_se	-0.002217	-0.189189	0.916695	-1.162988	-12.867168	0.000531	0.001346	0.003352	
perimeter_se	0.544135	4.803550	2.449449	34.053028	517.009995	0.008420	0.058612	0.106443	
area_se	12.068819	117.968162	50.840865	823.492755	12808.517580	0.157742	1.094708	2.239119	
smoothness_se	-0.000097	-0.002355	0.000085	-0.014788	-0.176221	0.000014	0.000021	0.000024	
compactness_se	0.002539	0.013001	0.014787	0.109111	1.339725	0.000080	0.000699	0.000957	
concavity_se	0.003706	0.020659	0.018604	0.167296	2.205952	0.000105	0.000910	0.001663	
concave points_se	0.001218	0.008180	0.004348	0.061055	0.808460	0.000033	0.000209	0.000336	
symmetry_se	-0.000026	-0.003039	0.000325	-0.016396	-0.210896	0.000023	0.000100	0.000117	
fractal_dimension_se	0.000100	-0.000398	0.000620	-0.000355	-0.018519	0.000011	0.000071	0.000095	
radius_worst	1.816042	16.513749	7.329267	113.858063	1637.521341	0.014487	0.136643	0.265181	

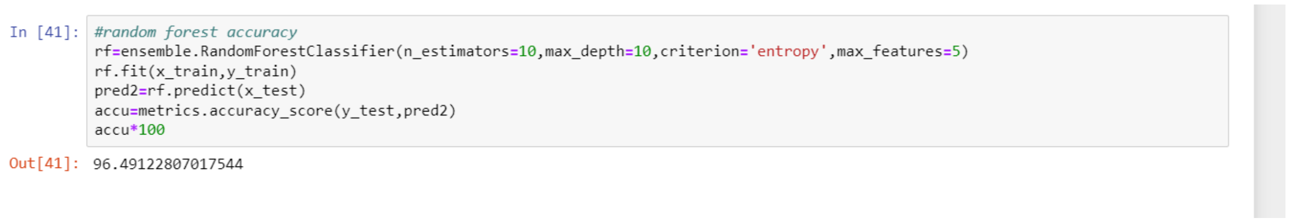


```
In [33]: count_miss=(y_test!=prediction).sum()

In [34]: count_miss
Out[34]: 47

In [35]: #SVM Accuracy
         from sklearn import metrics
         accuracy=metrics.accuracy_score(y_test,prediction)
         accuracy*100
Out[35]: 58.77192982456141
```

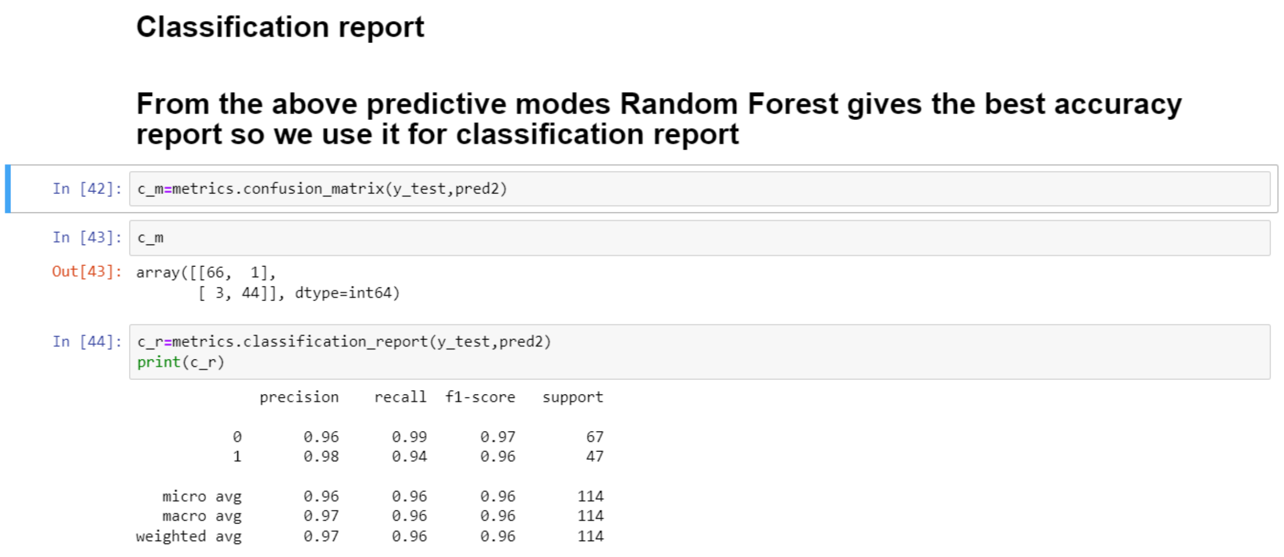


Decision Tree

```
In [36]: # Decision Trees are excellent tools for helping us to choose between several courses of action.
         from sklearn.model_selection import GridSearchCV
         from sklearn import metrics
         from sklearn import tree

In [37]: #decision tree accuracy
         dtree=tree.DecisionTreeClassifier(criterion='entropy',max_depth=10,max_features=5)
         dtree.fit(x_train,y_train)
         predict=dtree.predict(x_test)
         acc=metrics.accuracy_score(y_test,predict)
         acc*100
Out[37]: 96.49122807017544
```



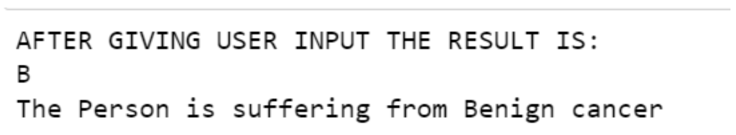
```
In [41]: #random forest accuracy
         rf=ensemble.RandomForestClassifier(n_estimators=10,max_depth=10,criterion='entropy',max_features=5)
         rf.fit(x_train,y_train)
         pred2=rf.predict(x_test)
         accu=metrics.accuracy_score(y_test,pred2)
         accu*100
Out[41]: 96.49122807017544
```



Classification report

From the above predictive modes Random Forest gives the best accuracy report so we use it for classification report

```
In [42]: c_m=metrics.confusion_matrix(y_test,pred2)

In [43]: c_m
Out[43]: array([[66,  1],
                [ 3, 44]], dtype=int64)

In [44]: c_r=metrics.classification_report(y_test,pred2)
         print(c_r)

              precision    recall  f1-score   support

           0       0.96      0.99      0.97        67
           1       0.98      0.94      0.96        47

   micro avg       0.96      0.96      0.96       114
   macro avg       0.97      0.96      0.96       114
weighted avg       0.97      0.96      0.96       114
```



```
AFTER GIVING USER INPUT THE RESULT IS:
B
The Person is suffering from Benign cancer
```

CONCLUSION

We have collected the raw data from online sourceskaggle.
Then we have taken this raw data and format it.
We have used some models specifically :

- SVM
- Decision Tee
- Random Forest

Random Forest classifier model is best one as MSE (Mean Squared Error).

BIBLIOGRAPHY

- www.wikipedia.com
- Used of jupyter notebook for implementing
- www.kaggle.com for the dataset.
- Theory images from google.

THANK YOU