**Chapter 1**

**Introduction**

Breast cancer (BC) is one of the most common cancers among women worldwide, representing the majority of new cancer cases and cancer-related deaths according to global statistics, making it a significant public health problem in today’s society.

The early diagnosis of BC can improve the prognosis and chance of survival significantly, as it can promote timely clinical treatment to patients. Further accurate classification of benign tumors can prevent patients undergoing unnecessary treatments. Thus, the correct diagnosis of BC and classification of patients into malignant or benign groups is the subject of much research. Because of its unique advantages in critical features detection from complex BC datasets, machine learning (ML) is widely recognized as the methodology of choice in BC pattern classification and forecast modelling.

Classification and data mining methods are an effective way to classify data. Especially in medical field, where those methods are widely used in diagnosis and analysis to make decisions.

Recommended Screening Guidelines:

Mammography. The most important screening test for breast cancer is the mammogram. A mammogram is an X-ray of the breast. It can detect breast cancer up to two years before the tumor can be felt by you or your doctor.

Women age 40–45 or older who are at average risk of breast cancer should have a mammogram once a year.

Women at high risk should have yearly mammograms along with an MRI starting at age

**Some Risk Factors for Breast Cancer**

The following are some of the known risk factors for breast cancer. However, most cases of breast cancer cannot be linked to a specific cause. Talk to your doctor about your specific risk.

**Age:** The chance of getting breast cancer increases as women age. Nearly 80 percent of breast cancers are found in women over the age of 50.

**Personal history of breast cancer:** A woman who has had breast cancer in one breast is at an increased risk of developing cancer in her other breast.

**Family history of breast cancer:** A woman has a higher risk of breast cancer if her mother, sister or daughter had breast cancer, especially at a young age (before 40). Having other relatives with breast cancer may also raise the risk.

**Genetic factors:** Women with certain genetic mutations, including changes to the BRCA1 and BRCA2 genes, are at higher risk of developing breast cancer during their lifetime. Other gene changes may raise breast cancer risk as well.

**Childbearing and menstrual history:** The older a woman is when she has her first child, the greater her risk of breast cancer. Also at higher risk are:

* Women who menstruate for the first time at an early age (before 12)
* Women who go through menopause late (after age 55)
* Women who’ve never had children

**The Data**

The dataset used in this story is publicly available and was created by Dr. William H. Wolberg, physician at the University Of Wisconsin Hospital at Madison, Wisconsin, USA. To create the dataset Dr. Wolberg used fluid samples, taken from patients with solid breast masses and an easy-to-use graphical computer program called Xcyt, which is capable of perform the analysis of cytological features based on a digital scan. The program uses a curve-fitting algorithm, to compute ten features from each one of the cells in the sample, than it calculates the mean value, extreme value and standard error of each feature for the image, returning a 30 real-valuated vector.

Attribute Information:

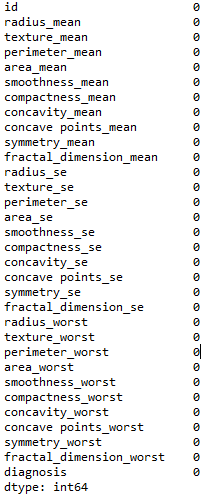
1. ID number 2) Diagnosis (M = malignant, B = benign) 3–32)  
     
   Ten real-valued features are computed for each cell nucleus:
2. radius (mean of distances from center to points on the perimeter)
3. texture (standard deviation of gray-scale values)
4. perimeter
5. area
6. smoothness (local variation in radius lengths)
7. compactness (perimeter² / area — 1.0)
8. concavity (severity of concave portions of the contour)
9. concave points (number of concave portions of the contour)
10. symmetry
11. fractal dimension (“coastline approximation” — 1)



**Fig : Dataset**

We can observe that the data set contain 569 rows and 32 columns. ‘Diagnosis’ is the column which we are going to predict , which says if the cancer is M = malignant or B = benign. 1 means the cancer is malignant and 0 means benign. We can identify that out of the 569 persons, 357 are labeled as B (benign) and 212 as M (malignant).

Lets find out if there are any missing or null data points of the data set (if there is any).



**Fig : Observe missing data**

The 0's show no presence of null values.

**Chapter 2**

**Tools & Technology Used**

**Python**

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

Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.

Python is mainly stated as high-level, general-purpose programming language, which emphasizes code readability. The syntax helps the programmers to express their concepts in few general "lines of code" when compared with other promising languages, like Java or C++. Through our courses, you can easily construct clear programs, on both large and small scales. As the importance of Python programming language is gaining huge popularity, therefore; the need to understand and know the language is increasing among people.

The “Python library” contains several different kinds of components.

It contains data types that would normally be considered part of the “core” of a language, such as numbers and lists. For these types, the Python language core defines the form of literals and places some constraints on their semantics, but does not fully define the semantics. (On the other hand, the language core does define syntactic properties like the spelling and priorities of operators.)

The library also contains built-in functions and exceptions — objects that can be used by all Python code without the need of an [import](https://docs.python.org/3/reference/simple_stmts.html) statement. Some of these are defined by the core language, but many are not essential for the core semantics and are only described here.

The bulk of the library, however, consists of a collection of modules. There are many ways to dissect this collection. Some modules are written in C and built in to the Python interpreter; others are written in Python and imported in source form. Some modules provide interfaces that are highly specific to Python, like printing a stack trace; some provide interfaces that are specific to particular operating systems, such as access to specific hardware; others provide interfaces that are specific to a particular application domain, like the World Wide Web. Some modules are available in all versions and ports of Python; others are only available when the underlying system supports or requires them; yet others are available only when a particular configuration option was chosen at the time when Python was compiled and installed.

This manual is organized “from the inside out:” it first describes the built-in functions, data types and exceptions, and finally the modules, grouped in chapters of related modules.

This means that if you start reading this manual from the start, and skip to the next chapter when you get bored, you will get a reasonable overview of the available modules and application areas that are supported by the Python library. Of course, you don’t *have* to read it like a novel — you can also browse the table of contents (in front of the manual), or look for a specific function, module or term in the index (in the back). And finally, if you enjoy learning about random subjects, you choose a random page number (see module [random](https://docs.python.org/3/library/random.html)) and read a section or two. Regardless of the order in which you read the sections of this manual, it helps to start with chapter [Built-in Functions](https://docs.python.org/3/library/functions.html), as the remainder of the manual assumes familiarity with this material.

The Python Standard Library

While [The Python Language Reference](https://docs.python.org/3/reference/index.html) describes the exact syntax and semantics of the Python language, this library reference manual describes the standard library that is distributed with Python. It also describes some of the optional components that are commonly included in Python distributions.

Python’s standard library is very extensive, offering a wide range of facilities as indicated by the long table of contents listed below. The library contains built-in modules (written in C) that provide access to system functionality such as file I/O that would otherwise be inaccessible to Python programmers, as well as modules written in Python that provide standardized solutions for many problems that occur in everyday programming. Some of these modules are explicitly designed to encourage and enhance the portability of Python programs by abstracting away platform-specifics into platform-neutral APIs.

The Python installers for the Windows platform usually include the entire standard library and often also include many additional components. For Unix-like operating systems Python is normally provided as a collection of packages, so it may be necessary to use the packaging tools provided with the operating system to obtain some or all of the optional components.

In addition to the standard library, there is a growing collection of several thousand components (from individual programs and modules to packages and entire application development frameworks), available from the Python Package Index.

**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 with contributions from several other developers. In 2005, Travis Oliphant created NumPy by incorporating features of the competing Numarray into Numeric, with extensive modifications. NumPy is open-source software and has many contributors.

The Python programming language was not originally designed for numerical computing, but attracted the attention of the scientific and engineering community early on. In 1995 the special interest group (SIG) matrix-sig was founded with the aim of defining an array computing package; among its members was Python designer and maintainer Guido van Rossum, who extended Python's syntax (in particular the indexing syntax) to make array computing easier.

A new package called Numarray was written as a more flexible replacement for Numeric. Like Numeric, it too is now deprecated.Numarray had faster operations for large arrays, but was slower than Numeric on small ones, so for a time both packages were used in parallel for different use cases. The last version of Numeric (v24.2) was released on 11 November 2005, while the last version of numarray (v1.5.2) was released on 24 August 2006.

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. NumPy addresses the slowness problem partly by providing multidimensional arrays and functions and operators that operate efficiently on arrays, requiring rewriting some code, mostly inner loops, using NumPy.

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. In comparison, MATLAB boasts a large number of additional toolboxes, notably Simulink, whereas NumPy is intrinsically integrated with Python, a more modern and complete programming language. Moreover, complementary Python packages are available; SciPy is a library that adds more MATLAB-like functionality and Matplotlib is a plotting package that provides MATLAB-like plotting functionality. Internally, both MATLAB and NumPy rely on BLAS and LAPACK for efficient linear algebra computations.

Python bindings of the widely used computer vision library OpenCV utilize NumPy arrays to store and operate on data. Since images with multiple channels are simply represented as three-dimensional arrays, indexing, slicing or masking with other arrays are very efficient ways to access specific pixels of an image. The NumPy array as universal data structure in OpenCV for images, extracted feature points, filter kernels and many more vastly simplifies the programming workflow and debugging.

**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. The name is derived from the term "panel data", an econometrics term for data sets that include observations over multiple time periods for the same individuals.

Pandas is mainly used for data analysis. Pandas allows importing data from various file formats such as comma-separated values, JSON, SQL, Microsoft Excel. Pandas allows various data manipulation operations such as merging, reshaping, selecting, as well as data cleaning, and data wrangling features.

**Tkinter**

Tkinter is Python's de-facto standard GUI (Graphical User Interface) package. It is a thin object-oriented layer on top of Tcl/Tk.

Tkinter is not the only GuiProgramming toolkit for Python. It is however the most commonly used one. CameronLaird calls the yearly decision to keep TkInter "one of the minor traditions of the Python world.

Tkinter is a Python binding to the Tk GUI toolkit. It is the standard Python interface to the Tk GUI toolkit, and is Python's de facto standard GUI. Tkinter is included with standard Linux, Microsoft Windows and Mac OS X installs of Python. The name Tkinter comes from Tk interface.

The generic term for any of the building blocks that make up an application in a graphical user interface.

Core widgets: The containers: frame, labelframe, toplevel, paned window. The buttons: button, radiobutton, checkbutton (checkbox), and menubutton. The text widgets: label, message, text. The entry widgets: scale, scrollbar, listbox, slider, spinbox, entry (singleline), optionmenu, text (multiline), and canvas (vector and pixel graphics).

Tkinter provides three modules that allow pop-up dialogs to be displayed: tk.messagebox (confirmation, information, warning and error dialogs), tk.filedialog (single file, multiple file and directory selection dialogs) and tk.colorchooser (colour picker).

Python 2.7 and Python 3.1 incorporate the "themed Tk" ("ttk") functionality of Tk This allows Tk widgets to be easily themed to look like the native desktop environment in which the application is running, thereby addressing a long-standing criticism of Tk (and hence of Tkinter). Some widgets are exclusive to ttk, such as the combobox, progressbar and treeview widgets

**JupyterLab 1.0: Jupyter’s Next-Generation Notebook Interface**

JupyterLab is a web-based interactive development environment for Jupyter notebooks, code, and data. JupyterLab is flexible: configure and arrange the user interface to support a wide range of workflows in data science, scientific computing, and machine learning. JupyterLab is extensible and modular: write plugins that add new components and integrate with existing ones.

**The Jupyter Notebook**

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

**Sklearn**

Scikit-learn (formerly scikits.learn and also known as sklearn) 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. The original codebase was later rewritten by other developers. In 2010 Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort and Vincent Michel, all from the French Institute for Research in Computer Science and Automation in Rocquencourt, France, took leadership of the project and made the first public release on February the 1st 2010. Of the various scikits, scikit-learn as well as scikit-image were described as "well-maintained and popular" in November 2012. Scikit-learn is one of the most popular machine learning libraries on GitHub.

Scikit-learn is largely written in Python, and uses numpy extensively for high-performance linear algebra and array operations. Furthermore, some core algorithms are written in Cython to improve performance. Support vector machines are implemented by a Cython wrapper around LIBSVM; logistic regression and linear support vector machines by a similar wrapper around LIBLINEAR. In such cases, extending these methods with Python may not be possible.

Scikit-learn integrates well with many other Python libraries, such as matplotlib and plotly for plotting, numpy for array vectorization, pandas dataframes, scipy, and many more.

**Chapter 3**

**Data Visualization**

**Identify the problem**

Breast cancer is the most common malignancy among women, accounting for nearly 1 in 3 cancers diagnosed among women in the United States, and it is the second leading cause of cancer death among women. Breast Cancer occurs as a results of abnormal growth of cells in the breast tissue, commonly referred to as a Tumor. A tumor does not mean cancer - tumors can be benign (not cancerous), pre-malignant (pre-cancerous), or malignant (cancerous). Tests such as MRI, mammogram, ultrasound and biopsy are commonly used to diagnose breast cancer performed.

**Expected outcome**

Given breast cancer results from breast fine needle aspiration (FNA) test (is a quick and simple procedure to perform, which removes some fluid or cells from a breast lesion or cyst (a lump, sore or swelling) with a fine needle similar to a blood sample needle). Since this build a model that can classify a breast cancer tumor using two training classification:

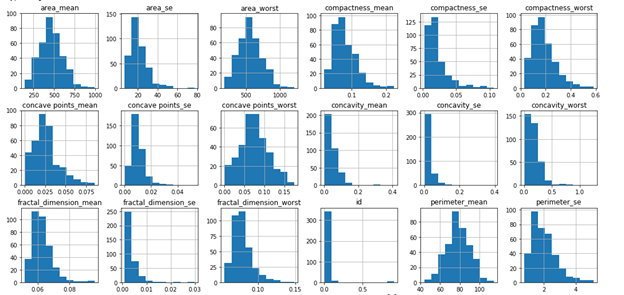
1. 1= Malignant (Cancerous) - Present
2. 0= Benign (Not Cancerous) -Absent

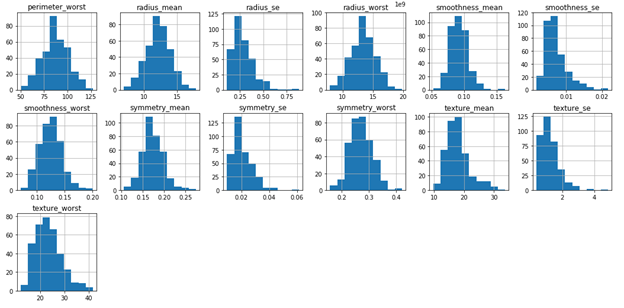
**Identify data sources**

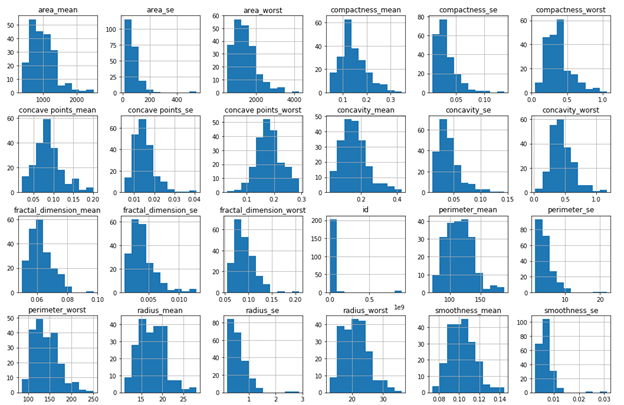
The Breast Cancer datasets is available machine learning repository maintained by the University of California, Irvine. The dataset contains 569 samples of malignant and benign tumor cells.

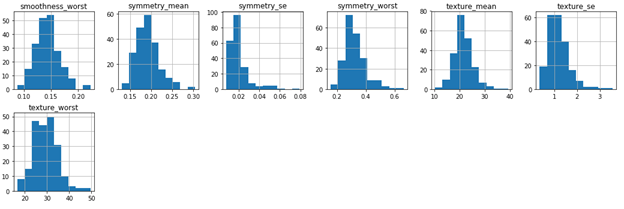
* The first two columns in the dataset store the unique ID numbers of the samples and the corresponding diagnosis (M=malignant, B=benign), respectively.
* The columns 3-32 contain 30 real-value features that have been computed from digitized images of the cell nuclei, which can be used to build a model to predict whether a tumor is benign or malignant.

Visualization of data is an imperative aspect of data science. It helps to understand data and also to explain the data to another person. Python has several interesting visualization libraries such as Matplotlib, Seaborn etc.









**Fig : Visualization of Dataset**

**Chapter 4**

**Predictive model using Support Vector Machine (SVM)**

**Supervised learning**

Supervised learning is the machine learning task of learning a function that maps an input to an output based on example input-output pairs. It infers a function from labeled training data consisting 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 (also called the supervisory signal). A supervised learning algorithm analyzes 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

**Predictive model using Support Vector Machine (SVM)**

Support vector machines (SVMs) learning algorithm will be used to build the predictive model. SVMs are one of the most popular classification algorithms, and have an elegant way of transforming nonlinear data so that one can use a linear algorithm to fit a linear model to the data (Cortes and Vapnik 1995)

Kernelized support vector machines are powerful models and perform well on a variety of datasets.

1. SVMs allow for complex decision boundaries, even if the data has only a few features.
2. They work well on low-dimensional and high-dimensional data (i.e., few and many features), but don’t scale very well with the number of samples.
3. SVMs requires careful preprocessing of the data and tuning of the parameters. This is why, these days, most people instead use tree-based models such as random forests or gradient boosting (which require little or no preprocessing) in many applications.
4. SVM models are hard to inspect; it can be difficult to understand why a particular prediction was made, and it might be tricky to explain the model to a nonexpert.

Running an SVM on data with up to 10,000 samples might work well, but working with datasets of size 100,000 or more can become challenging in terms of runtime and memory usage.

**Divide records in training and testing sets**

X\_train, X\_test, y\_train, y\_test = train\_test\_split(Xs, y, test\_size=0.3, random\_state=2, stratify=y)

**Creating our model**

clf = SVC(probability=True)

**Training our model with our training data**

model = clf.fit(X\_train, y\_train)

X\_train = ARRAY containing all the data to train our model.

y\_train = ARRAY that tells our model which cell contains data of person diagnosed with Breast Cancer

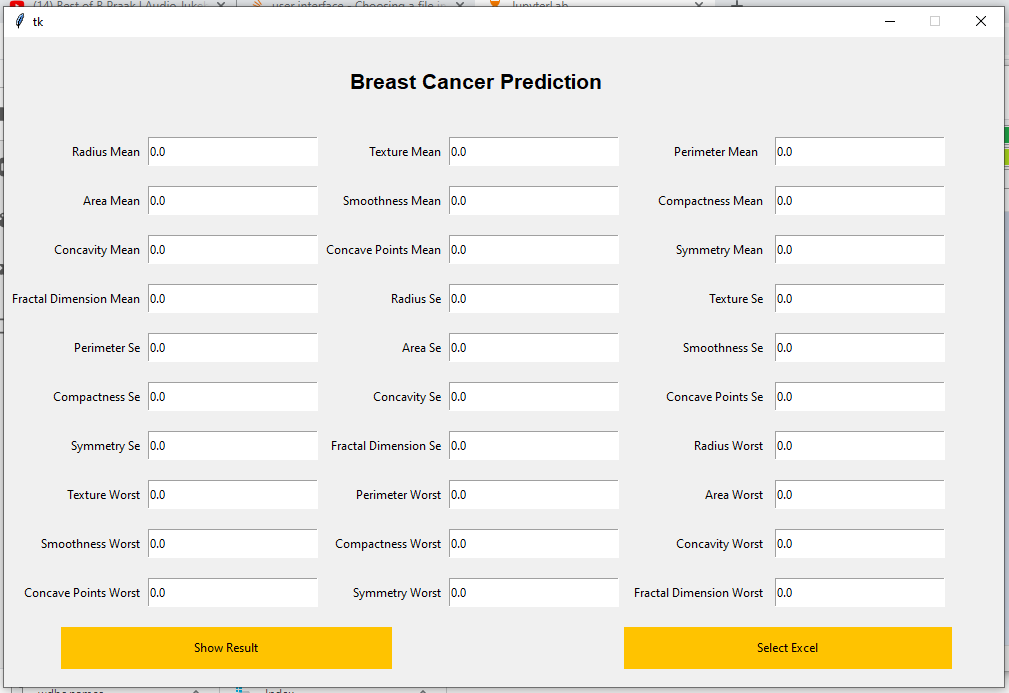
**Making Predictions using our model**

y\_pred = model.predict(X\_test)

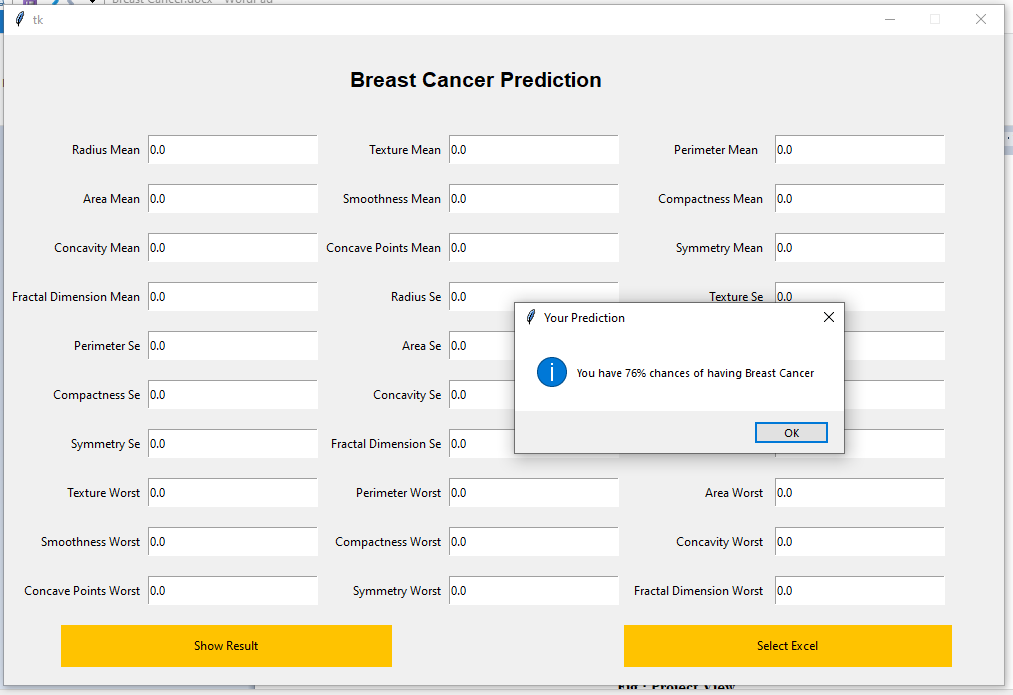
X\_test = ARRAY containing input data to gain output

**Chapter 5**

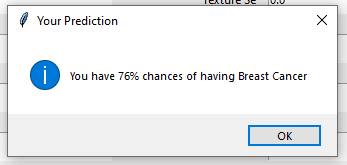
**Project Snapshots**



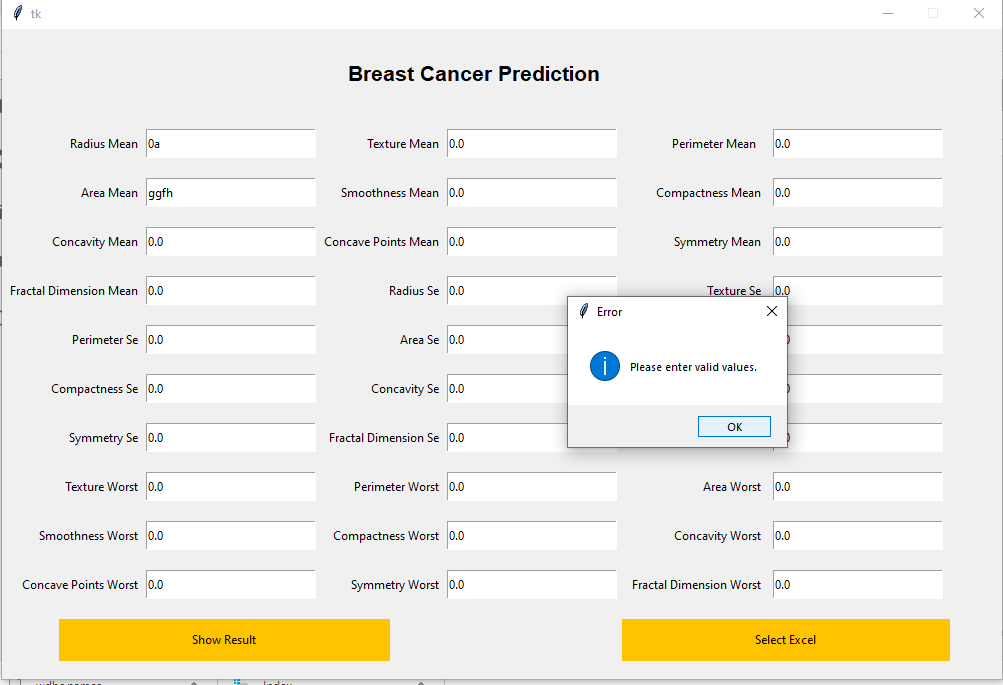
**Fig : Project View**



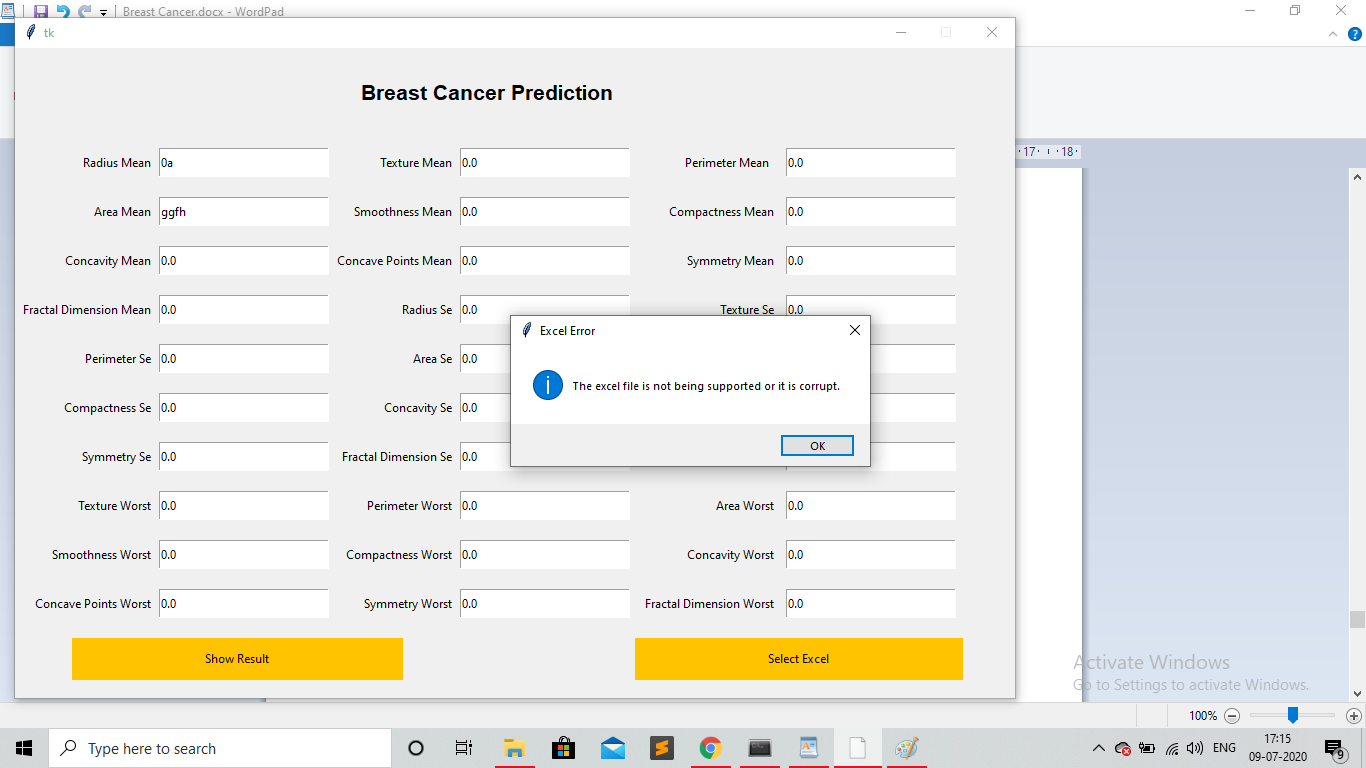
**Fig : Project View**



**Fig : Project Prediction message**

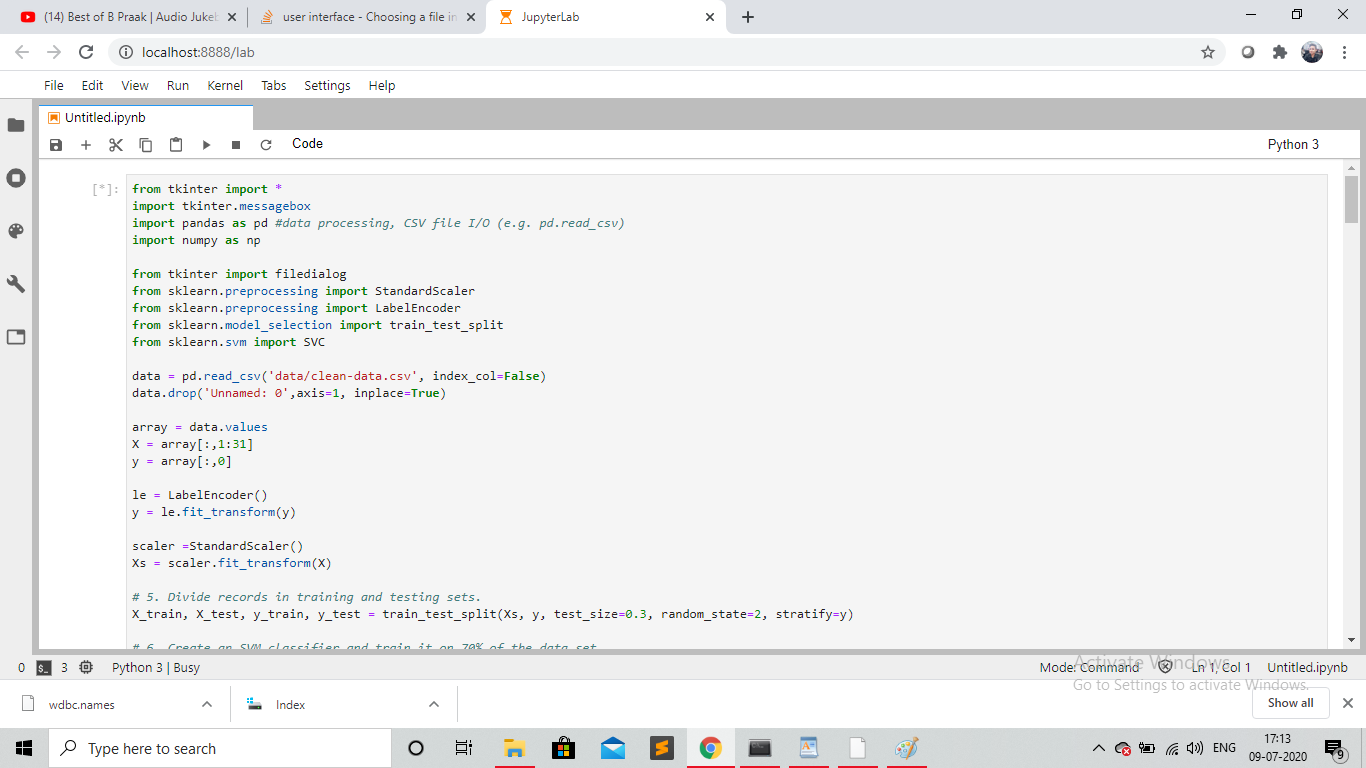


**Fig : Error Message 1**



**Fig : Error Message 2**

**The Project Code**



**Fig : View from project source code**

**Chapter 6**

**The Conclusion**

Breast cancer is cancer that forms in the cells of the breasts.

After skin cancer, breast cancer is the most common cancer diagnosed in women in the United States. Breast cancer can occur in both men and women, but it's far more common in women.

Substantial support for breast cancer awareness and research funding has helped created advances in the diagnosis and treatment of breast cancer. Breast cancer survival rates have increased, and the number of deaths associated with this disease is steadily declining, largely due to factors such as earlier detection, a new personalized approach to treatment and a better understanding of the disease.

It is difficult to manually determine the odds of getting breast cancer based on risk factors. However, machine learning techniques are useful to predict the output from existing data.

We have successfully used our dataset to predict whether a person may have Breast Cancer.

**References**

[1] <https://www.programiz.com/python-programming>

[2] [www.w3schools.com](http://www.w3schools.com/)

[3] wikipedia

[4] <https://pandas.pydata.org/pandas-docs/stable/user_guide/io.html>

[5] <https://wiki.python.org/moin/TkInter>

[6] <https://www.mayoclinic.org/diseases-conditions/breast-cancer/symptoms-causes/syc-20352470>