**Put your college Logo, College Name, Project Name, Guide Name in this page**

**ABSTRACT**

**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 world. 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 create 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.

This report presents a machine learning approach for predicting breast cancer risk in patients. The dataset used for this study contains various clinical, demographic, and behavioral attributes of patients, as well as their respective medical image data. The objective of this study is to build an accurate prediction model that can assist in identifying patients at high risk of developing Breast Cancer in advance.

Various machine learning core algorithms were applied to the dataset, including logistic regression, decision tree, random forest, and support vector machines. Feature selection and hyper parameter tuning techniques were also employed to optimize the performance of the models.

The results of the study demonstrate that the **random forest algorithm** achieved the highest accuracy of 99 % in predicting the risk of Breast Cancer. Overall, the study highlights the potential of machine learning in predicting Breast Cancer risk, which can aid in early detection and prevention of Tumour related diseases. The findings of this report may be useful in improving clinical decision-making and developing targeted interventions for patients at high risk of developing Cancer.

**CONTENTS**

**Table of Contents Page No**

1. **INTRODUCTION**

1.1 Introduction to the project 5

1. **SYSTEM ANALYSIS 9**
   1. Existing system 10
   2. Proposed System 12
   3. Context Diagram 14
   4. Feasibility Study 15
      1. Economic Feasibility 15
      2. Operational Feasibility 16
      3. Technical Feasibility 16
2. **SYSTEM SPECIFICATION 17** 
   1. System Requirements 17
   2. Hardware Requirements 17
   3. Software Requirements 17
   4. Technologies used 18
      1. Tableau 18
      2. MySQL Community Edition 18
      3. Google Colab 19
   5. Algorithms used 19
3. **SYSTEM DESIGN 20**
   1. System Design Methodology 20
   2. Input Design 20
   3. Output Design 21
   4. Data Flow Diagram 24
   5. Database Design 37
4. **SYSTEM DEVELOPMENT 47**
5. **GANTT CHART 48**
6. **SYSTEM TESTING 49**
   1. Unit Testing 49
   2. Module Testing 49
   3. Integration Testing 50
   4. Validation Testing 50
   5. Black Box testing 50
   6. White Box testing 51
7. **SYSTEM IMPLEMENTATION 52**
8. **APPENDIX 54**
   1. Program Code 54
   2. Screenshots 80
9. **CONCLUSION 85**
10. **ANNEXURE 86**
11. **WEBLIOGRAPH 87**
12. **INTRODUCTION**

**1.1 INTRODUCTION TO THE PROJECT**

“Breast Cancer PREDICTION” is a supervised machine learning system developed for predicting cancer disease risk in patients by utilizing the clinical, demographic and medical image of patients. as well as their respective MRI details. The system utilizes historical data for implementation.

Upon the machine learning algorithms applied such as Logistic Regression, Random Forest and Decision Tree, **Random Forest showed an accuracy of 99%** in predicting the Breast Cancer risk in patients.

The model is successfully built with Google Colab Cloud Platform where users can input data as a bulk in Google sheets, Excel sheets or an individual input of result of a single patient using a Google GUI Form.

The objective of this project is to develop a Breast Cancer prediction system that utilizes machine learning algorithms to provide efficient prediction of cancer disease risk in patients.

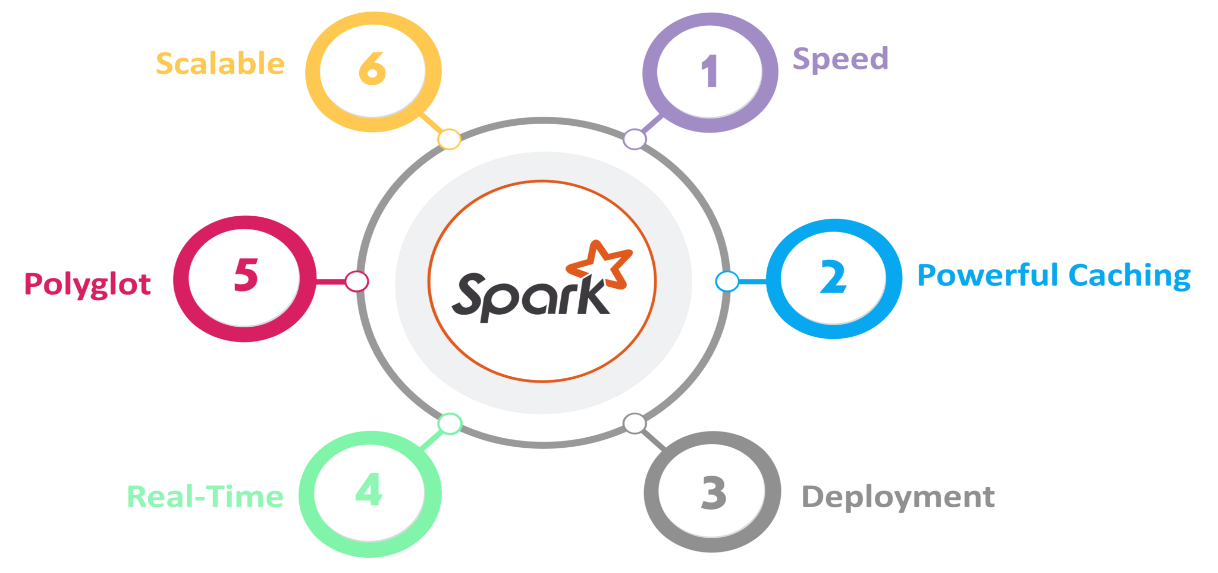
The project also aims to evaluate the performance of the system through a pilot study, with the goal of achieving a higher accuracy rate in predicting cancer disease risk and improving the overall efficiency of clinical decision making.

**Understanding the functionalities of the Technologies used in this Project**

**The core programming technology is Python-Spark with Cloud Computing.**

Apache Spark is an open source cluster computing framework for real-time data processing. The main feature of Apache Spark is its in-memory cluster computing that increases the processing speed of an application. Spark provides an interface for programming entire clusters with implicit data parallelism and fault tolerance. It is designed to cover a wide range of workloads such as batch applications, iterative algorithms, interactive queries, and streaming.

**Features of Apache Spark:**



1. **Speed**Spark runs up to 100 times faster than Hadoop MapReduce for large-scale data processing. It is also able to achieve this speed through controlled partitioning.
2. **Powerful caching**Simple programming layer provides powerful caching and disk persistence capabilities.
3. **Deployment**This AI-ML model can be deployed in any Operating Systems like Windows, Linux, Android and iOS without any changes in codes.
4. **Scalable :** Can Process huge datasets, Size no limit

**Spark Architecture Overview**

Apache Spark has a well-defined layered architecture where all the spark components and layers are loosely coupled. This architecture is further integrated with various extensions and libraries. Apache Spark Architecture is based on two main abstractions:

* Resilient Distributed Dataset (RDD)
* Directed Acyclic Graph (DAG)

**Resilient Distributed Dataset (RDD)**

RDDs are the building blocks of any Spark application. RDDs Stands for:

**Resilient:** Fault tolerant and is capable of rebuilding data on failure

**Distributed:** Distributed data among the multiple nodes in a cluster

**Dataset:** Collection of partitioned data with values

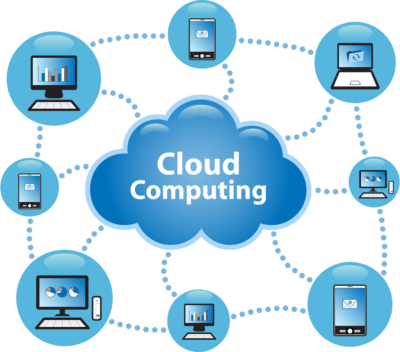
Moreover, once you create an RDD it becomes **immutable**. By immutable, an object whose state cannot be modified after it is created, but they can surely be **transformed.**

Talking about the distributed environment, each dataset in RDD is divided into logical partitions, which may be computed on different nodes of the cluster. Due to this, you can perform transformations or actions on the complete data parallelism. Also, you don’t have to worry about the distribution, because Spark takes care of that.

**Cluster**

**Cluster**: It means that multiple servers are grouped together to achieve the same business and can be regarded as one computer.

A group of computers consisting of multiple servers, as a whole, provides users with a set of network resources, which are the nodes of the cluster.



**Cloud Computing**

Sharing resources such as hardware, software, and data is one of the principles of cloud computing with different levels of openness to the software and concurrency, it’s easier to process data simultaneously through multiple processors. The more fault-tolerant an application is, the more quickly it can recover from a system failure.

**VECTORS**

The use of vectors in machine learning draws on both mathematical and statistical concepts, and it is an interdisciplinary field that blends both mathematics and statistics. In mathematics Vectors are basically originated from **Linear Algebra**

For example, in supervised learning, vectors are often used to represent feature values of data instances, and statistical methods such as mean and covariance are used to summarize the distribution of these feature vectors. In unsupervised learning, vectors are used to represent the data itself, and statistical methods such as clustering and dimensionality reduction are used to uncover hidden structure in the data.

Citation : <https://machinelearningmastery.com/examples-of-linear-algebra-in-machine-learning/>

**Fit and Transform**

In Spark machine learning, the terms "fit" and "transform" are used to describe the two main stages in the training and prediction process of a machine learning model.

**Steps involved in Breast Cancer Prediction Model – Supervised Learning**

**The dataset contains the following information: -**

1. id - Patient id,

2. diagnosis - 'M' (Malignant=Cancerous, Benign - Non Cancerous )

3. radius\_mean = radius of the tumor

4. texture\_mean = variation in grayscale levels within cancer affected area

5. perimeter\_mean = Circumference value of the tumour

6. area\_mean - total area enclosed by the tumor boundary

7. smoothness\_mean - variations in radius lengths of internal tumour cells

8. compactness\_mean - how closely the tumor area is to its perimeter (spreding of - tumour cells)

9. concavity\_mean - average concave portions of the shape in tumour cells

10. concave\_points\_mean": It gives the avg number of concave points on the tumour boundary.

11. symmetry\_mean - which measures how symmetric or asymmetric the tumor shape is.

12. fractal\_dimension\_mean - Complexity of the tumour's perimeter (or ratio)

**The formula for standard error is = ​Standard Deviation / sqrt(Number of Measurements)**

**(\_se = Standard Error)**

13. radius\_se : screening mammograms miss about 1 in 8 breast cancers.

: Women with dense breasts are more likely to get false-negative

: results. (\_se = Standard Error)

14. texture\_se : standard error of texture within the tumor region(missing grayscale)

15. perimeter\_se : This column gives the standard error of the tumor perimeter

16. area\_se: It represents the standard error of the tumor area.

17. smoothness\_se - Standard error of smoothness

18. compactness\_se :represents the standard error of compactness.

19. concavity\_se: This column provides the standard error of concavity.

20. concave points\_se: standard error of the number of concave points

21. symmetry\_se": This column gives the standard error of symmetry.

22. fractal\_dimension\_se: It provides the standard error of fractal dimension

**Largest or worst values** data

23. radius\_worst : the worst (largest) value of the tumor radius among several measurements.

24. texture\_worst : largest value

25. perimeter\_worst : largest value

26. area\_worst : largest value of tumour area

27. smoothness\_worst :largest value

28. compactness\_worst :largest value

29. concavity\_worst : largest value

30. concave points\_worst : largest value

31. symmetry\_worst : largest value

32. fractal\_dimension\_worst : largest value (fractal = ratio)

3. In the original dataset of Breast Cancer Data there are,

* Total number of patients = 20,000
* Total columns = 32

Since AI algorithms can only understand vectors, the above columns should be converted to vectors using an instance of **Vector Assembler.**

The vector assembler in machine learning is particularly useful for combining features that are numerical or categorical in nature, as it allows the machine learning algorithm to process these features as a single entity. This can improve the accuracy of the machine learning model and reduce the amount of time and effort required for feature engineering.

The **transform** command converts the feature columns into a single array using the dataset **‘label\_df’.**

4. Similarly convert the **label** column to vector using an instance of **String Indexer** algorithm.

The string indexer works by assigning a numerical index to each unique string value in the categorical column. The most frequent string value is assigned index 0, the second most frequent string value is assigned index 1, and so on. This process is called **encoding or mapping**, and it creates a mapping between the string values and their corresponding numerical indices.

5. **Split train and test data:** After vectorization use pyspark randomize technique to split the original data into training and testing. In this project the split ratio is 70 % for training and 30 % for testing. A statistical value **5043 is used as seed** in **random Split** operation to fix the 70-30 ratio value constantly without any change in each run.

**Core Algorithms Used in this Project**

**Random Forest**

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of **ensemble learning,** which is a process of *combining multiple classifiers to solve a complex problem and to improve the performance of the model.*

As the name suggests, **"Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset."** Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

**The greater number of trees in the forest leads to higher accuracy and prevents the problem of over fitting.**

**Example:**Suppose there is a dataset that contains multiple fruit images. So, this dataset is given to the Random Forest classifier. The dataset is divided into subsets and given to each decision tree. During the training phase, each decision tree produces a prediction result, and when a new data point occurs, then based on the majority of results, the Random Forest classifier predicts the final decision. Consider the below image:



In this project the random forest is declared as follows:-

rf = RandomForestClassifier() \

.setImpurity("entropy") \

.setNumTrees(20) \

.setMaxDepth(20) \

.setFeatureSubsetStrategy("all") \

.setSeed(5043) # Used to get fixed values in each run

The **entropy** index, **or entropy** coefficient, or **impurity** computes the degree of probability of a specific variable that is wrongly being classified when chosen randomly and a variation of the entropy coefficient. It works on categorical variables, provides outcomes either be **“successful” or “failure” or** hence conducts binary splitting only.

7. **How to test accuracy of your model.**

evaluator = MulticlassClassificationEvaluator() \  
 .setLabelCol ("label") \  
 .setPredictionCol("prediction") \  
 .setMetricName("weightedPrecision")

The multi-class classification evaluator works by comparing the predicted class labels generated by the model to the actual class labels in the test dataset. It calculates various metrics to evaluate the performance of the model, such as precision, recall, F1 score, and weightedPrecision.

**This project has got an accuracy of *99 percent using Random Forest***

8. Save your model for predicting results using actual end user data.

**Logistic Regression Algorithm**

Logistic regression is a supervised machine learning algorithm that accomplishes binary classification tasks by predicting the probability of an outcome, event, or observation. The model delivers a binary or dichotomous outcome limited to two possible outcomes: yes/no, 0/1, or true/false.

Logical regression analyzes the relationship between one or more independent variables and classifies data into discrete classes. It is extensively used in predictive modeling, where the model estimates the mathematical probability of whether an instance belongs to a specific category or not.

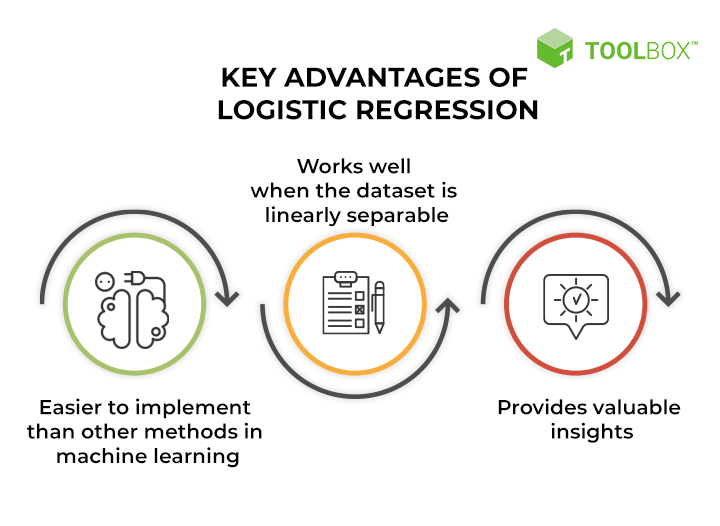
For example, 0 – represents a negative class; 1 – represents a positive class. Logistic regression is commonly used in binary classification problems where the outcome variable reveals either of the two categories (0 and

**Some examples of such classifications and instances where the binary response is expected or implied are:**

1. Determine the probability of heart attacks: With the help of a logistic model, medical practitioners can determine the relationship between variables such as the weight, exercise, etc., of an individual and use it to predict whether the person will suffer from a heart attack or any other medical complication.

2. Possibility of enrolling into a university: Application aggregators can determine the probability of a student getting accepted to a particular university or a degree course in a college by studying the relationship between the estimator variables, such as GRE, GMAT, or TOEFL scores.

3. Identifying spam emails: Email inboxes are filtered to determine if the email communication is promotional/spam by understanding the predictor variables and applying a logistic regression algorithm to check its authenticity.



1. Easier to implement machine learning methods: A machine learning model can be effectively set up with the help of training and testing. The training identifies patterns in the input data (image) and associates them with some form of output (label). Training a logistic model with a regression algorithm does not demand higher computational power. As such, logistic regression is easier to implement, interpret, and train than other ML methods.

2. Suitable for linearly separable datasets: A linearly separable dataset refers to a graph where a straight line separates the two data classes. In logistic regression, the y variable takes only two values. Hence, one can effectively classify data into two separate classes if linearly separable data is used.

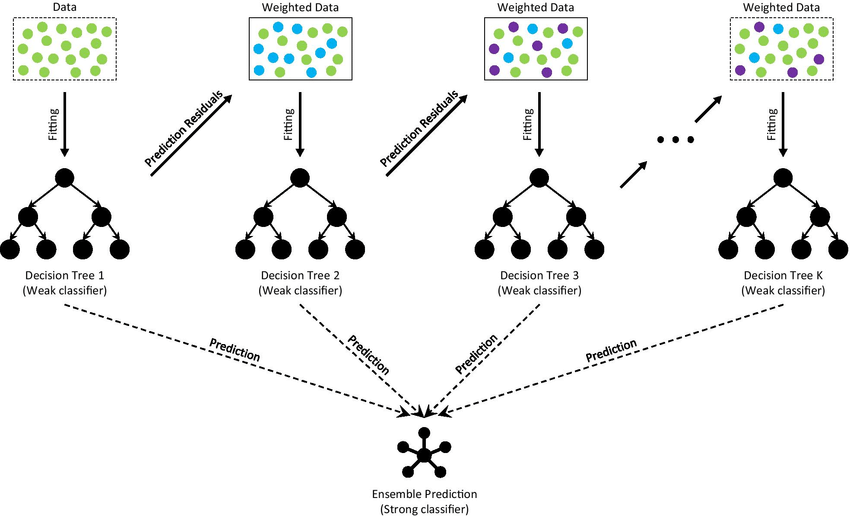
3. Provides valuable insights: Logistic regression measures how relevant or appropriate an independent/predictor variable is (coefficient size) and also reveals the direction of their relationship or association (positive or negative).

**Logistic regression analysis yields reliable, robust, and valid results when a larger sample size of the dataset is considered.**

**Gradient Boosted Trees**

Gradient Boosting is a machine learning algorithm, used for both classification and regression problems. It works on the principle that many weak learners (eg: shallow trees) can together make a more accurate predictor.

Boosting works on the principle of improving mistakes of the previous learner through the next learner. In boosting, weak learner are used which perform only slightly better than a random chance. Boosting focuses on sequentially adding up these weak learners and filtering out the observations that a learner gets correct at every step. Basically the stress is on developing new weak learners to handle the remaining difficult observations at each step.



Gradient Boosting is a powerful boosting algorithm that combines several weak learners into strong learners, in which each new model is trained to minimize the loss function such as mean squared error or cross-entropy of the previous model using gradient descent. In each iteration, the algorithm computes the gradient of the loss function with respect to the predictions of the current ensemble and then trains a new weak model to minimize this gradient. The predictions of the new model are then added to the ensemble, and the process is repeated until a stopping criterion is met.

**2. SYSTEM ANALYSIS**

System analysis refers to an orderly structured process for identifying and solving problems using computer. It is the most essential part of the project development. It is the process of gathering and interpreting facts, diagnosing problems and using the information to recommend improvements to the system. The system is viewed as a whole and inputs to the system are in defined the output from the system are traced through the various data are collected on available files. Description points and transaction held by the present system. Based on analysis, a cost or benefit analysis. To analyze a system one has to study the system work in detail, before designing the appropriate computer based system analysis specifies what the system should do. System analysis includes the following steps:

* Requirement Specification
* Preliminary Investigation
* Feasibility Study
* Analysis
* Design
* Implementation
* Post Implementation and Maintenance

**2.1 EXISTING SYSTEM**

**Description of the existing system**

1. In India ICT (Intelligent Clinical Technology) is used only in Appolo Hospital Group with the help of Google, other hospitals or individuals will not get access to this system.
2. ICT Model works in dedicated Cloud based platforms like AWS S3, Google Cloud and so on
3. The above ML Models are totally owned by respective companies and won’t be accessible by others.
4. In western and European countries perfect Artificial Intelligence backed diagnosis systems are in place. These models can analyze medical images, patient records, lab reports, and other health-related data.
5. ICT technologies are highly expensive and harder to maintain. Only rich countries can afford the infrastructure costs.

**Limitations of the existing system**

1. Major disadvantage of existing AI based health care systems are the hardware requirements. These models will work only in a complex Distributed File Systems, in other words strong cloud computing platforms are inevitable.
2. An individual clinic or small hospitals could not access the present ICT models available because of high price tag.
3. You cannot customize the AI software to suit your business needs.
4. End users are not allowed to change the core algorithms implemented in the health care system model.
5. Separate models are needed for Windows and other operating systems. Mobile versions incurred more price tag. That is you cannot use one model to run in different operating systems.

**2.2 PROPOSED SYSTEM**

**Advantages**

1. The end user will get outputs immediately after entering clinical details.
2. The execution time will be less than one or two minutes.
3. There is no need to upgrade your existing hardware or operating system to get results.
4. The entire codes used in this software is compatible with Cloud computing platforms like Google Colab,AWS-S3 Buckets and other popular Cloud computing platforms.
5. Absolutely there is no limit in total number of users that can be processed using this model. This is achieved by Distributed File System concepts implemented in this model using SPARK and PYTHON API’s.
6. The biggest advantage of this model is, without altering a single line of code this software can be executed in **Windows OS, Linux-Unix OS, Android and iPhones.** That is this model can be used as a Web Application or Mobile Application or in a Local Area Net Work application.

**Programming Technology Features**

* The Cross platform compatibility is achieved by Python which is running on top of Java Virtual Machine. The speed of execution is achieved by the Artificial Intelligent tool **PYSPARK** which is running on top of the popular distributed file system **HADOOP Ecosystem and YARN Resource manager.**
* This AI model will automatically integrate new data along with existing dataset. This results in Reimbursed learning with new data.
* The salient feature is that the user can update the model with new data without incurring any additional cost. This model is designed and developed using the top down approach methodology technically termed as **‘Cascade Methodology’**. Entire software is divided into **Python functions** and each function is called using appropriate Menus. This eliminates duplication of codes again and again.
* The login program is designed in such a way that it incorporate all the security features laid down by Google Gmail Account. Using this the user can recover forgotten passwords or user id’s, they can create new users to share the model. When user execute the **login program** control is transferred to Gmail Authentication Dialogue window.
* The end user can inject new data with the help of a text editor or using excel worksheets.
* All instances of classifiers (Random Forest) and Evaluators (Accuracy Testing) are declared using **Object Oriented Concepts** laid down by python.

**2.4 FEASIBILTY STUDY**

Feasibility Study is conducted to determine whether the proposed system is feasible. Now it is running in manual system, in which all processing is done manually. Lots of stress and manpower is needed here. Implementing the proposed system can eliminate failures of the current system which saves money and time. Thus the proposed system is economically feasible. The new system can support in the later versions of windows operating system. So the proposed system is technically feasible. The new system is also behaviorally feasible as the person who is currently doing the student detail process can reduce his works by implementing the proposed system and the state of his/her job will not change.

#### **2.4.1 ECONOMIC FEASIBILITY**

Economic feasibility aims to balance the costs of designing and implementing a new system against the benefits of having the new system in place. This feasibility research provides senior management with an economic justification for the new system.

A simple economic analysis which gives the actual comparison of costs and benefits are much more meaningful in this case. In addition, this proves to be a useful point of reference to compare actual costs as the project progresses. There could be various types of intangible benefits on account of automation. These could include increased customer satisfaction, improvement in services quality better decision making timeliness of information, expediting activities, improved accuracy of operations, better documentation and record keeping, faster retrieval of information, better employee morale.

**2.4.2 OPERATIONAL FEASIBILITY**

Proposed project is beneficial only if it can be turned into information systems that will meet to the organizations operating requirements. Simply stated, this test of feasibility ask if the system will work when it is developed and installed. Here are questions that will help test the operational feasibility of a project:

Is there sufficient support for the project from management from users? If the current system is well liked and used to the extent that persons will not be able to see reasons for change, there may be resistance are the current business methods acceptable to the user. If they are not, Users may welcome a change that will bring about a more operational and useful systems. Have the user been involved in the planning and development of the project.

Early involvement reduces the chances of resistance to the system and in general and increases the likelihood of successful project. Since the proposed system was to help reduce the hardships encountered. In the existing manual system, the new system was considered to be operational feasible.

#### **2.4.3 TECHNICAL FEASIBILITY**

Evaluating the technical feasibility is the trickiest part of a feasibility. This is because, at this point of time, not too many detailed design of the system, making it difficult to access issues like performance, cost on (account of the kind of technology to be deployed) etc. This model provides easy and efficient access of the details of child including their considered while doing a technical analysis.

Understanding the different technologies involved in the proposed system before commencing the project we have to be very clear about what are the technologies that are to be required for the development of the new system. Find out whether the organization currently possesses the required technologies.

**3. SYSTEM SPECIFICATION**

System specification specifies the hardware and software configuration of the new proposed system. To develop applications software, we use different type of the new proposed system. To develop application software we use different type of software. The software for the development has been selected based on several factors such as

* Support and stability
* Cost of effectiveness
* Development speed
* Ability or create robust application least time

**3.1 SYSTEM REQUIREMENTS**

The system requirements of the project include

* Hardware specification
* Software specification

**3.2 HARDWARE REQUIREMENTS**

* Processor - i3 or above
* Memory - 8 GB RAM
* Hard Disk - 256 GB HDD

**3.3 SOFTWARE REQUIREMENTS**

* PYTHON-SPARK Version 3.9 with full Apache AI Libraries
* PANDAS & KERAS Version above 3.xx Release 2021
* MATPLOTLIB version 3.3.1
* GOOGLE COLAB – Cloud Platform used for Final Coding and AI Model Development

**4. SYSTEM DESIGN**

**4.1. SYSTEM DESIGN METHODOLOGY**

System design focuses on the final system and the process by which it is developed. It leads to transition from a user - oriented document which means system proposal a programmer - oriented document. The design method followed for this project is the bottom -up design strategy in this approach, the basic sets of elements are individual modules. Each module is developed individually as a separate project and adds the modules into this as a reference if necessary. The benefit of the design is that it allows the review of the modules during the system development process.

Design is the first development phase for any engineered product or system. System design is a process of evaluating alternative solution, evaluating the choice of following up the specification for the chosen alternative. System design is to improve the existing system or design a new system and implement the system with improved facilities.

**4.2 INPUT DESIGN**

Input design is the process of converting the user-designed input to a computerized format. Input data are collected and organized into group of similar data. Error entered by the user can be controlled by the input design. The goal of designing the input data is to make data entry easy, logical and free from errors as much possible. Input consists of developing specifications and procedures for the data preparation, those steps necessary to put transaction data into a usable form of processing and data entry.

## **4.3 OUTPUT DESIGN**

Designing the computer output should proceed in an organized manner. The right output must develop while ensuring that each output elements is designed so that people will find system easy to use effectively. When analyst design computer output, they should;

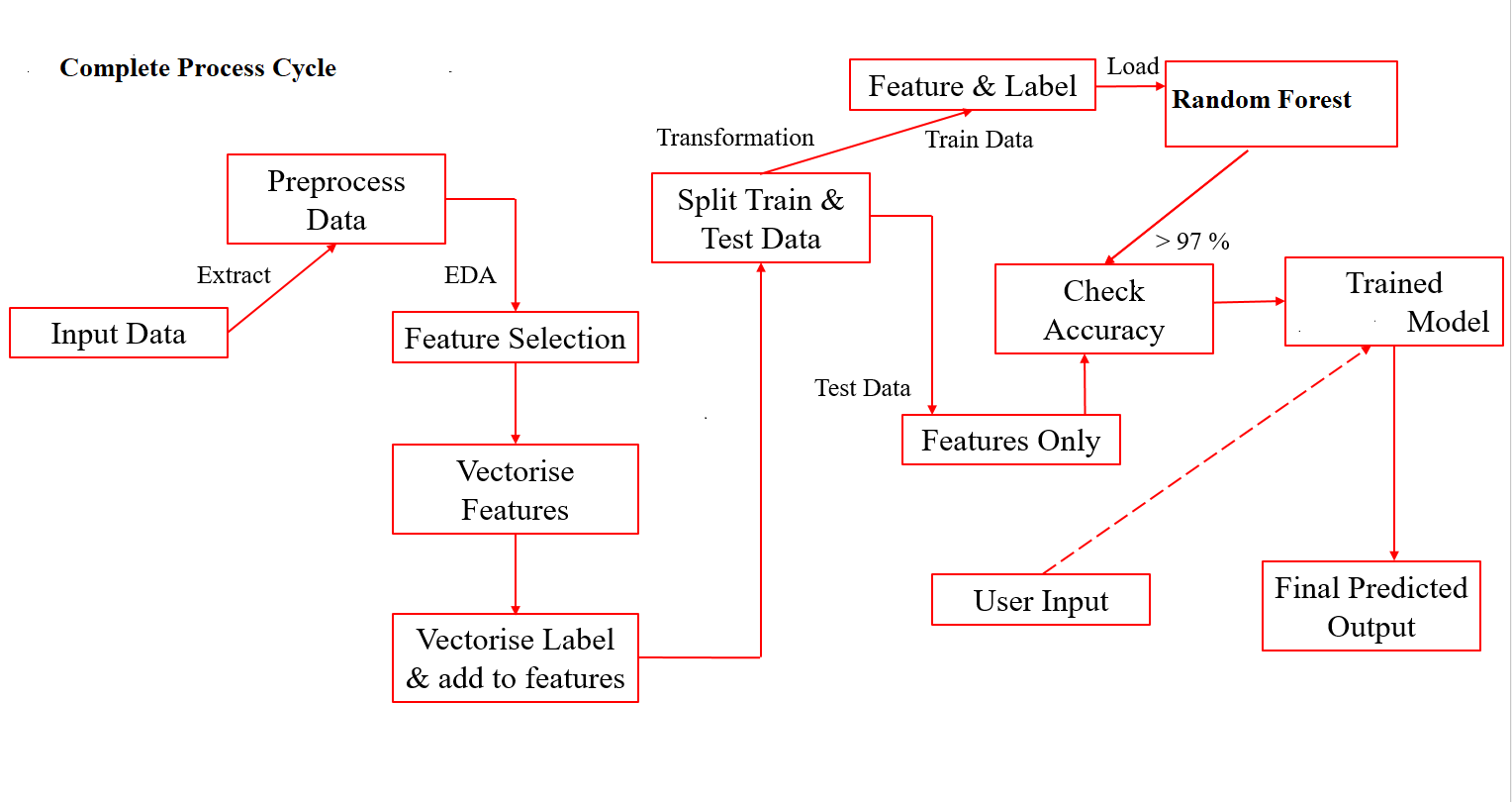
* Identity the specific output that needed to meet the requirements.
* Select methods for presenting information.

Create document, report or order that information produced by the system.

**CONTEXT DIAGRAM**

In software engineering, a context diagram is a high-level visual representation of the system being developed that shows the system in relation to its external entities. It is a type of diagram that provides an overview of the system and the context in which it operates.

Context diagrams are used during the requirements analysis and design phases of software development projects to help end users understand the scope and context of the system being developed. They can also be used to identify the system's external dependencies and the data or information flows between the system and its external entities.

****

## **4.4 DATA FLOW DIAGRAM**

A DFD is a graphical representation of the flow of dots through an information system, its process aspects. A DFD is often used as a preliminary step to create an overview of the system without going into detail, which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design).

A DFD shows what kind of information will be input to and output from the system, how the data will advance through the system, and where the data will be stored. It does not show information about process timing or whether processes will use in sequence or in parallel, unlike a traditional structured flowchart which focuses on control flow, or a UML activity workflow diagram, which presents both control and data flows as a unified model.

A DFD is a network that describes the flow of data systems, data stores and processes that change or transform data flows.

There are four commonly encountered data flow diagrams symbols. They are square, circle, directed line, open - ended rectangle. The bubble or circle is the transformation process and the label inside the circle describes the process, using an active verb to do so. Data flows are directed line that finds the input data flows and output data flows at each process circle. Data storage is represented by an open-ended rectangle with a label that finds the data store or file. The square is label to find an external entity that is a source or destination of data flow.

**BASIC DFD SYMBOLS**

**1. Rectangle (Data Source)**

A process represents transformation where incoming data flows are changed into outgoing data flows.

**2. Arrow (Data Source)**

A data flow is a route, which enables packets travel from one point to another. Data may flow from a source to a processor and from data store or process. An arrow line depicts the flow, with arrowhead pointing in the direction of the flow.

**3. Ellipse (Process)**

A data store is a repository of data that is to be stored for use by one or more processes may be as simple as a buffer or sophisticated as a relational database. They should have clear names. If a process merely uses the content of store and docs sot alter it, the arrowhead goes only from the store to the process if a process alters the details in the store, then a double-headed arrow is used.

**4. Side Open Rectangle**

Data Store

A source or sink is a person or part of an organization which enters or receives information from the system but is considered to be the contest of dataflow model.

**5. Circle (Process)**

Process

Circle represents a process that transforms data stream.

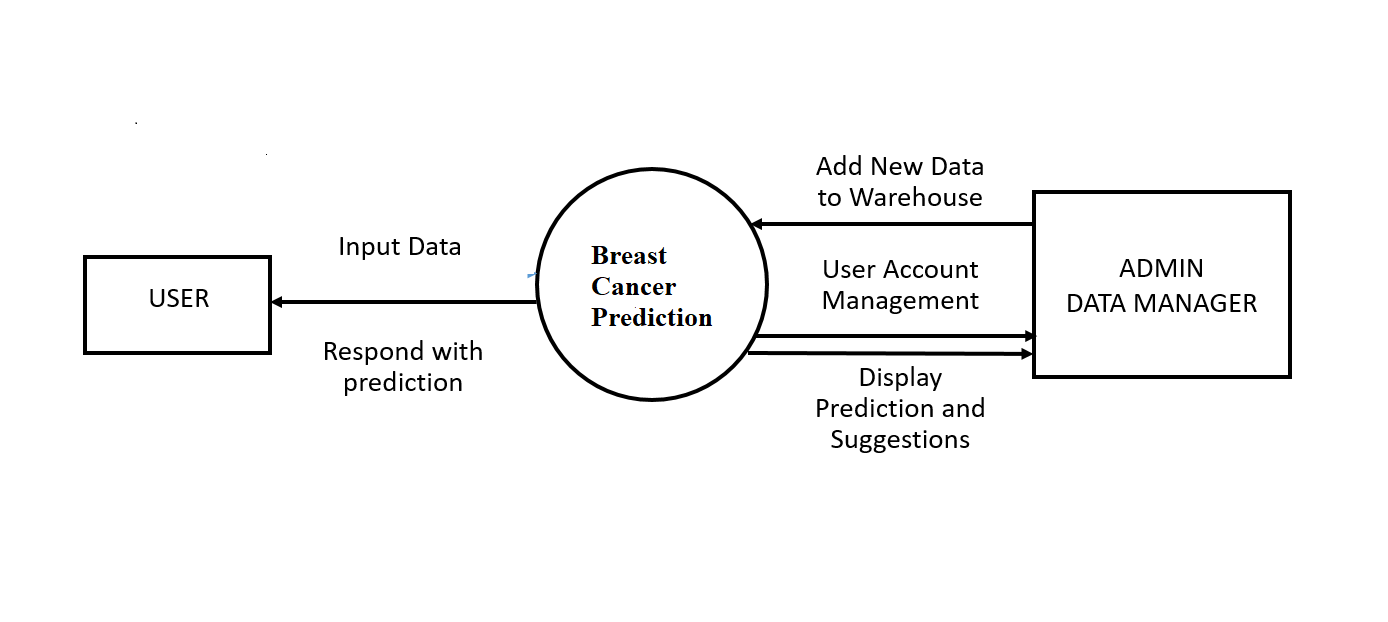
**6. Side Opened Rectangle (Data Store)**

Data Store

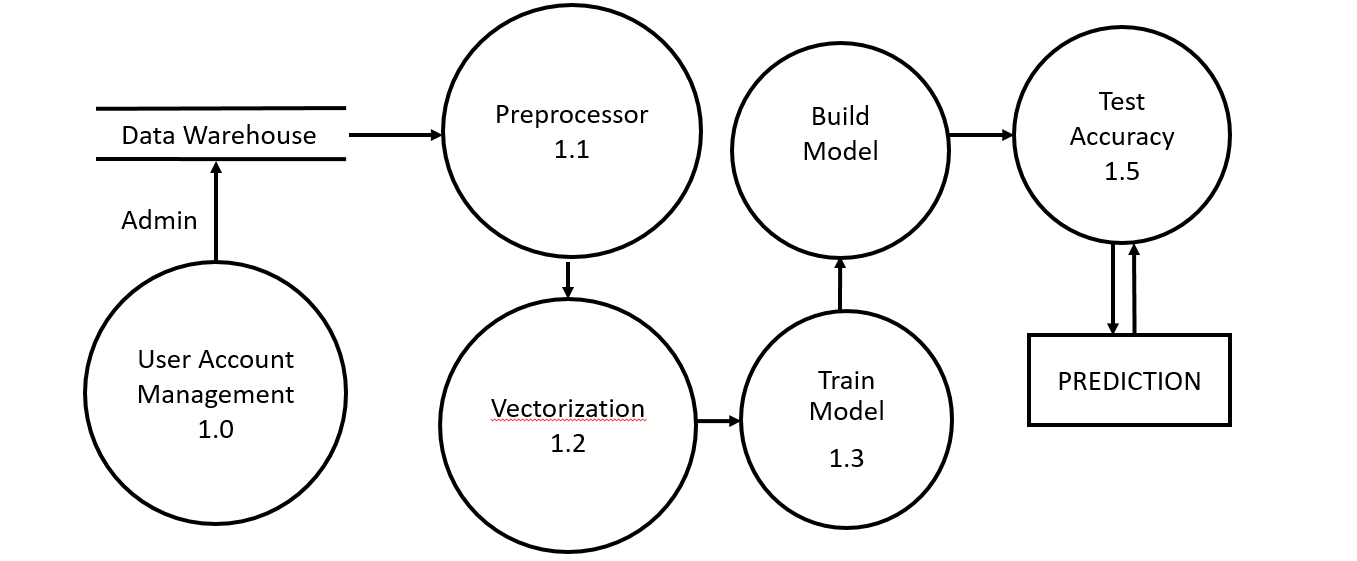
Data store Open-ended rectangle represents a data store.

**DATA FLOW DIAGRAM**

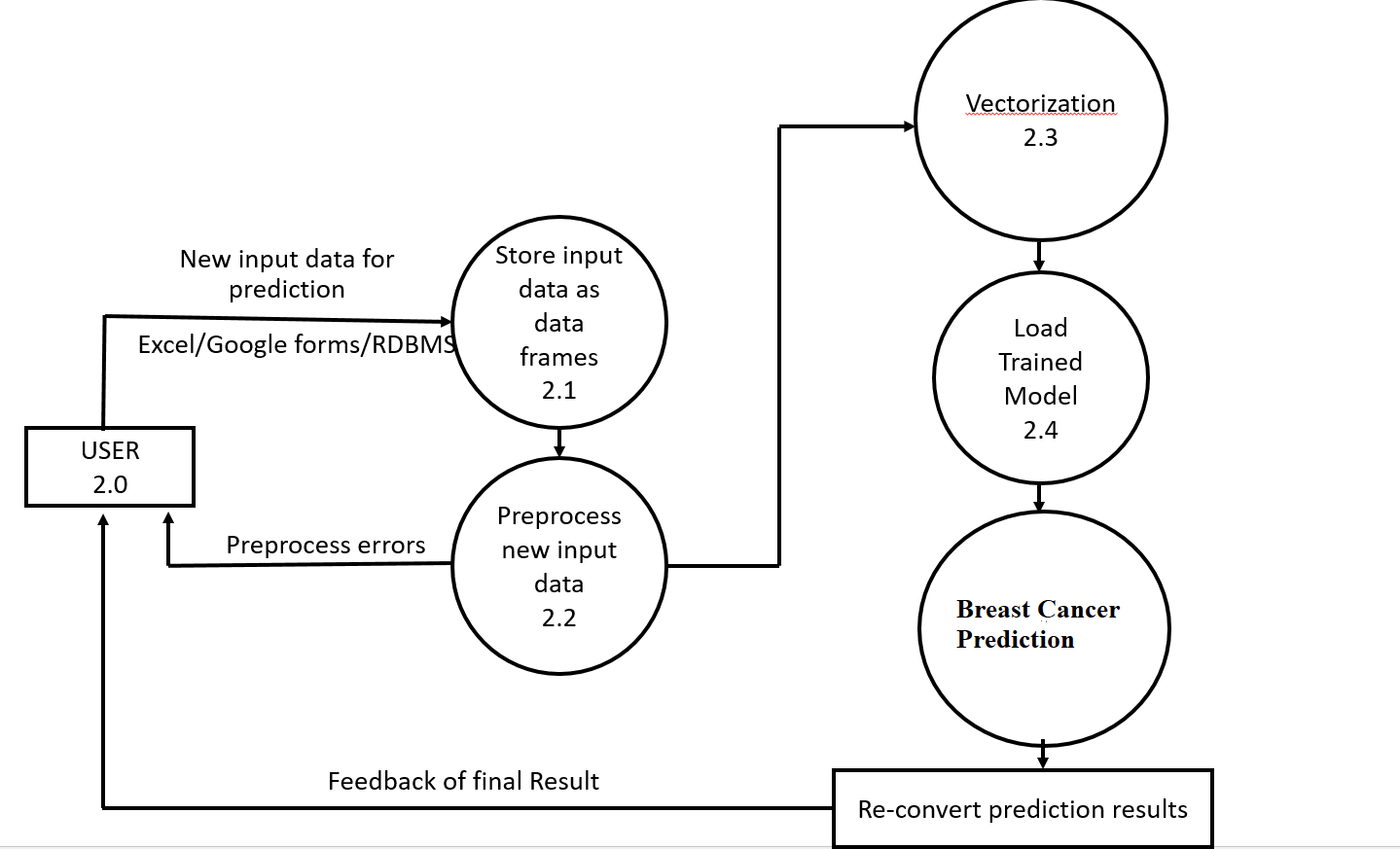
**Level – 0 ( Context Level)**

****

**Level 1**

****

**Level -2**



**Structure Chart/E-R diagrams of the System**

Structure Chart represent hierarchical structure of modules. It breaks down the entire system into lowest functional modules, describe functions and sub-functions of each module of a system to a greater detail. Structure Chart partitions the system into black boxes (functionality of the system is known to the users but inner details are unknown). Inputs are given to the black boxes and appropriate outputs are generated.

**Type of Structure Chart used in this Project:**

**1. Transform Centered Structure:**

**This type of structure** chart are designed for the systems that receive input which are transformed by a sequence of operations being carried out by one module. For example, a login program and a Menu driven program (Spam Detection software).

Symbols used in construction of structured chart:

* **Control Module (Menu Options)**

A control module branches to more than one sub module.

* **Sub Module (Functions)**

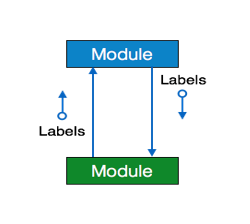
Sub Module is a module which is the part (Child) of another module.

* **Library Module (SVM, Vector Indexer, String Indexer & Accuracy Metrics)**

Library Module are reusable and can be called from any module.

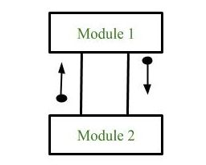
* **Data Flow**

It represents the flow of data between the modules. It is represented by directed arrow with an empty circle at the end.

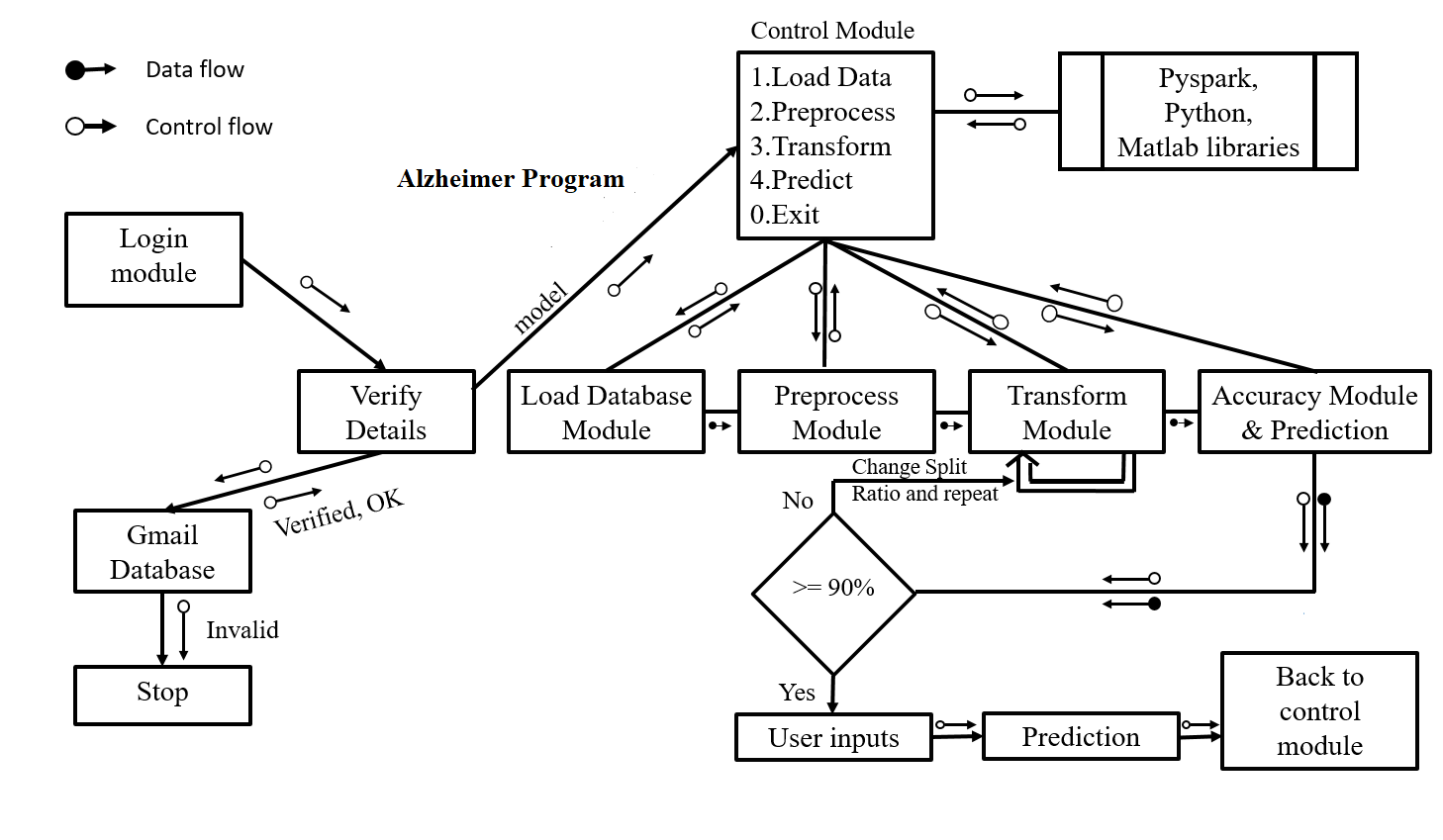


* **Control Flow**

It represents the flow of control between the modules. It is represented by directed arrow with filled circle at the end.

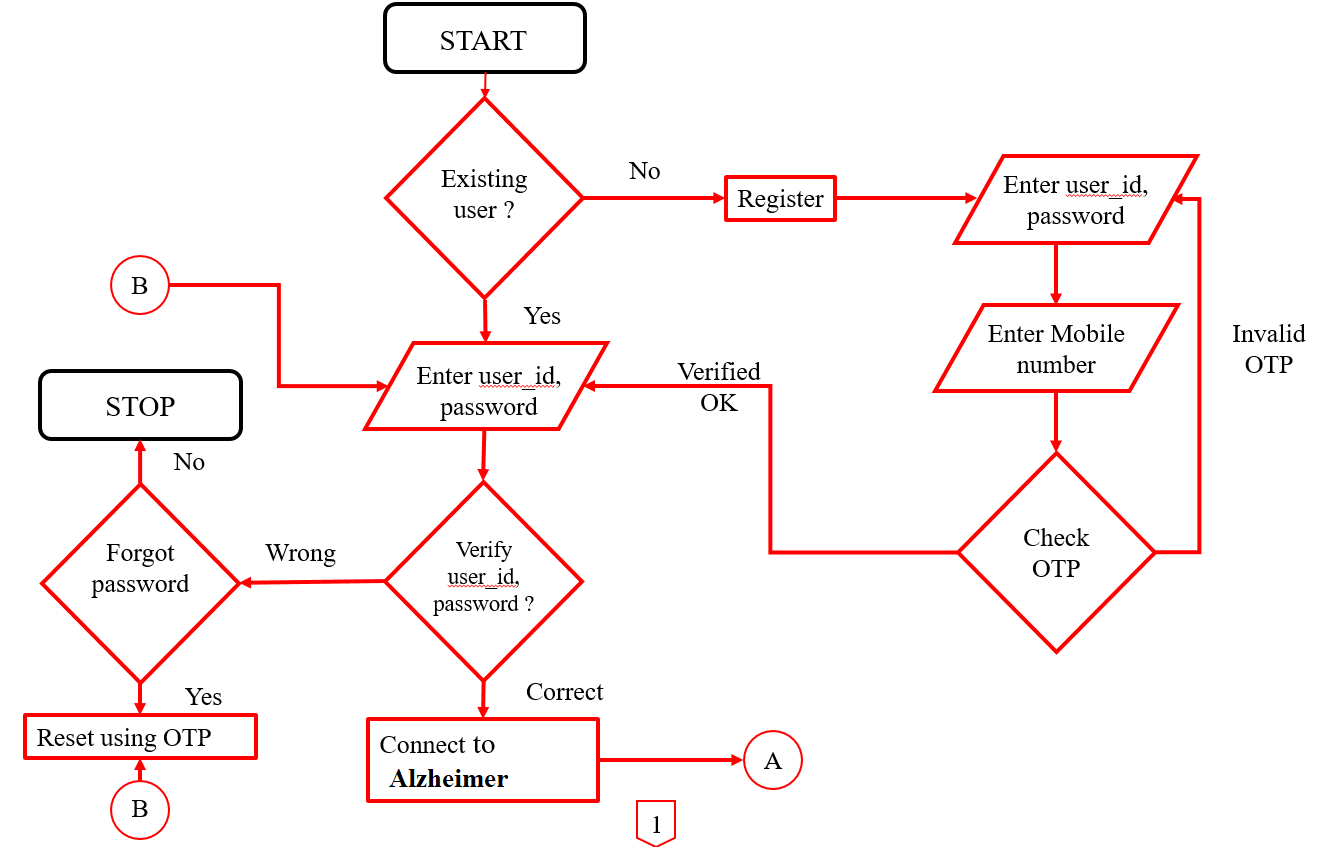


**Structure chart of Breast Cancer Prediction Model**

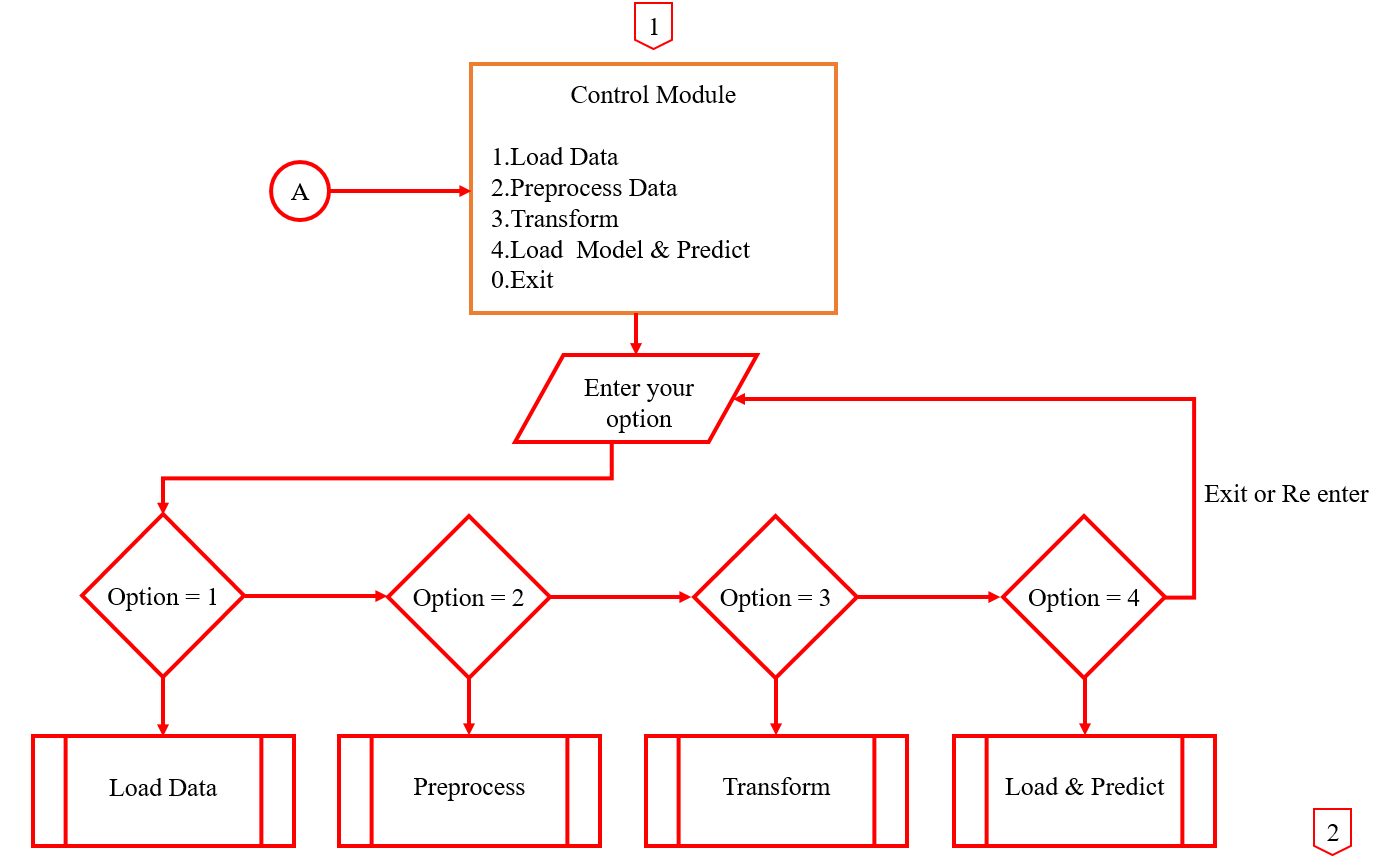
****

**4.6. System flowchart**

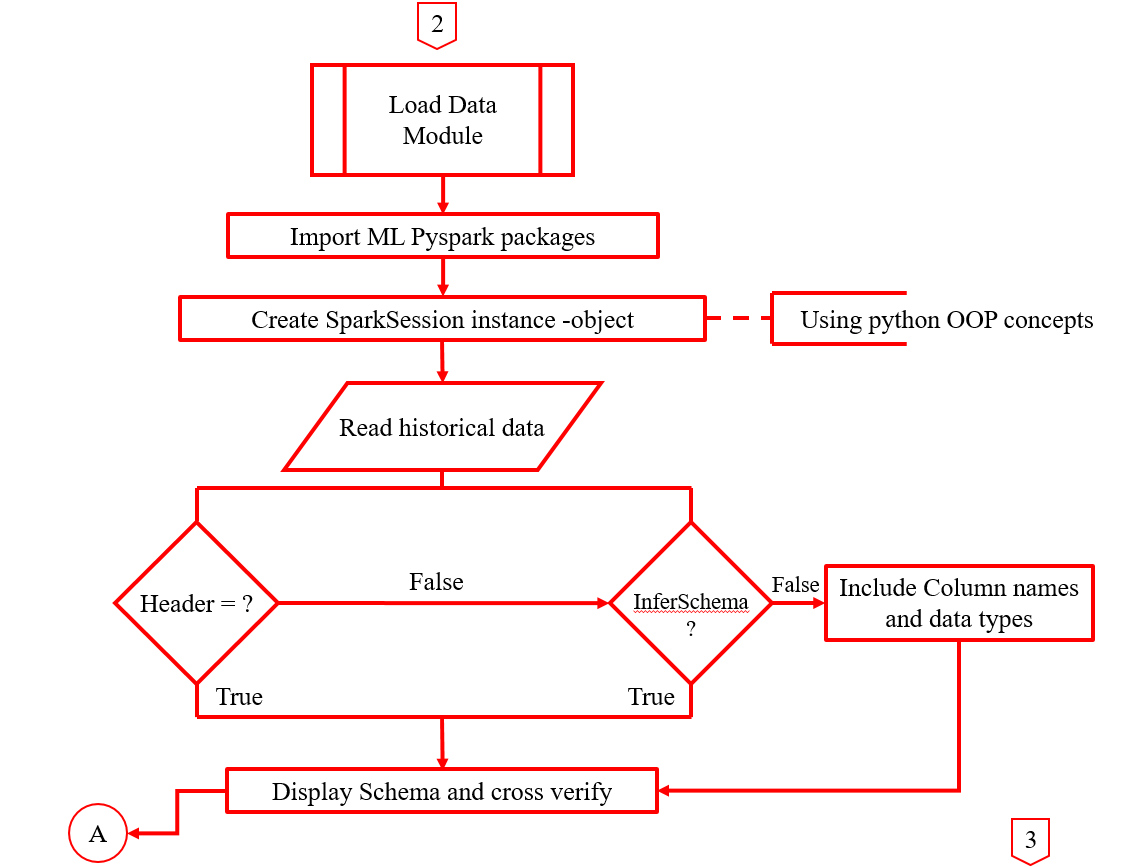
**4.6.1 Login Module**

****

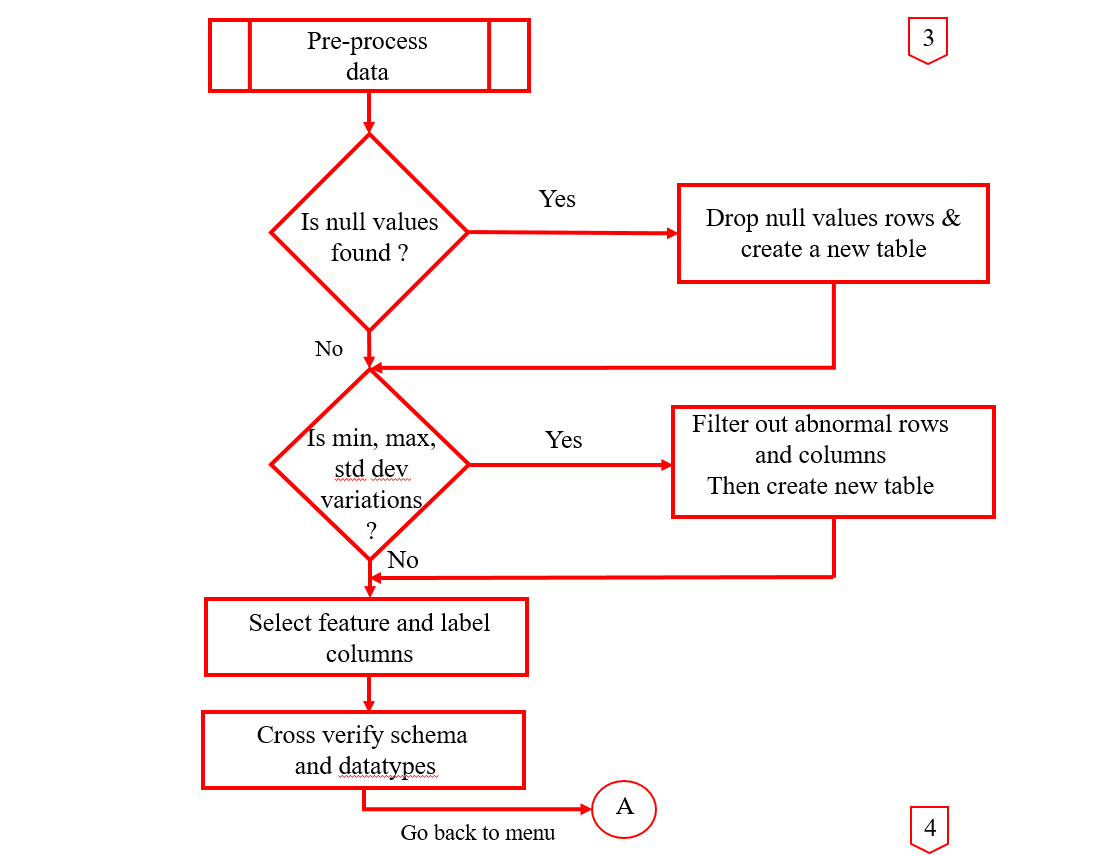
**4.6.2 Control Module**

****

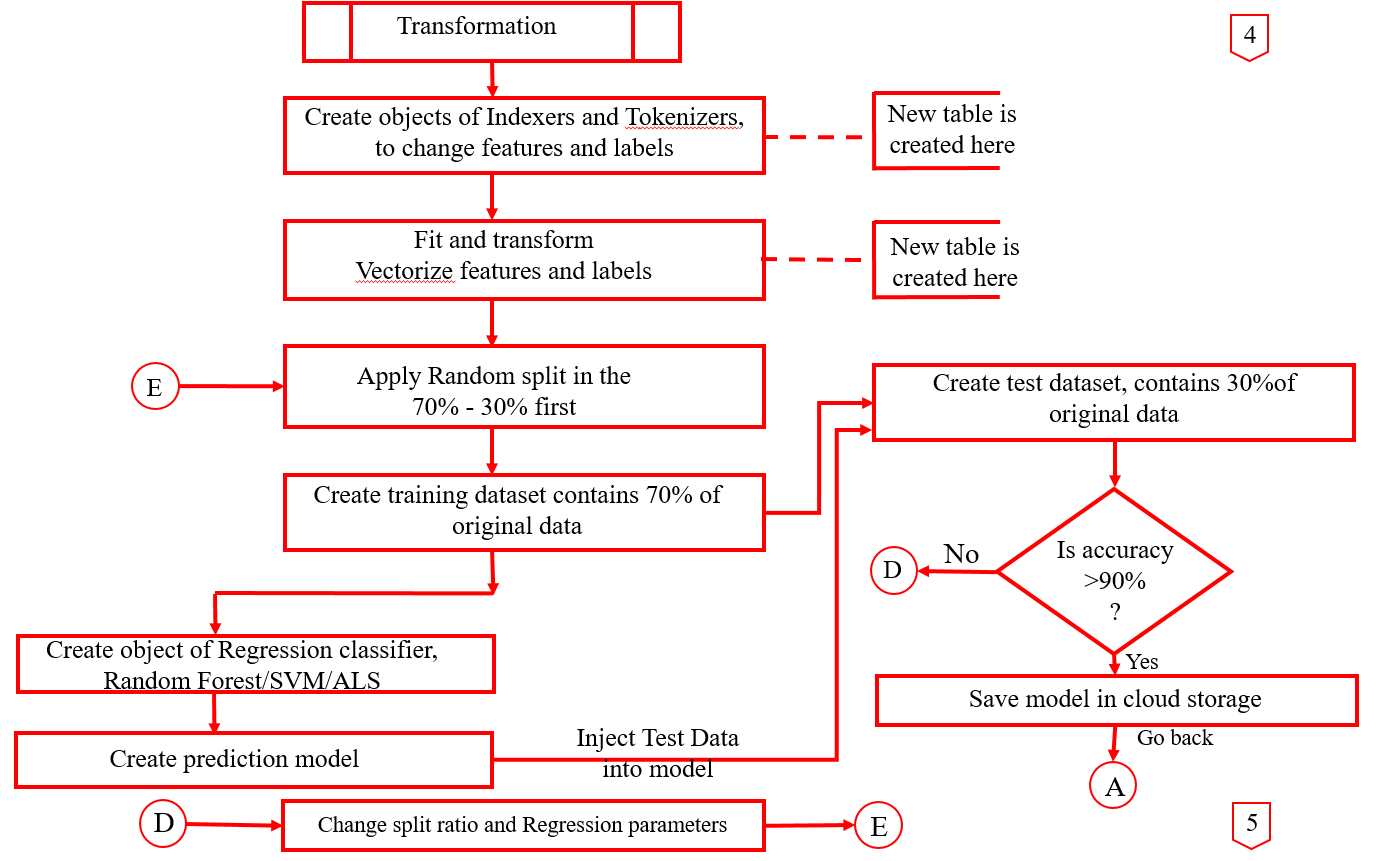
**4.6.3 Load Data Module**



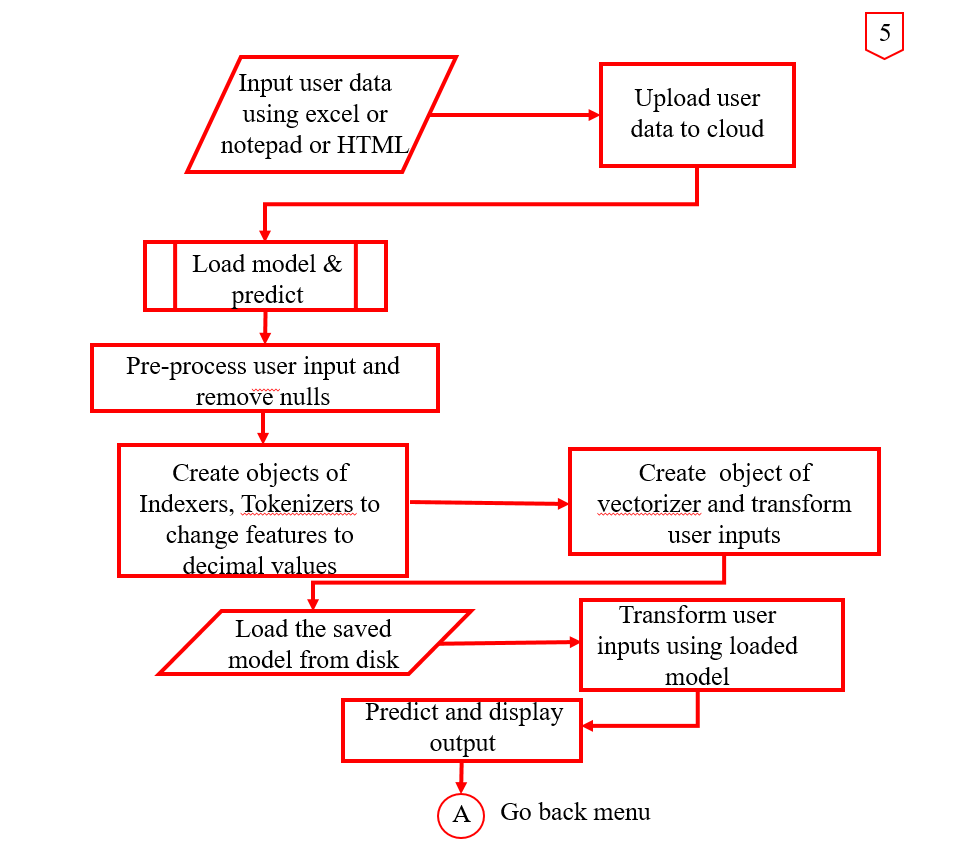
**4.6.4 Pre-process Data Module**

****

**4.6.5 Transformation Module**

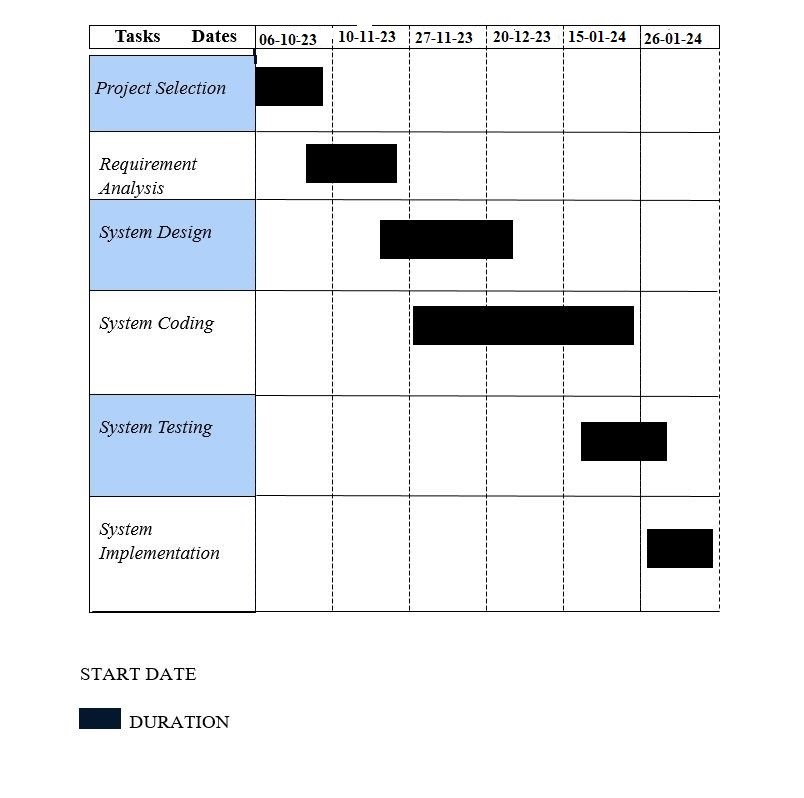


**4.6.6 Load and Predict Module**



1. **GANTT CHART**

A Gantt chart is a bar chart that illustrates a project schedule. It was designed and popularized by Henry Gantt around the years 1910–1915. Modern Gantt charts also show the dependency relationships between activities and the current schedule status.



**DATABASE DESIGN**

Spark Tables are not like traditional RDBMS structure. Apache Spark is a distributed computing framework that is designed for processing large-scale data sets in parallel. While Spark does support SQL-like queries and has its own implementation of a relational database management system (RDBMS) called Spark SQL, it does not actually create databases or tables using traditional RDBMS concepts.

In Spark, data is typically stored in distributed data structures called Resilient Distributed Datasets (RDDs) or in newer Data Frame and Dataset APIs. These data structures can be manipulated and queried using Spark's SQL-like syntax, but they are not stored in a traditional RDBMS.

Spark does have connectors to traditional RDBMSs like MySQL, Microsoft SQL Server, and other popular RDBMS, which allow data to be read from or written to these databases. In these cases, Spark acts as a data processing engine on top of the RDBMS rather than a replacement for it.

**List of fields or attributes in each file or table:**

Spark Tables are immutable. That is the user cannot update data inside a spark table. Instead users can create new Tables with updated data wherever necessary. This is performed by transform commands in Pyspark.

**In this project the following Data frames or Tables are created in RAM:**

**Summary**

1. Database **: Apache SPARK-SQL**

2. Total number of Tables created during training and testing **: 14 Tables**

3. Total number of Tables created for end user Predictions **: 5 Tables**

**Total : 19 Tables**

**Column Name and Attributes of each Table is attached with program listing**

# SYSTEM DEVELOPMENT

System development is the process of defining, designing, testing, and implementing a new software application or program. It could include the internal development of customized systems, the creation of database systems, or the acquisition of third party developed software processing functions. The organization’s management must define and implement standards and adopt an appropriate system development life cycle methodology governing the process of developing, acquiring, and implementing and maintain computerized information systems and related technology.

**SYSTEM TESTING**

Software Testing is the process of executing a program or system with the intent of finding errors. Testing involves any activity aimed at evaluating an attribute or capabilities of a program or system and determining that it meets its required results. The scope of software testing includes the examination of code as well as execution of that code in various environments and conditions as well as examining the quality aspect of code - does it do what it is supposed to do and do what it needs to do. Testing helps not only to uncover errors introduced during coding, but also locates errors committed during the previous phases. There are different testing strategy adopted:

**Unit testing**

This is the first step in testing. In this different module are tested against the specification produces during the design of the modules. It refers to the verification of single program module in an isolated environment. Unit testing focuses on the module independently of one another to locate errors.

**Module testing**

Module Testing will be done to test the interaction between the various programs within one module. It checks the functionality of each program with relation to other programs within the same module. It then tests the overall functionality of each module.

**Integration testing**

In integration testing, all the modules that had completed unit test, is combined. After integration all the modules we get fully working system.

**Validation testing**

Validation testing checks the quality of the software in both simulated environments. After scheduled time, failure and errors are documented and final correction and enhancements are made before the package is released for use. Validation succeeds when the software functions in a manner in which user wishes. Validation refers to process of using software in live environment in order to find errors.

**SYSTEM IMPLEMENTATION**

Implementation is the final and important phase. This is the most critical stage in achieving a successful new system and in giving the users confidence that the new system will work is effective. The system can be implemented only after through testing is done and if it found to working according to the specification. This method also offers the greatest security since the old system can take over with the errors are found or inability to handle certain type of transactions by while using the new system. One major task of preparing for implementation is training the users. The implementation phase of the software development is concerned with translating design specification into source code.

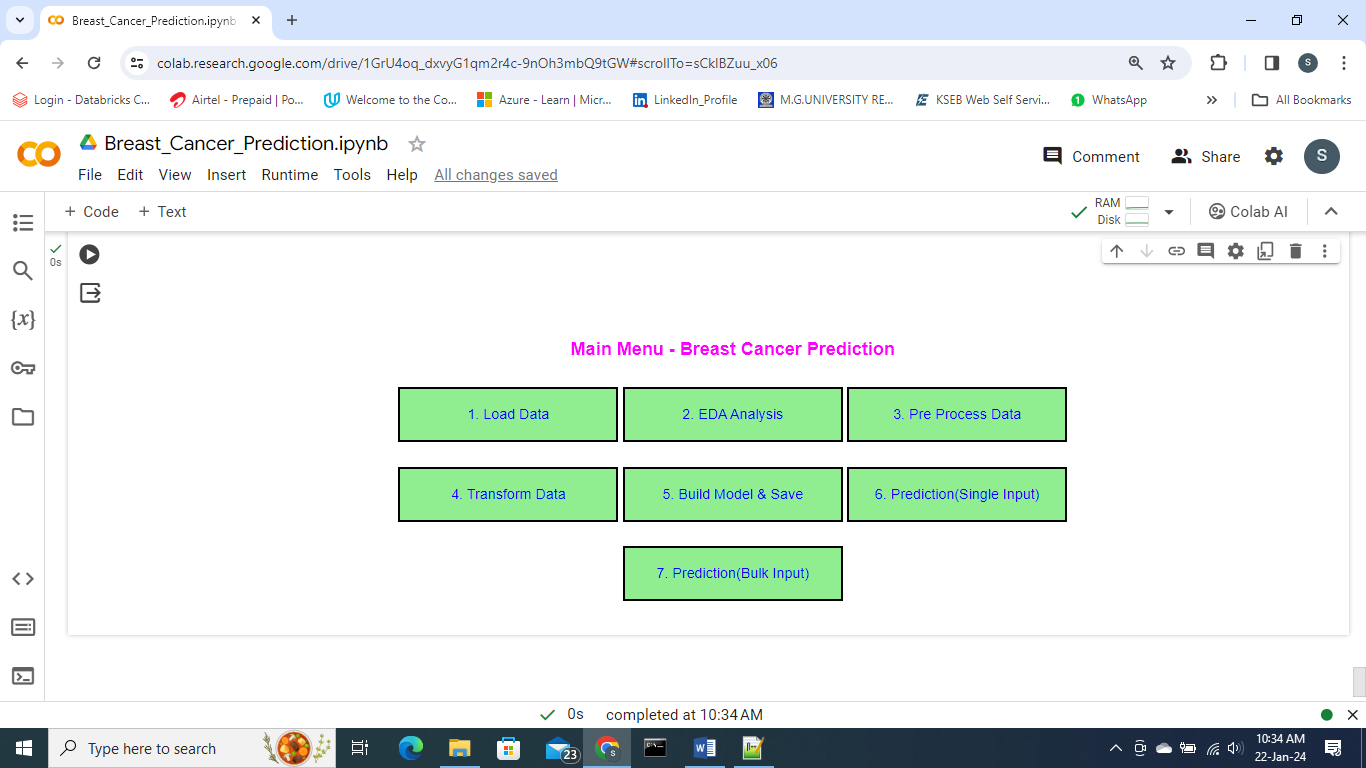
**Design of this Project**

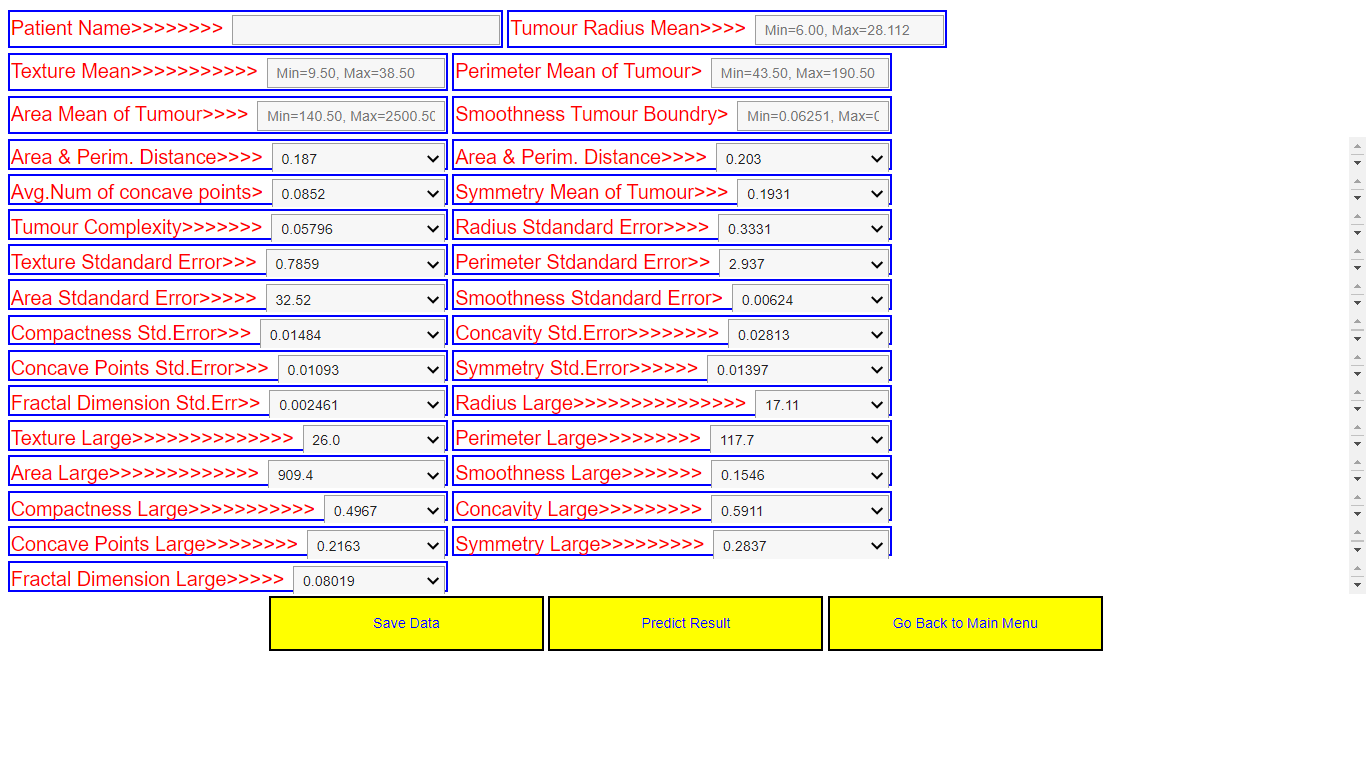
The entire operation of this Machine Learning application is completely controlled by Button-Click-Enabled Menus.

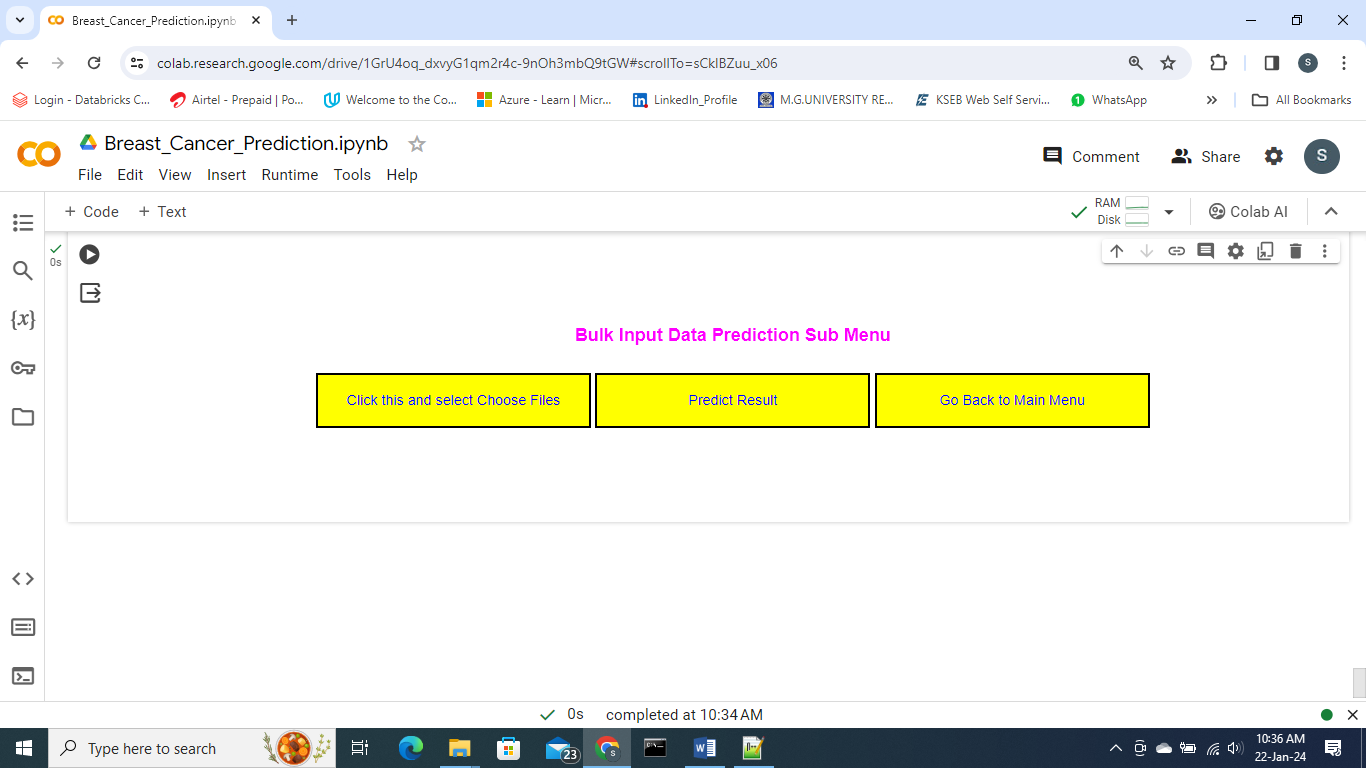
The Button-Click-Enabled Menu-Driven Software Project provides users with a straightforward and interactive way to access different features through a well-designed graphical interface. By incorporating efficient button-click events and clear menu structures, the software enhances user experience and productivity. Continuous testing, documentation, and user training contribute to the overall success and usability of the application.

**Design of each subsystems/modules**

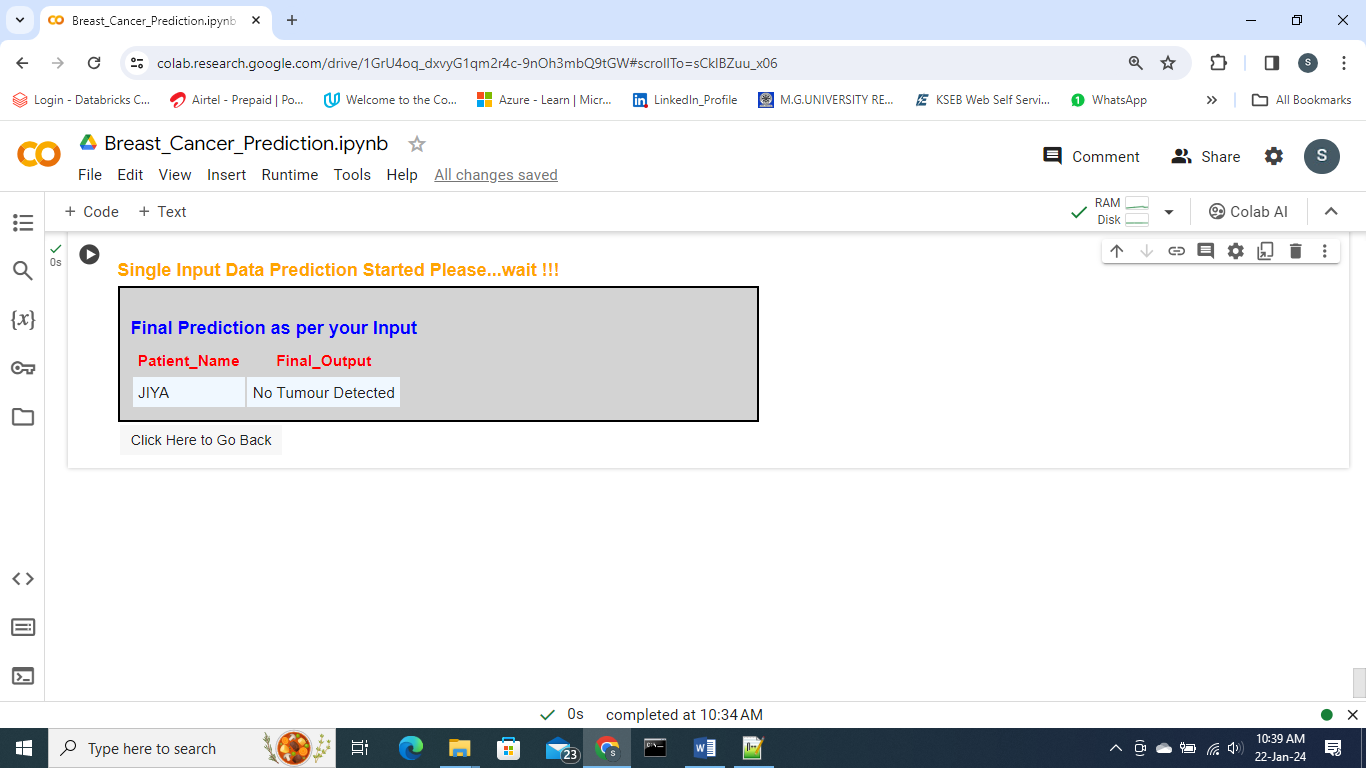
1. Driver Module or Main Menu Module: Controls the entire operation of this Project. Screen shot is attached below.



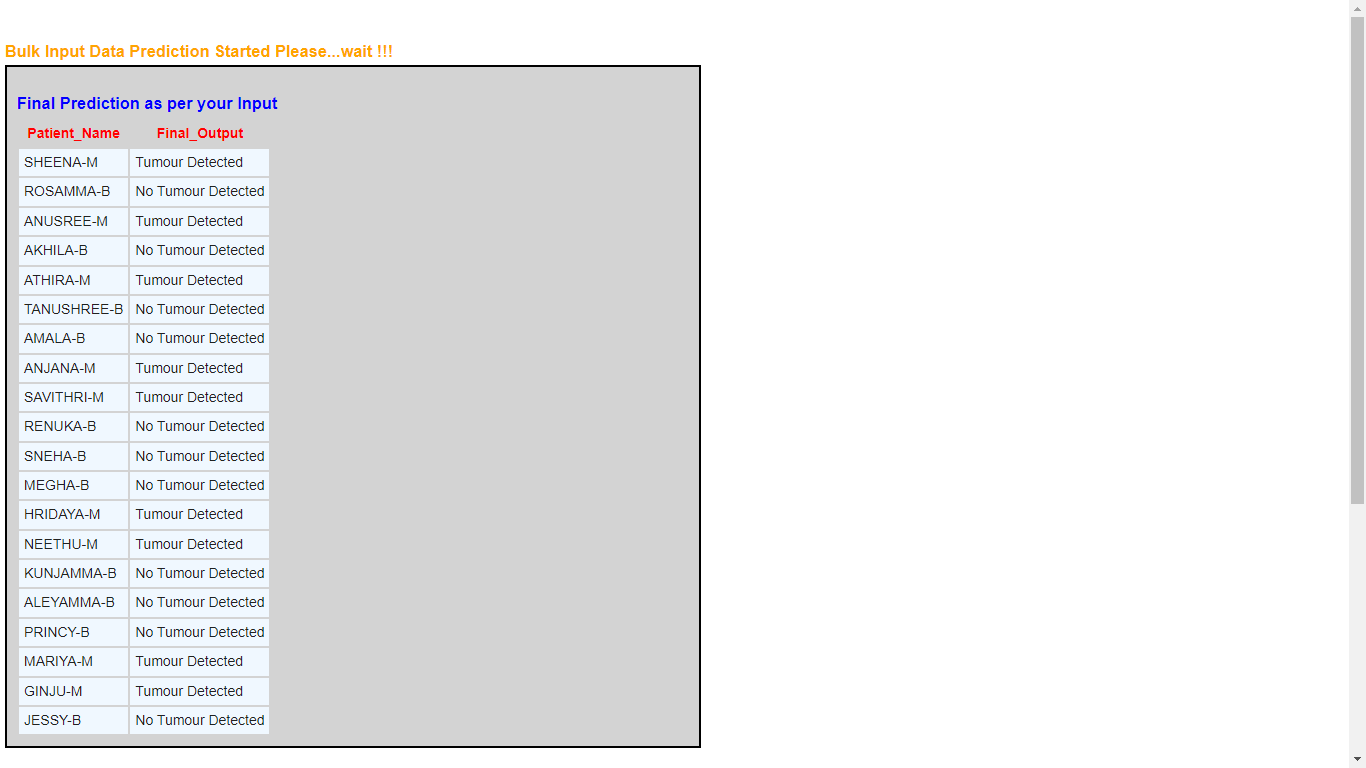




**Final Output** Single **Entry and Predict**

****

**Final Output Bulk Entry and Predict**



**Main Program Code Listing**

Copy the following in Google Colab Code Cell – 1

######### Breast Cancer Prediction using MRI and other Imaging Data

# Depending on the prediction result doctor confirmed Fine Needle Aspiration (FNA)

# procedure and the sample is sent for Biopsy

#########

# Dataset Details

# Total Patients : 20,000

# Total Columns : 32

###### Core Algorithm is Random Forest with Accuracy 99 % ######

# Individual Column Details

# 1. id - Patient id,

# 2. diagnosis - 'M' (Malignant=Cancerous, Benign - Non Cancerous )

# 3. radius\_mean = radius of the tumor

# 4. texture\_mean = variation in grayscale levels within cancer affected area

# 5. perimeter\_mean = Circumference value of the tumour

# 6. area\_mean - total area enclosed by the tumor boundary

# 7. smoothness\_mean - variations in radius lengths of internal tumour cells

# 8. compactness\_mean - how closely the tumor area is to its perimeter (spreding of

# - tumour cells)

# 9. concavity\_mean - average concave portions of the shape in tumour cells

# 10. concave\_points\_mean": It gives the avg number of concave points on the

# : tumour boundary.

# 11. symmetry\_mean - which measures how symmetric or asymmetric the tumor shape is.

# 12. fractal\_dimension\_mean - Complexity of the tumour's perimeter (or ratio)

### The formula for standard error is = ​Standard Deviation / sqrt(Number of Measurements)

# (\_se = Standard Error)

# 13. radius\_se : screening mammograms miss about 1 in 8 breast cancers.

# : Women with dense breasts are more likely to get false-negative

# : results. (\_se = Standard Error)

# 14. texture\_se : standard error of texture within the tumor region(missing grayscale)

# 15. perimeter\_se : This column gives the standard error of the tumor perimeter

# 16. area\_se: It represents the standard error of the tumor area.

# 17. smoothness\_se - Standard error of smoothness

# 18. compactness\_se :represents the standard error of compactness.

# 19. concavity\_se: This column provides the standard error of concavity.

# 20. concave points\_se: standard error of the number of concave points

# 21. symmetry\_se": This column gives the standard error of symmetry.

# 22. fractal\_dimension\_se: It provides the standard error of fractal dimension

## largest or worst values data

# 23. radius\_worst : the worst (largest) value of the tumor radius among several measurements.

# 24. texture\_worst : largest value

# 25. perimeter\_worst : largest value

# 26. area\_worst : largest value of tumour area

# 27. smoothness\_worst :largest value

# 28. compactness\_worst :largest value

# 29. concavity\_worst : largest value

# 30. concave points\_worst : largest value

# 31. symmetry\_worst : largest value

# 32. fractal\_dimension\_worst : largest value (fractal = ratio)

########################

# Step 1 : By default COLAB save your notebook in folder

# "/content/drive/MyDrive/Colab Notebooks/'

# Copy your notebook from "Colab Notebooks" folder to "/content/drive/MyDrive/

# Stpe 2 : Use the UNIX Command 'mv' Move command as follows

!mv "/content/drive/MyDrive/Colab Notebooks/Breast\_Cancer\_Prediction.ipynb" \

"/content/drive/MyDrive"

# Import relevant packages to display buttone events

from IPython.display import display, HTML,Javascript

import ipywidgets as VBox

import ipywidgets as widgets

from IPython.display import clear\_output

# Define a function to scroll your code cell directly to output grid, other wise you will

# have to manually

def set\_focus\_cell1():

display(Javascript("""

document.getElementById('output-area').scrollIntoView({behavior: 'smooth'});

"""))

# Call set\_focus\_cell1 to scroll down automatically this cell

set\_focus\_cell1()

# To get the Processing time of a particular operation import

# the following

import datetime

import time

# Install pyspark, pandas and other ML Packages, 'pip' stands for 'python install packages'

!pip install pyspark

import pyspark

import pandas

from pyspark.ml.classification import LogisticRegression

# Accuracy Evluators

from pyspark.ml.evaluation import BinaryClassificationEvaluator

from pyspark.ml.evaluation import MulticlassClassificationEvaluator

from pyspark.ml.feature import VectorAssembler, StringIndexer, IndexToString

from pyspark.ml.classification import LogisticRegressionModel

from pyspark.sql import SparkSession

from pyspark.sql.functions import col, count, isnull, regexp\_replace, trim, \

when, format\_number

from pyspark.sql.types import DoubleType

# used for array manipulation

import numpy as np

# used for graph plotting

import matplotlib.pyplot as plt

from pyspark.ml.classification import RandomForestClassifier, \

RandomForestClassificationModel

from pyspark.ml.classification import GBTClassifier

from pyspark.ml.classification import GBTClassificationModel

# File Upload package import

from google.colab import files

# For writing single input as csv file

import csv

# To plot graphs import the followng packages

# Seaborn, a statistical data visualization library in Python, is commonly used

# in machine learning for various purposes.

from sklearn.metrics import confusion\_matrix

import seaborn as sns

# Co\_relation package import

from pyspark.ml.stat import Correlation

from functools import reduce

# Create an instance of pyspark as follows:-

my\_spark = SparkSession.builder.getOrCreate()

###### Copy in CODEL CELL 2

def set\_focus\_cell2():

display(Javascript("""

document.getElementById('output-area').scrollIntoView({behavior: 'smooth'});

"""))

# Call set\_focus to scroll down automatically this cell

set\_focus\_cell2()

# Clear output aread Fn

def clear\_result\_grid():

clear\_output(wait=True)

def load\_data(event):

print()

# Record the start time (current\_time) before the process starts

current\_time = datetime.datetime.now()

print("Current Time :", current\_time.strftime("%I:%M:%S %p")) # Format the datetime

display(HTML("<h3 style='color:orange; text-align:left;'>Loading Data, this may take few seconds...wait !!!</h3>"))

global df\_from\_csv

# Read and Load your data

df\_from\_csv = my\_spark.read \

.format("csv") \

.option("header", "True") \

.option("inferSchema", "True") \

.load("/content/drive/MyDrive/breast\_cancer\_syn.csv")

# Cross Verification

print("Schema & Summary of df\_from\_CSV")

df\_from\_csv.printSchema()

# Schema & Summary of df\_from\_CSV

# root

# |-- id: integer (nullable = true)

# |-- diagnosis: string (nullable = true)

# |-- radius\_mean: double (nullable = true)

# |-- texture\_mean: double (nullable = true)

# |-- perimeter\_mean: double (nullable = true)

# |-- area\_mean: double (nullable = true)

# |-- smoothness\_mean: double (nullable = true)

# |-- compactness\_mean: double (nullable = true)

# |-- concavity\_mean: double (nullable = true)

# |-- concave\_points\_mean: double (nullable = true)

# |-- symmetry\_mean: double (nullable = true)

# |-- fractal\_dimension\_mean: double (nullable = true)

# |-- radius\_se: double (nullable = true)

# |-- texture\_se: double (nullable = true)

# |-- perimeter\_se: double (nullable = true)

# |-- area\_se: double (nullable = true)

# |-- smoothness\_se: double (nullable = true)

# |-- compactness\_se: double (nullable = true)

# |-- concavity\_se: double (nullable = true)

# |-- concave\_points\_se: double (nullable = true)

# |-- symmetry\_se: double (nullable = true)

# |-- fractal\_dimension\_se: double (nullable = true)

# |-- radius\_worst: double (nullable = true)

# |-- texture\_worst: double (nullable = true)

# |-- perimeter\_worst: double (nullable = true)

# |-- area\_worst: double (nullable = true)

# |-- smoothness\_worst: double (nullable = true)

# |-- compactness\_worst: double (nullable = true)

# |-- concavity\_worst: double (nullable = true)

# |-- concave\_points\_worst: double (nullable = true)

# |-- symmetry\_worst: double (nullable = true)

# |-- fractal\_dimension\_worst: double (nullable = true)

print("Dispaly few data in df\_from\_CSV")

df\_from\_csv.show(5, False)

# Dispaly few data in df\_from\_CSV

# +-------+---------+-----------+------------+--------------+---------+---------------+----------------+--------------+-------------------+-------------+----------------------+---------+----------+------------+-------+-------------+--------------+------------+-----------------+-----------+--------------------+------------+-------------+---------------+----------+----------------+-----------------+---------------+--------------------+--------------+-----------------------+

# |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|

# +-------+---------+-----------+------------+--------------+---------+---------------+----------------+--------------+-------------------+-------------+----------------------+---------+----------+------------+-------+-------------+--------------+------------+-----------------+-----------+--------------------+------------+-------------+---------------+----------+----------------+-----------------+---------------+--------------------+--------------+-----------------------+

# |846226 |B |20.58 |24.8 |138.4 |1306.0 |0.0974 |0.04227 |0.0 |0.0 |0.2397 |0.078 |0.6242 |3.568 |4.319 |70.71 |0.003139 |0.08297 |0.0889 |0.0409 |0.04484 |0.01284 |26.08 |29.94 |NULL |1976.0 |0.1037 |0.06348 |0.0 |0.0 |0.3176 |0.1023 |

# |915276 |B |8.734 |13.14 |55.27 |234.3 |0.1079 |0.07428 |0.0 |0.0 |0.2097 |0.0652 |0.5169 |1.39 |3.167 |28.85 |0.02171 |0.03284 |0.05001 |0.0152 |0.02632 |0.00454 |10.17 |18.04 |64.01 |317.0 |0.1437 |0.131 |0.0 |0.0 |0.2838 |0.09816 |

# |8610404|M |20.6 |19.65 |140.1 |1265.0 |0.09168 |0.277 |0.3514 |0.152 |0.1798 |0.05391 |0.726 |1.016 |5.772 |86.22 |0.01082 |0.02203 |0.035 |0.01809 |0.0155 |0.001948 |25.74 |24.56 |184.6 |1821.0 |0.15 |0.8781 |0.9387 |0.265 |0.265 |0.06387 |

# |9113846|B |9.904 |29.97 |64.6 |302.4 |0.07699 |0.1294 |0.1307 |0.03716 |0.1701 |0.0596 |0.4311 |3.647 |3.132 |27.48 |0.007339 |0.008243 |0.0 |0.0 |0.03141 |0.003136 |11.26 |38.05 |73.07 |390.2 |0.09422 |0.295 |0.3486 |0.0991 |0.2409 |0.06743 |

# |921644 |B |15.1 |25.42 |99.58 |674.5 |0.08275 |0.1807 |0.1138 |0.08534 |0.184 |0.0568 |0.4309 |1.385 |2.796 |39.84 |0.004775 |0.01172 |0.01947 |0.01269 |0.0187 |0.002626 |16.11 |32.29 |105.9 |762.6 |0.106 |0.2883 |0.196 |0.1423 |0.2722 |0.06956 |

# +-------+---------+-----------+------------+--------------+---------+---------------+----------------+--------------+-------------------+-------------+----------------------+---------+----------+------------+-------+-------------+--------------+------------+-----------------+-----------+--------------------+------------+-------------+---------------+----------+----------------+-----------------+---------------+--------------------+--------------+-----------------------+

# Record the end time after the process has finished

end\_time = datetime.datetime.now()

print("End Time H:M:S :", end\_time.strftime("%I:%M:%S %p")) # Format the datetime

# End Time H:M:S : 06:39:35 AM

# Calculate the time difference

time\_difference = end\_time - current\_time

# Print the time difference

formatted\_time\_difference = str(time\_difference).split('.')[0] # Remove microseconds

print("Loading Time :", formatted\_time\_difference)

# Loading Time : 0:00:09, 9 Seconds

tot\_count = df\_from\_csv.count()

print("Total number of Patients = ",tot\_count)

# Total number of Patients = 15500

print()

display(HTML("<h3 style='color:magenta; text-align:left;'>All your Data loaded Successfully...</h3>"))

main\_function()

def EDA\_start(event):

print()

clear\_result\_grid()

set\_focus\_cell2()

display(HTML("<h3 style='color:orange; text-align:left;'>EDA Started, Please wait !!!</h3>"))

# Display overall statistics of df\_from\_csv data

print("Statistics of df\_from\_csv Dataframe...")

df\_from\_csv.describe().show()

# The following statistics shows no null values or noises in the dataset

# Statistics of df\_from\_csv Dataframe...

# +-------+--------------------+---------+------------------+------------------+------------------+-----------------+------------------+-------------------+-------------------+-------------------+--------------------+----------------------+-------------------+------------------+------------------+-----------------+--------------------+-------------------+-------------------+--------------------+--------------------+--------------------+-----------------+------------------+------------------+-----------------+--------------------+-------------------+------------------+--------------------+--------------------+-----------------------+

# |summary| 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|

# +-------+--------------------+---------+------------------+------------------+------------------+-----------------+------------------+-------------------+-------------------+-------------------+--------------------+----------------------+-------------------+------------------+------------------+-----------------+--------------------+-------------------+-------------------+--------------------+--------------------+--------------------+-----------------+------------------+------------------+-----------------+--------------------+-------------------+------------------+--------------------+--------------------+-----------------------+

# | count| 15500| 15500| 15500| 15500| 15500| 15500| 15500| 15500| 15500| 15500| 15500| 15500| 15500| 15500| 15500| 15500| 15500| 15500| 15500| 15500| 15500| 15500| 15500| 15500| 15499| 15500| 15500| 15500| 15500| 15500| 15500| 15500|

# | mean|3.1080592104709677E7| NULL|14.118614838709515|19.269053548386946| 91.87803677419399|651.4123483870953|0.0963802941935474|0.10354432129032413|0.08807373552903272|0.04853251967741937| 0.18022749032257973| 0.06263311419354899| 0.3999363677419335|1.2091348709677363|2.8097802774193457|39.21633587096738|0.007020247548387051|0.02491077722580628|0.03120139466451588| 0.01178645916129046| 0.02034320322580649|0.003747217258064479|16.22344554838708|25.595256129032396|106.80446544938351|869.9541806451607| 0.1323740135483874|0.25144158838709685|0.2704644070967788| 0.11423316703225739| 0.2886600516129015| 0.08367591548386995|

# | stddev|1.2840527886609587E8| NULL|3.3950013151645333| 4.183981932055307|23.390109123252643|338.0907059987154|0.0135718211324345|0.05082081768919198|0.07685490956585075|0.03745243038207485|0.026078128353025976| 0.006806737254723...|0.26465746753171054|0.5275599576596772|1.9139044596096333|41.89598125126967|0.002892388758860...|0.01681258668257634|0.02954378711872959|0.005892718029682935|0.007702300637379537|0.002533148591634411|4.655080080192638|5.9393224965798845|32.270008268853005|542.3276870436767|0.021852350356253804|0.14882652380104175|0.2003300065899336| 0.06377286459712683|0.059555527879103565| 0.01711727226369945|

# | min| 8670| B| 6.981| 9.71| 43.79| 143.5| 0.06251| 0.01938| 0.0| 0.0| 0.106| 0.04996| 0.1115| 0.3602| 0.757| 6.802| 0.001713| 0.002252| 0.0| 0.0| 0.007882| 8.948E-4| 7.93| 12.02| 50.41| 185.2| 0.07117| 0.02729| 0.0| 0.0| 0.1565| 0.05504|

# | max| 911320502| M| 28.11| 39.28| 188.5| 2501.0| 0.1634| 0.3454| 0.4268| 0.2012| 0.304| 0.09744| 2.873| 4.885| 21.98| 542.2| 0.03113| 0.1354| 0.396| 0.05279| 0.07895| 0.02984| 36.04| 49.54| 251.2| 4254.0| 0.2226| 1.058| 1.252| 0.291| 0.6638| 0.2075|

# +-------+--------------------+---------+------------------+------------------+------------------+-----------------+------------------+-------------------+-------------------+-------------------+--------------------+----------------------+-------------------+------------------+------------------+-----------------+--------------------+-------------------+-------------------+--------------------+--------------------+--------------------+-----------------+------------------+------------------+-----------------+--------------------+-------------------+------------------+--------------------+--------------------+-----------------------+

# Display the total count of distinct rows

tot\_count = df\_from\_csv.count()

distinct\_count = df\_from\_csv.distinct().count()

print("Total count of Duplicate Rows = ", tot\_count - distinct\_count)

# Total count of Duplicate Rows = 135

result\_counts = df\_from\_csv.groupBy("diagnosis").count()

print("Total Count of Malignant/Benign...")

result\_counts.show()

# Total Count of Malignant/Benign...

# +---------+-----+

# |diagnosis|count|

# +---------+-----+

# | B| 9773|

# | M| 5727|

# +---------+-----+

print()

display(HTML("<h3 style='color:magenta; text-align:left;'>EDA Operations finished...</h3>"))

# Return control to main\_function

main\_function()

def pre\_process\_data(event):

print()

print()

clear\_result\_grid()

global df\_final

display(HTML("<h3 style='color:orange; text-align:left;'>Pre-Processing of Data Started, Please wait !!!</h3>"))

# The EDA Report shows 135 duplicate rows, remove it

# Drop the 135 duplicate Rows

df\_drop = df\_from\_csv.dropDuplicates()

# Cross Verify

tot\_count = df\_drop.count()

distinct\_count = df\_drop.distinct().count()

print("Total count of Duplicate Rows after dropping = ", tot\_count - distinct\_count)

# Total count of Duplicate Rows after dropping = 0

print("Statistics of df\_final")

df\_drop.describe().show()

# Cross Verify

######## Important Note : How to interpret Standard Deviation with mean

# Spread around the Mean:

# The standard deviation provides information about how spread out the values in

# a dataset are. A low standard deviation indicates that the values are close to

# the mean, suggesting less variability. A high standard deviation suggests that

# the values are more spread out from the mean, indicating greater variability.

########

# 15500 - 135 = 15365 Tallied

# +-------+--------------------+---------+-----------------+------------------+------------------+-----------------+--------------------+-------------------+-------------------+--------------------+--------------------+----------------------+------------------+------------------+------------------+-----------------+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+------------------+------------------+------------------+-----------------+------------------+-------------------+-------------------+--------------------+-------------------+-----------------------+

# |summary| 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|

# +-------+--------------------+---------+-----------------+------------------+------------------+-----------------+--------------------+-------------------+-------------------+--------------------+--------------------+----------------------+------------------+------------------+------------------+-----------------+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+------------------+------------------+------------------+-----------------+------------------+-------------------+-------------------+--------------------+-------------------+-----------------------+

# | count| 15365| 15365| 15365| 15365| 15365| 15365| 15365| 15365| 15365| 15365| 15365| 15365| 15365| 15365| 15365| 15365| 15365| 15365| 15365| 15365| 15365| 15365| 15365| 15365| 15364| 15365| 15365| 15365| 15365| 15365| 15365| 15365|

# | mean|3.1095856477448747E7| NULL|14.11680032541491|19.264161405792247| 91.86419329645294|651.1903351773507| 0.09639507842499175|0.10349591929710371| 0.0880143486300032|0.048495708167914084| 0.18025485193621835| 0.06263953660917652|0.3997146046208902| 1.209282811584776| 2.808174630654097|39.19708564920258|0.007019449788480315| 0.02489235248942429|0.031172968909860217|0.011781431435079672|0.020346717279531194|0.003746918926130...|16.220665148063713|25.589880247315357|106.78435238219224|869.5783403839927|0.1323958789456554| 0.2514671890660593|0.27048506313049187| 0.11419135366091765| 0.2887142792059874| 0.08367908037748063|

# | stddev|1.2839187298512949E8| NULL|3.391651781033534| 4.183623536084027|23.367682082086706|337.7271245561209|0.013572346873828842|0.05079507972927303|0.07676976146155022| 0.03741917529087799|0.026063882690720852| 0.006808402421778706| 0.264688393967292|0.5282276561203036|1.9134793959118612| 41.9468876573421|0.002880916691704...|0.016793598725729574|0.029554054566571897|0.005892655092650598|0.007704472533005554|0.002534625910690...| 4.650959108941981| 5.939139845103091| 32.23773630270208|541.7550973210069|0.0218483366351756|0.14891835423188962|0.20039197613961204| 0.06373372150363642|0.05951219733424841| 0.017109277790362566|

# | min| 8670| B| 6.981| 9.71| 43.79| 143.5| 0.06251| 0.01938| 0.0| 0.0| 0.106| 0.04996| 0.1115| 0.3602| 0.757| 6.802| 0.001713| 0.002252| 0.0| 0.0| 0.007882| 8.948E-4| 7.93| 12.02| 50.41| 185.2| 0.07117| 0.02729| 0.0| 0.0| 0.1565| 0.05504|

# | max| 911320502| M| 28.11| 39.28| 188.5| 2501.0| 0.1634| 0.3454| 0.4268| 0.2012| 0.304| 0.09744| 2.873| 4.885| 21.98| 542.2| 0.03113| 0.1354| 0.396| 0.05279| 0.07895| 0.02984| 36.04| 49.54| 251.2| 4254.0| 0.2226| 1.058| 1.252| 0.291| 0.6638| 0.2075|

# +-------+--------------------+---------+-----------------+------------------+------------------+-----------------+--------------------+-------------------+-------------------+--------------------+--------------------+----------------------+------------------+------------------+------------------+-----------------+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+------------------+------------------+------------------+-----------------+------------------+-------------------+-------------------+--------------------+-------------------+-----------------------+

# df\_final = df\_drop.withColumn(

# "fractal\_dimension\_se",

# format\_number(col("fractal\_dimension\_se"), 7)

# )

# df\_final = df\_final.withColumn("fractal\_dimension\_se", col("fractal\_dimension\_se") \

# .cast(DoubleType()))

# # Cross Verify 'min', 'max' col of fractal\_dimension\_se

# df\_final.describe().show()

# Remove null values from any of the columns below, 'null' values will cause

# Vector Assembler to raise 'p4j' error

columns\_to\_clean = ["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"]

# Create a condition for each column, to get 'null' value rows

conditions = [col(col\_name).isNull() for col\_name in columns\_to\_clean]

# Combine conditions using logical OR

# lambda x, y: x | y creates an anonymous function (lambda function)

# that takes two arguments (x and y) and returns their bitwise OR (|) result.

# bitwise OR operation between the conditions in the list.

# For example, if conditions is [cond1, cond2, cond3], the reduce function

# would first apply cond1 | cond2, then take the 'result' and apply result | cond3,

# and so on.

combined\_condition = reduce(lambda x, y: x | y, conditions)

# Find rows with null values

rows\_with\_null = df\_drop.filter(combined\_condition)

# Show the rows with null values

print("Rows with null values...")

rows\_with\_null.show()

# Rows with null values...perimeter\_worst col contains 'NULL' value

# +------+---------+-----------+------------+--------------+---------+---------------+----------------+--------------+-------------------+-------------+----------------------+---------+----------+------------+-------+-------------+--------------+------------+-----------------+-----------+--------------------+------------+-------------+---------------+----------+----------------+-----------------+---------------+--------------------+--------------+-----------------------+

# | 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|

# +------+---------+-----------+------------+--------------+---------+---------------+----------------+--------------+-------------------+-------------+----------------------+---------+----------+------------+-------+-------------+--------------+------------+-----------------+-----------+--------------------+------------+-------------+---------------+----------+----------------+-----------------+---------------+--------------------+--------------+-----------------------+

# |846226| B| 20.58| 24.8| 138.4| 1306.0| 0.0974| 0.04227| 0.0| 0.0| 0.2397| 0.078| 0.6242| 3.568| 4.319| 70.71| 0.003139| 0.08297| 0.0889| 0.0409| 0.04484| 0.01284| 26.08| 29.94| NULL| 1976.0| 0.1037| 0.06348| 0.0| 0.0| 0.3176| 0.1023|

# +------+---------+-----------+------------+--------------+---------+---------------+----------------+--------------+-------------------+-------------+----------------------+---------+----------+------------+-------+-------------+--------------+------------+-----------------+-----------+--------------------+------------+-------------+---------------+----------+----------------+-----------------+---------------+--------------------+--------------+-----------------------+

# Drop the row contains 'NULL' value

df\_final = df\_drop.na.drop(subset=columns\_to\_clean)

# Cross Verify

print("Statistics after removing null values...")

df\_final.describe().show()

# Statistics after removing null values... Now total cols = 15364, One 'null' removed

# +-------+--------------------+---------+------------------+------------------+-----------------+------------------+--------------------+-------------------+-------------------+--------------------+-------------------+----------------------+-------------------+-----------------+------------------+------------------+--------------------+-------------------+--------------------+--------------------+--------------------+--------------------+------------------+------------------+------------------+-----------------+-------------------+-------------------+-------------------+--------------------+-------------------+-----------------------+

# |summary| 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|

# +-------+--------------------+---------+------------------+------------------+-----------------+------------------+--------------------+-------------------+-------------------+--------------------+-------------------+----------------------+-------------------+-----------------+------------------+------------------+--------------------+-------------------+--------------------+--------------------+--------------------+--------------------+------------------+------------------+------------------+-----------------+-------------------+-------------------+-------------------+--------------------+-------------------+-----------------------+

# | count| 15364| 15364| 15364| 15364| 15364| 15364| 15364| 15364| 15364| 15364| 15364| 15364| 15364| 15364| 15364| 15364| 15364| 15364| 15364| 15364| 15364| 15364| 15364| 15364| 15364| 15364| 15364| 15364| 15364| 15364| 15364| 15364|

# | mean| 3.109782534170789E7| NULL|14.116379653736011|19.263801093465105|91.86116441030977| 651.1477154386873| 0.09639501301744326| 0.1034999043217911| 0.0880200772390002|0.048498864618588905| 0.1802509828169744| 0.06263853683936457|0.39969999349127683|1.209129289247597|2.8080762952356286|39.195034561312006|0.007019702356157255|0.02488857237698544|0.031169211618068356|0.011779536188492526|0.020345123079926895|0.003746327082790...|16.220023431398005|25.589597110127603|106.78435238219224| 869.506326477483|0.13239774668055163|0.25147942462900297|0.27050266825045605| 0.11419878605831812| 0.2887123991148136| 0.08367786839364683|

# | stddev|1.2839581956501059E8| NULL|3.3913612799894657| 4.18352127730807|23.36542614579766|337.69679228100256|0.013572786166849307|0.05079433089638366|0.07676897571404431| 0.03741834745038695|0.02606031789757567| 0.006807496006732...|0.26469081168882397|0.527901937750755|1.9135028449580005| 41.94748229718155|0.002880840332101...|0.01678760694551607|0.029551346321378062|0.005888161868053247|0.007702188552861104|0.002533646442569...| 4.650430187683613| 5.939229432711341| 32.23773630270208|541.6991799921647|0.02184782098620706|0.14891547706904718|0.20038661490501902| 0.06372913643969555|0.05951367784970861| 0.017109175005455543|

# | min| 8670| B| 6.981| 9.71| 43.79| 143.5| 0.06251| 0.01938| 0.0| 0.0| 0.106| 0.04996| 0.1115| 0.3602| 0.757| 6.802| 0.001713| 0.002252| 0.0| 0.0| 0.007882| 8.948E-4| 7.93| 12.02| 50.41| 185.2| 0.07117| 0.02729| 0.0| 0.0| 0.1565| 0.05504|

# | max| 911320502| M| 28.11| 39.28| 188.5| 2501.0| 0.1634| 0.3454| 0.4268| 0.2012| 0.304| 0.09744| 2.873| 4.885| 21.98| 542.2| 0.03113| 0.1354| 0.396| 0.05279| 0.07895| 0.02984| 36.04| 49.54| 251.2| 4254.0| 0.2226| 1.058| 1.252| 0.291| 0.6638| 0.2075|

# +-------+--------------------+---------+------------------+------------------+-----------------+------------------+--------------------+-------------------+-------------------+--------------------+-------------------+----------------------+-------------------+-----------------+------------------+------------------+--------------------+-------------------+--------------------+--------------------+--------------------+--------------------+------------------+------------------+------------------+-----------------+-------------------+-------------------+-------------------+--------------------+-------------------+-----------------------+

display(HTML("<h3 style='color:magenta; text-align:left;'>Pre-Processing Finished...</h3>"))

main\_function()

def transformation(event):

print()

clear\_result\_grid()

set\_focus\_cell2()

global label\_df

display(HTML("<h3 style='color:orange; text-align:left;'> \

Transformation Started may take less than 1 Minute, Please wait !!!</h3>"))

print()

# Select your feature columns for training & testing continue from here

feature\_cols =["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"]

output\_col = "features"

# Vectorize the above feature\_cols and add it as a list in df\_final\_transformed

assembler = VectorAssembler(inputCols=feature\_cols, outputCol=output\_col)

df\_final\_vectors = assembler.transform(df\_final)

# Cross verify schema of df\_final\_vectors, a new column features will be added

print("Schema of df\_final\_vectors")

df\_final\_vectors.printSchema()

# Schema of df\_final\_vectors

# root

# |-- id: integer (nullable = true)

# |-- diagnosis: string (nullable = true)

# |-- radius\_mean: double (nullable = true)

# |-- texture\_mean: double (nullable = true)

# |-- perimeter\_mean: double (nullable = true)

# |-- area\_mean: double (nullable = true)

# |-- smoothness\_mean: double (nullable = true)

# |-- compactness\_mean: double (nullable = true)

# |-- concavity\_mean: double (nullable = true)

# |-- concave\_points\_mean: double (nullable = true)

# |-- symmetry\_mean: double (nullable = true)

# |-- fractal\_dimension\_mean: double (nullable = true)

# |-- radius\_se: double (nullable = true)

# |-- texture\_se: double (nullable = true)

# |-- perimeter\_se: double (nullable = true)

# |-- area\_se: double (nullable = true)

# |-- smoothness\_se: double (nullable = true)

# |-- compactness\_se: double (nullable = true)

# |-- concavity\_se: double (nullable = true)

# |-- concave\_points\_se: double (nullable = true)

# |-- symmetry\_se: double (nullable = true)

# |-- fractal\_dimension\_se: double (nullable = true)

# |-- radius\_worst: double (nullable = true)

# |-- texture\_worst: double (nullable = true)

# |-- perimeter\_worst: double (nullable = true)

# |-- area\_worst: double (nullable = true)

# |-- smoothness\_worst: double (nullable = true)

# |-- compactness\_worst: double (nullable = true)

# |-- concavity\_worst: double (nullable = true)

# |-- concave\_points\_worst: double (nullable = true)

# |-- symmetry\_worst: double (nullable = true)

# |-- fractal\_dimension\_worst: double (nullable = true)

# |-- features: vector (nullable = true)

print("Display few Data in df\_final\_vectors")

df\_final\_vectors.show(5, False)

# Display few Data in df\_final\_vectors

# +--------+---------+-----------+------------+--------------+---------+---------------+----------------+--------------+-------------------+-------------+----------------------+---------+----------+------------+-------+-------------+--------------+------------+-----------------+-----------+--------------------+------------+-------------+---------------+----------+----------------+-----------------+---------------+--------------------+--------------+-----------------------+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+

# |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|features |

# +--------+---------+-----------+------------+--------------+---------+---------------+----------------+--------------+-------------------+-------------+----------------------+---------+----------+------------+-------+-------------+--------------+------------+-----------------+-----------+--------------------+------------+-------------+---------------+----------+----------------+-----------------+---------------+--------------------+--------------+-----------------------+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+

# |891647 |B |14.97 |14.86 |95.5 |690.2 |0.09834 |0.0595 |0.01945 |0.01939 |0.1717 |0.0627 |0.184 |1.215 |1.286 |16.64 |0.006703 |0.01122 |0.01282 |0.008849 |0.0151 |0.002817 |15.98 |19.22 |102.3 |782.1 |0.1358 |0.2105 |0.1865 |0.05754 |0.252 |0.08022 |[14.97,14.86,95.5,690.2,0.09834,0.0595,0.01945,0.01939,0.1717,0.0627,0.184,1.215,1.286,16.64,0.006703,0.01122,0.01282,0.008849,0.0151,0.002817,15.98,19.22,102.3,782.1,0.1358,0.2105,0.1865,0.05754,0.252,0.08022] |

# |90602302|B |11.54 |21.08 |74.65 |402.9 |0.112 |0.112 |0.06737 |0.02594 |0.2085 |0.06864 |0.2784 |1.213 |1.628 |20.86 |0.008198 |0.03889 |0.04493 |0.02139 |0.02018 |0.005815 |12.26 |27.65 |78.78 |457.8 |0.1517 |0.2118 |0.1797 |0.06918 |0.3003 |0.1048 |[11.54,21.08,74.65,402.9,0.112,0.112,0.06737,0.02594,0.2085,0.06864,0.2784,1.213,1.628,20.86,0.008198,0.03889,0.04493,0.02139,0.02018,0.005815,12.26,27.65,78.78,457.8,0.1517,0.2118,0.1797,0.06918,0.3003,0.1048] |

# |8410138 |B |10.17 |21.54 |64.55 |311.9 |0.08782 |0.08061 |0.01084 |0.0129 |0.2039 |0.06248 |0.5158 |1.255 |3.312 |34.62 |0.005508 |0.03303 |0.03304 |0.01369 |0.0197 |0.004531 |11.02 |30.86 |69.86 |368.6 |0.1313 |0.09866 |0.02168 |0.02579 |0.3179 |0.09664 |[10.17,21.54,64.55,311.9,0.08782,0.08061,0.01084,0.0129,0.2039,0.06248,0.5158,1.255,3.312,34.62,0.005508,0.03303,0.03304,0.01369,0.0197,0.004531,11.02,30.86,69.86,368.6,0.1313,0.09866,0.02168,0.02579,0.3179,0.09664]|

# |8712289 |B |12.19 |22.04 |79.08 |455.8 |0.08439 |0.09509 |0.02855 |0.02882 |0.1801 |0.05553 |0.2005 |0.8561 |1.973 |15.24 |0.00491 |0.02544 |0.02822 |0.01623 |0.01956 |0.00374 |13.34 |28.22 |91.38 |545.2 |0.1228 |0.2585 |0.09915 |0.08187 |0.3589 |0.09187 |[12.19,22.04,79.08,455.8,0.08439,0.09509,0.02855,0.02882,0.1801,0.05553,0.2005,0.8561,1.973,15.24,0.00491,0.02544,0.02822,0.01623,0.01956,0.00374,13.34,28.22,91.38,545.2,0.1228,0.2585,0.09915,0.08187,0.3589,0.09187]|

# |86730502|B |12.87 |21.54 |82.62 |509.3 |0.1008 |0.07858 |0.02955 |0.02862 |0.216 |0.05891 |0.2589 |1.265 |1.775 |31.79 |0.004877 |0.01952 |0.02219 |0.009231 |0.01535 |0.002373 |14.26 |31.68 |90.86 |626.1 |0.1395 |0.2247 |0.1664 |0.08688 |0.348 |0.07619 |[12.87,21.54,82.62,509.3,0.1008,0.07858,0.02955,0.02862,0.216,0.05891,0.2589,1.265,1.775,31.79,0.004877,0.01952,0.02219,0.009231,0.01535,0.002373,14.26,31.68,90.86,626.1,0.1395,0.2247,0.1664,0.08688,0.348,0.07619] |

# +--------+---------+-----------+------------+--------------+---------+---------------+----------------+--------------+-------------------+-------------+----------------------+---------+----------+------------+-------+-------------+--------------+------------+-----------------+-----------+--------------------+------------+-------------+---------------+----------+----------------+-----------------+---------------+--------------------+--------------+-----------------------+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+

# only showing top 5 rows

# Cross Verify features data

# Create 'label' column as follows:-

# The label column or the Diagnosis column in this project is 'Group'

# Convert this to decimal vector using String Indexer

diagnosis\_indexer = StringIndexer(

inputCol="diagnosis",

outputCol="label",

stringOrderType="alphabetAsc"

)

# Since String Indexer is used you should use 'fit' and 'transform'

label\_df = diagnosis\_indexer.fit(df\_final\_vectors).transform(df\_final\_vectors)

# Cross Verify, whether a label column is added

print("Schema of label\_df")

label\_df.printSchema()

# Schema of label\_df

# root

# |-- id: integer (nullable = true)

# |-- diagnosis: string (nullable = true)

# |-- radius\_mean: double (nullable = true)

# |-- texture\_mean: double (nullable = true)

# |-- perimeter\_mean: double (nullable = true)

# |-- area\_mean: double (nullable = true)

# |-- smoothness\_mean: double (nullable = true)

# |-- compactness\_mean: double (nullable = true)

# |-- concavity\_mean: double (nullable = true)

# |-- concave\_points\_mean: double (nullable = true)

# |-- symmetry\_mean: double (nullable = true)

# |-- fractal\_dimension\_mean: double (nullable = true)

# |-- radius\_se: double (nullable = true)

# |-- texture\_se: double (nullable = true)

# |-- perimeter\_se: double (nullable = true)

# |-- area\_se: double (nullable = true)

# |-- smoothness\_se: double (nullable = true)

# |-- compactness\_se: double (nullable = true)

# |-- concavity\_se: double (nullable = true)

# |-- concave\_points\_se: double (nullable = true)

# |-- symmetry\_se: double (nullable = true)

# |-- fractal\_dimension\_se: double (nullable = true)

# |-- radius\_worst: double (nullable = true)

# |-- texture\_worst: double (nullable = true)

# |-- perimeter\_worst: double (nullable = true)

# |-- area\_worst: double (nullable = true)

# |-- smoothness\_worst: double (nullable = true)

# |-- compactness\_worst: double (nullable = true)

# |-- concavity\_worst: double (nullable = true)

# |-- concave\_points\_worst: double (nullable = true)

# |-- symmetry\_worst: double (nullable = true)

# |-- fractal\_dimension\_worst: double (nullable = true)

# |-- features: vector (nullable = true)

# |-- label: double (nullable = false)

print("Display few Data in label\_df")

label\_df.show(5, False)

# Display few Data in label\_df

# +--------+---------+-----------+------------+--------------+---------+---------------+----------------+--------------+-------------------+-------------+----------------------+---------+----------+------------+-------+-------------+--------------+------------+-----------------+-----------+--------------------+------------+-------------+---------------+----------+----------------+-----------------+---------------+--------------------+--------------+-----------------------+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+-----+

# |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|features |label|

# +--------+---------+-----------+------------+--------------+---------+---------------+----------------+--------------+-------------------+-------------+----------------------+---------+----------+------------+-------+-------------+--------------+------------+-----------------+-----------+--------------------+------------+-------------+---------------+----------+----------------+-----------------+---------------+--------------------+--------------+-----------------------+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+-----+

# |891647 |B |14.97 |14.86 |95.5 |690.2 |0.09834 |0.0595 |0.01945 |0.01939 |0.1717 |0.0627 |0.184 |1.215 |1.286 |16.64 |0.006703 |0.01122 |0.01282 |0.008849 |0.0151 |0.002817 |15.98 |19.22 |102.3 |782.1 |0.1358 |0.2105 |0.1865 |0.05754 |0.252 |0.08022 |[14.97,14.86,95.5,690.2,0.09834,0.0595,0.01945,0.01939,0.1717,0.0627,0.184,1.215,1.286,16.64,0.006703,0.01122,0.01282,0.008849,0.0151,0.002817,15.98,19.22,102.3,782.1,0.1358,0.2105,0.1865,0.05754,0.252,0.08022] |0.0 |

# |90602302|B |11.54 |21.08 |74.65 |402.9 |0.112 |0.112 |0.06737 |0.02594 |0.2085 |0.06864 |0.2784 |1.213 |1.628 |20.86 |0.008198 |0.03889 |0.04493 |0.02139 |0.02018 |0.005815 |12.26 |27.65 |78.78 |457.8 |0.1517 |0.2118 |0.1797 |0.06918 |0.3003 |0.1048 |[11.54,21.08,74.65,402.9,0.112,0.112,0.06737,0.02594,0.2085,0.06864,0.2784,1.213,1.628,20.86,0.008198,0.03889,0.04493,0.02139,0.02018,0.005815,12.26,27.65,78.78,457.8,0.1517,0.2118,0.1797,0.06918,0.3003,0.1048] |0.0 |

# |8410138 |B |10.17 |21.54 |64.55 |311.9 |0.08782 |0.08061 |0.01084 |0.0129 |0.2039 |0.06248 |0.5158 |1.255 |3.312 |34.62 |0.005508 |0.03303 |0.03304 |0.01369 |0.0197 |0.004531 |11.02 |30.86 |69.86 |368.6 |0.1313 |0.09866 |0.02168 |0.02579 |0.3179 |0.09664 |[10.17,21.54,64.55,311.9,0.08782,0.08061,0.01084,0.0129,0.2039,0.06248,0.5158,1.255,3.312,34.62,0.005508,0.03303,0.03304,0.01369,0.0197,0.004531,11.02,30.86,69.86,368.6,0.1313,0.09866,0.02168,0.02579,0.3179,0.09664]|0.0 |

# |8712289 |B |12.19 |22.04 |79.08 |455.8 |0.08439 |0.09509 |0.02855 |0.02882 |0.1801 |0.05553 |0.2005 |0.8561 |1.973 |15.24 |0.00491 |0.02544 |0.02822 |0.01623 |0.01956 |0.00374 |13.34 |28.22 |91.38 |545.2 |0.1228 |0.2585 |0.09915 |0.08187 |0.3589 |0.09187 |[12.19,22.04,79.08,455.8,0.08439,0.09509,0.02855,0.02882,0.1801,0.05553,0.2005,0.8561,1.973,15.24,0.00491,0.02544,0.02822,0.01623,0.01956,0.00374,13.34,28.22,91.38,545.2,0.1228,0.2585,0.09915,0.08187,0.3589,0.09187]|0.0 |

# |86730502|B |12.87 |21.54 |82.62 |509.3 |0.1008 |0.07858 |0.02955 |0.02862 |0.216 |0.05891 |0.2589 |1.265 |1.775 |31.79 |0.004877 |0.01952 |0.02219 |0.009231 |0.01535 |0.002373 |14.26 |31.68 |90.86 |626.1 |0.1395 |0.2247 |0.1664 |0.08688 |0.348 |0.07619 |[12.87,21.54,82.62,509.3,0.1008,0.07858,0.02955,0.02862,0.216,0.05891,0.2589,1.265,1.775,31.79,0.004877,0.01952,0.02219,0.009231,0.01535,0.002373,14.26,31.68,90.86,626.1,0.1395,0.2247,0.1664,0.08688,0.348,0.07619] |0.0 |

# +--------+---------+-----------+------------+--------------+---------+---------------+----------------+--------------+-------------------+-------------+----------------------+---------+----------+------------+-------+-------------+--------------+------------+-----------------+-----------+--------------------+------------+-------------+---------------+----------+----------------+-----------------+---------------+--------------------+--------------+-----------------------+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+-----+

# Co\_relation example of feature columns, How to interpret results of

# Pearson coefficient

# pearson coeff = 0.0 Weak linearlity, or correlation is weak

# pearson coeff = 1.0 High linearlity, or correlation is Good

# pearson coeff = -0.0, or negative values No linearlity, or No correlation

# Compute the correlation matrix using Correlation module, this method use

# PEARSON correlation coefficient

# Using DataFrame API to compute the correlation matrix

correlation\_matrix\_coeff = label\_df.stat.corr("radius\_mean","texture\_mean",

method='pearson')

print("Correlation Matrix using radius\_mean,texture\_mean= ", correlation\_matrix\_coeff)

# Correlation Matrix using radius\_mean,texture\_mean= -0.007773313337191215

correlation\_matrix\_coeff = label\_df.stat.corr("radius\_mean","perimeter\_mean",

method='pearson')

print("Correlation Matrix using radius\_mean,perimeter\_mean=", correlation\_matrix\_coeff)

correlation\_matrix\_coeff = label\_df.stat.corr("radius\_mean","area\_mean",

method='pearson')

print("Correlation Matrix using radius\_mean,area\_mean = ", correlation\_matrix\_coeff)

correlation\_matrix\_coeff = label\_df.stat.corr("radius\_mean","smoothness\_mean",

method='pearson')

print("Correlation Matrix using radius\_mean,smoothness\_mean= ", correlation\_matrix\_coeff)

correlation\_matrix\_coeff = label\_df.stat.corr("radius\_mean","compactness\_mean",

method='pearson')

print("Correlation Matrix using radius\_mean,compactness\_mean=", correlation\_matrix\_coeff)

# Correlation Matrix using radius\_mean,texture\_mean= -0.007773313337191215

# Correlation Matrix using radius\_mean,perimeter\_mean= 0.9967140806938268

# Correlation Matrix using radius\_mean,area\_mean = 0.9866676251437023

# Correlation Matrix using radius\_mean,smoothness\_mean= 0.00198948462104386

# Correlation Matrix using radius\_mean,compactness\_mean 0.5284340761762971

######## Similarly combine other columns taking 2 columns at a time

display(HTML("<h3 style='color:magenta; text-align:left;'>Transformation Finished...</h3>"))

main\_function()

def build\_model\_save(event):

print()

clear\_result\_grid()

display(HTML("<h3 style='color:magenta; text-align:left;'> \

Model Building Started Please Wait...will take 1 to 2 minutes..</h3>"))

# Record the start time (current\_time) before the process starts

current\_time = datetime.datetime.now()

print("Current Time :", current\_time.strftime("%I:%M:%S %p")) # Format the datetime

##### Prepare Training and Testing Data

# Split label\_df, 70 % for Training and 30 % for Testing

train\_data\_df, test\_data\_df = label\_df.randomSplit([0.70, 0.30], seed=5043)

train\_data\_df.show(5, False)

# +----+---------+-----------+------------+--------------+---------+---------------+----------------+--------------+-------------------+-------------+----------------------+---------+----------+------------+-------+-------------+--------------+------------+-----------------+-----------+--------------------+------------+-------------+---------------+----------+----------------+-----------------+---------------+--------------------+--------------+-----------------------+----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+-----+

# |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|features |label|

# +----+---------+-----------+------------+--------------+---------+---------------+----------------+--------------+-------------------+-------------+----------------------+---------+----------+------------+-------+-------------+--------------+------------+-----------------+-----------+--------------------+------------+-------------+---------------+----------+----------------+-----------------+---------------+--------------------+--------------+-----------------------+----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+-----+

# |8670|B |11.06 |19.48 |70.31 |378.1 |0.1092 |0.04768 |0.02712 |0.007246 |0.1931 |0.05796 |0.1855 |0.7859 |1.263 |12.98 |0.00624 |0.01484 |0.02813 |0.01093 |0.01397 |0.002461 |12.68 |26.0 |80.79 |496.7 |0.1546 |0.1879 |0.2079 |0.05556 |0.2837 |0.08019 |[11.06,19.48,70.31,378.1,0.1092,0.04768,0.02712,0.007246,0.1931,0.05796,0.1855,0.7859,1.263,12.98,0.00624,0.01484,0.02813,0.01093,0.01397,0.002461,12.68,26.0,80.79,496.7,0.1546,0.1879,0.2079,0.05556,0.2837,0.08019]|0.0 |

# |8670|B |11.74 |19.48 |76.31 |426.0 |0.1092 |0.06915 |0.04695 |0.02339 |0.1931 |0.05796 |0.1924 |0.7859 |1.345 |13.04 |0.00624 |0.01484 |0.02893 |0.01386 |0.01397 |0.002461 |12.45 |26.0 |81.25 |473.8 |0.1546 |0.183 |0.1471 |0.1067 |0.2837 |0.08019 |[11.74,19.48,76.31,426.0,0.1092,0.06915,0.04695,0.02339,0.1931,0.05796,0.1924,0.7859,1.345,13.04,0.00624,0.01484,0.02893,0.01386,0.01397,0.002461,12.45,26.0,81.25,473.8,0.1546,0.183,0.1471,0.1067,0.2837,0.08019] |0.0 |

# |8670|B |12.21 |19.48 |78.31 |458.4 |0.1092 |0.07175 |0.04392 |0.02027 |0.1931 |0.05796 |0.2527 |0.7859 |1.874 |18.57 |0.00624 |0.01484 |0.02813 |0.01093 |0.01397 |0.002461 |14.29 |26.0 |93.85 |624.6 |0.1546 |0.217 |0.2413 |0.08829 |0.2837 |0.08019 |[12.21,19.48,78.31,458.4,0.1092,0.07175,0.04392,0.02027,0.1931,0.05796,0.2527,0.7859,1.874,18.57,0.00624,0.01484,0.02813,0.01093,0.01397,0.002461,14.29,26.0,93.85,624.6,0.1546,0.217,0.2413,0.08829,0.2837,0.08019] |0.0 |

# |8670|B |13.21 |19.48 |84.88 |538.4 |0.1092 |0.06877 |0.02987 |0.03275 |0.1931 |0.05796 |0.2351 |0.7859 |1.539 |17.85 |0.00624 |0.01484 |0.02813 |0.01093 |0.01397 |0.002461 |14.37 |26.0 |92.48 |629.6 |0.1546 |0.1381 |0.1062 |0.07958 |0.2837 |0.08019 |[13.21,19.48,84.88,538.4,0.1092,0.06877,0.02987,0.03275,0.1931,0.05796,0.2351,0.7859,1.539,17.85,0.00624,0.01484,0.02813,0.01093,0.01397,0.002461,14.37,26.0,92.48,629.6,0.1546,0.1381,0.1062,0.07958,0.2837,0.08019] |0.0 |

# |8670|M |19.79 |19.48 |130.4 |1192.0 |0.1092 |0.1589 |0.2545 |0.1149 |0.1931 |0.05796 |0.4953 |0.7859 |2.765 |63.33 |0.00624 |0.01484 |0.02813 |0.01093 |0.01397 |0.002461 |22.63 |26.0 |148.7 |1589.0 |0.1546 |0.3861 |0.5673 |0.1732 |0.2837 |0.08019 |[19.79,19.48,130.4,1192.0,0.1092,0.1589,0.2545,0.1149,0.1931,0.05796,0.4953,0.7859,2.765,63.33,0.00624,0.01484,0.02813,0.01093,0.01397,0.002461,22.63,26.0,148.7,1589.0,0.1546,0.3861,0.5673,0.1732,0.2837,0.08019] |1.0 |

# +----+---------+-----------+------------+--------------+---------+---------------+----------------+--------------+-------------------+-------------+----------------------+---------+----------+------------+-------+-------------+--------------+------------+-----------------+-----------+--------------------+------------+-------------+---------------+----------+----------------+-----------------+---------------+--------------------+--------------+-----------------------+----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+-----+

# Count total number of rows in train\_data\_df, test\_data\_df

tot\_train = train\_data\_df.count()

print("Total Rows in Training Dataset..=", tot\_train)

# Total Rows in Training Dataset..= 10776

tot\_test = test\_data\_df.count()

print("Total Rows in Test Dataset..=", tot\_test)

# Total Rows in Test Dataset..= 4588

# Create a Logistic Regression model and set the labelCol to 'label'

lr = LogisticRegression(maxIter=10, regParam=0.3, elasticNetParam=0.8,

featuresCol="features",

labelCol='label')

# Fit the model to your data

lr\_model = lr.fit(train\_data\_df)

# Predict using LR, apply test data on top of lr\_model

prediction\_lr = lr\_model.transform(test\_data\_df)

####### Evaluate your accuracy of model using ACCURACY Metric

# For multi-class classification in PySpark, you can use metrics such as:

# accuracy: To calculate the accuracy of the model's predictions.

# weightedPrecision: To calculate the weighted precision.

# weightedRecall: To calculate the weighted recall.

# weightedFMeasure: To calculate the weighted F1 score.

# weightedFalsePositiveRate: To calculate the weighted false positive rate.

# evaluator\_lr\_auc = BinaryClassificationEvaluator(labelCol="label",

# rawPredictionCol="prediction",

# metricName="areaUnderROC")

####### Metrics Explanation

# Accuracy is a straightforward and widely used metric that measures the overall correctness

# of a model's predictions.

# Mathematically, it is defined as:

# Accuracy = (Number of Correct Predictions) / (Total Number of Predictions)

evaluator\_lr\_accuracy = MulticlassClassificationEvaluator(labelCol="label",

predictionCol="prediction",

metricName="accuracy")

evaluator\_lr\_auc\_accuracy = evaluator\_lr\_accuracy.evaluate(prediction\_lr)

lr\_accuracy\_rounded = round(evaluator\_lr\_auc\_accuracy \* 100, 2)

print()

print("Accuracy of Prediction using LR with Different Metrics....")

print("==========================================================")

print("Accuracy of LR using accuracy metrics = ", lr\_accuracy\_rounded)

# ###########

####### Evaluate your accuracy of model using 'weightedRecall' or F1 score

# Weighted recall is a more robust metric when dealing with imbalanced datasets, as it

# gives each class a fair contribution to the overall evaluation.

# The formula for weighted recall is:

# Weighted Recall = Σ (Recall for each class \* Proportion of that class in the dataset)

#########

evaluator\_lr\_recall = MulticlassClassificationEvaluator(labelCol="label",

predictionCol="prediction",

metricName="weightedRecall")

evaluator\_lr\_recall\_accuracy = evaluator\_lr\_recall.evaluate(prediction\_lr)

lr\_recall\_rounded = round(evaluator\_lr\_recall\_accuracy \* 100, 2)

print("Accuracy of LR using FMeasure = ", lr\_recall\_rounded)

print()

display(HTML("<h3 style='color:green; text-align:left;'>Processing Random Forest Please Wait !!!</h3>"))

# # Accuracy of Prediction using LR with Different Metrics...., with 70-30 split raio

# # ==========================================================

# Accuracy of LR using accuracy metrics = 82.56

# Accuracy of LR using FMeasure = 82.56

# ########### Logistic Regression finished

##### Create a Random Forest Model and set the labelCol to 'label'

# The 'setImpurity' criteria in RF can be 'gini', 'variance', 'entropy'

# 'entropy' is used to eliminate 'over fitting' of train data

# There are several different 'feature subset' strategies available, including:

# you should try different feature subsets to get max accuracy

# "all": Considers all features at each split.

# "random": Considers a random subset of features at each split.

# "sqrt": Considers a random subset of features that is equal to the square root of the

# total number of features.

# "log2": Considers a random subset of features that is equal to the log2 of the

# total number of features.

rf = RandomForestClassifier() \

.setImpurity("entropy") \

.setNumTrees(20) \

.setMaxDepth(20) \

.setFeatureSubsetStrategy("all") \

.setSeed(5043) # Used to get fixed values in each run

# Fit the model to your data

rf\_model = rf.fit(train\_data\_df)

# Predict using RF, apply test data on top of rf\_model

prediction\_rf = rf\_model.transform(test\_data\_df)

# Cross Verify

print("Display Data in predicition\_df...")

prediction\_rf.show(30, False)

# Display Data in predicition\_df...

# +----+---------+-----------+------------+--------------+---------+---------------+----------------+--------------+-------------------+-------------+----------------------+---------+----------+------------+-------+-------------+--------------+------------+-----------------+-----------+--------------------+------------+-------------+---------------+----------+----------------+-----------------+---------------+--------------------+--------------+-----------------------+-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+-----+-------------+-----------+----------+

# |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|features |label|rawPrediction|probability|prediction|

# +----+---------+-----------+------------+--------------+---------+---------------+----------------+--------------+-------------------+-------------+----------------------+---------+----------+------------+-------+-------------+--------------+------------+-----------------+-----------+--------------------+------------+-------------+---------------+----------+----------------+-----------------+---------------+--------------------+--------------+-----------------------+-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+-----+-------------+-----------+----------+

# |8670|M |15.46 |19.48 |103.8 |731.3 |0.1092 |0.187 |0.203 |0.0852 |0.1931 |0.05796 |0.3331 |0.7859 |2.937 |32.52 |0.00624 |0.01484 |0.02813 |0.01093 |0.01397 |0.002461 |17.11 |26.0 |117.7 |909.4 |0.1546 |0.4967 |0.5911 |0.2163 |0.2837 |0.08019 |[15.46,19.48,103.8,731.3,0.1092,0.187,0.203,0.0852,0.1931,0.05796,0.3331,0.7859,2.937,32.52,0.00624,0.01484,0.02813,0.01093,0.01397,0.002461,17.11,26.0,117.7,909.4,0.1546,0.4967,0.5911,0.2163,0.2837,0.08019] |1.0 |[0.0,20.0] |[0.0,1.0] |1.0 |

# |8912|B |10.82 |27.09 |68.89 |361.6 |0.1026 |0.06602 |0.01548 |0.00816 |0.1908 |0.06324 |0.5196 |2.233 |3.564 |33.0 |0.007004 |0.02261 |0.03051 |0.01197 |0.01715 |0.003876 |13.03 |33.15 |83.9 |505.6 |0.1461 |0.1633 |0.06194 |0.03264 |0.3103 |0.09581 |[10.82,27.09,68.89,361.6,0.1026,0.06602,0.01548,0.00816,0.1908,0.06324,0.5196,2.233,3.564,33.0,0.007004,0.02261,0.03051,0.01197,0.01715,0.003876,13.03,33.15,83.9,505.6,0.1461,0.1633,0.06194,0.03264,0.3103,0.09581] |0.0 |[20.0,0.0] |[1.0,0.0] |0.0 |

# |8913|B |10.36 |13.12 |66.62 |323.6 |0.06955 |0.06895 |0.02557 |0.01969 |0.1337 |0.05581 |0.1507 |0.469 |1.263 |11.06 |0.004731 |0.01345 |0.01652 |0.005905 |0.01619 |0.002081 |11.26 |15.54 |72.04 |394.5 |0.09616 |0.1841 |0.1941 |0.06588 |0.2309 |0.06915 |[10.36,13.12,66.62,323.6,0.06955,0.06895,0.02557,0.01969,0.1337,0.05581,0.1507,0.469,1.263,11.06,0.004731,0.01345,0.01652,0.005905,0.01619,0.002081,11.26,15.54,72.04,394.5,0.09616,0.1841,0.1941,0.06588,0.2309,0.06915]|0.0 |[20.0,0.0] |[1.0,0.0] |0.0 |

# |8913|B |11.64 |13.12 |75.17 |412.5 |0.06955 |0.1017 |0.0707 |0.03485 |0.1337 |0.05581 |0.306 |0.469 |2.155 |20.62 |0.004731 |0.01345 |0.01652 |0.005905 |0.01619 |0.002081 |13.14 |15.54 |85.51 |521.7 |0.09616 |0.266 |0.2873 |0.1218 |0.2309 |0.06915 |[11.64,13.12,75.17,412.5,0.06955,0.1017,0.0707,0.03485,0.1337,0.05581,0.306,0.469,2.155,20.62,0.004731,0.01345,0.01652,0.005905,0.01619,0.002081,13.14,15.54,85.51,521.7,0.09616,0.266,0.2873,0.1218,0.2309,0.06915] |0.0 |[20.0,0.0] |[1.0,0.0] |0.0 |

# |8913|B |11.8 |13.12 |75.26 |431.9 |0.06955 |0.06232 |0.02853 |0.01638 |0.1337 |0.05581 |0.3438 |0.469 |2.225 |25.06 |0.004731 |0.01345 |0.01652 |0.005905 |0.01619 |0.002081 |13.45 |15.54 |86.0 |562.0 |0.09616 |0.1726 |0.1449 |0.05356 |0.2309 |0.06915 |[11.8,13.12,75.26,431.9,0.06955,0.06232,0.02853,0.01638,0.1337,0.05581,0.3438,0.469,2.225,25.06,0.004731,0.01345,0.01652,0.005905,0.01619,0.002081,13.45,15.54,86.0,562.0,0.09616,0.1726,0.1449,0.05356,0.2309,0.06915] |0.0 |[20.0,0.0] |[1.0,0.0] |0.0 |

# +----+---------+-----------+------------+--------------+---------+---------------+----------------+--------------+-------------------+-------------+----------------------+---------+----------+------------+-------+-------------+--------------+------------+-----------------+-----------+--------------------+------------+-------------+---------------+----------+----------------+-----------------+---------------+--------------------+--------------+-----------------------+-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+-----+-------------+-----------+----------+

####### Evaluate your accuracy of model using ACCURACY Metric

evaluator\_rf\_accuracy = MulticlassClassificationEvaluator(labelCol="label",

predictionCol="prediction",

metricName="accuracy")

evaluator\_rf\_auc\_accuracy = evaluator\_rf\_accuracy.evaluate(prediction\_rf)

rf\_accuracy\_rounded = round(evaluator\_rf\_auc\_accuracy \* 100, 2)

print()

print("Accuracy of Prediction using RF with Different Metrics....")

print("==========================================================")

print("Accuracy of RF using accuracy metrics = ", rf\_accuracy\_rounded)

### Evaluate your accuracy of model using 'weightedRecall' or F1 score

evaluator\_rf\_recall = MulticlassClassificationEvaluator(labelCol="label",

predictionCol="prediction",

metricName="weightedRecall")

evaluator\_rf\_recall\_accuracy = evaluator\_rf\_recall.evaluate(prediction\_rf)

rf\_recall\_rounded = round(evaluator\_rf\_recall\_accuracy \* 100, 2)

print("Accuracy of RF using FMeasure = ", rf\_recall\_rounded)

# Accuracy of Prediction using RF with Different Metrics....

# ==========================================================

# Accuracy of RF using accuracy metrics = 99.19

# Accuracy of RF using FMeasure = 99.19

# ############ Random Forest ends here

# # Create a Gradient Booster Tree and set the labelCol to 'label'

# ##### Important : GBT Model will not work for this project because the label column is not

# # binary. That is labels are 0.0 - Converted, 1.0 - Demented and 2.0 -Nondemented

# # This will throw py4j error exception

# #########

# # print()

# # display(HTML("<h3 style='color:green; text-align:left;'>Processing GBT Please Wait !!!</h3>"))

# # gbt = GBTClassifier(

# # maxDepth=20,

# # maxIter=50, # This corresponds to the number of trees in GBT

# # seed=5043

# # )

# # # # # Fit the model to your data

# # gbt\_model = gbt.fit(train\_data\_df)

########## Plot Scatter and Line Graph

# To create a scatter plot extract 'label' and 'prediction' from

# prediction\_df Dataframe, then convert it to pandas dataframe

# Create Actual scatter plot using the predicted value and labels

labels = prediction\_rf.select("label").toPandas()

predictions = prediction\_rf.select("prediction").toPandas()

# Create a actual scatter plot

plt.figure(figsize=(4, 3))

plt.scatter(predictions, labels)

plt.xlabel("Predictions")

plt.ylabel("Labels")

plt.title("Scatter Plot Actual")

plt.show()

# Create a line plot

plt.figure(figsize=(4, 3))

plt.plot(predictions, labels, marker='.', linestyle='-', color='red')

plt.xlabel("Predictions")

plt.ylabel("Labels")

plt.title("Line Plot")

plt.show()

# Draw a Voilin Plot

confu\_matrix = confusion\_matrix(labels, predictions.round())

plt.figure(figsize=(4, 3))

sns.violinplot(confu\_matrix, annot=True, fmt="d", cmap="Dark2")

plt.xlabel("Predicted Labels")

plt.ylabel("Actual Labels")

plt.title("Violin Plot, with Noise")

plt.show()

# Draw a Confusion Matrix

# cmap or colormap codes are, 'Set1', "Dark2", "Blues", "Accent", "Paired"

confu\_matrix = confusion\_matrix(labels, predictions.round())

plt.figure(figsize=(4, 3))

sns.heatmap(confu\_matrix, annot=True, fmt="d", cmap="Set1")

plt.xlabel("Predicted Labels")

plt.ylabel("Actual Labels")

plt.title("Confusion Matrix Heatmap, with Noise")

plt.show()

####### Filter out unmatched predictions and display scatter plot

# Filter out points not on the diagonal, that is remove erroneous label and

# prediction.

filtered\_df = prediction\_rf.filter(col("label") == col("prediction"))

labels = filtered\_df.select("label").toPandas()

predictions = filtered\_df.select("prediction").toPandas()

# Create a scatter plot using only label and prediction are equal

plt.figure(figsize=(4, 3))

plt.scatter(predictions, labels)

plt.xlabel("Predictions")

plt.ylabel("Labels")

plt.title("Scatter Plot, Noise Removed")

plt.show()

# Create a LINE plot using only label and prediction are equal

plt.figure(figsize=(4, 3))

plt.plot(predictions, labels, marker='.', linestyle='-', color='red')

plt.xlabel("Predictions")

plt.ylabel("Labels")

plt.title("Line Plot, Noise Removed")

plt.show()

# cmap or colormap codes are, 'Set1', "Dark2", "Blues", "Accent", "Paired"

# Draw a Voilin Plot

confu\_matrix = confusion\_matrix(labels, predictions.round())

plt.figure(figsize=(4, 3))

sns.violinplot(confu\_matrix, annot=True, fmt="d", cmap="Dark2")

plt.xlabel("Predicted Labels")

plt.ylabel("Actual Labels")

plt.title("Violin Plot, Noise Removed")

plt.show()

###### Draw a Confusion Matrix, How to interpret Confusion Matrix

# If you see a dark cell for "Class A" (actual) and "Class B" (predicted) and it's not on

# the diagonal, it means that the model is misclassifying instances of "Class A" as "Class B."

# In a well-performing model, you would generally want to see bright colors (indicating high

# counts) along the diagonal and lighter colors (indicating lower counts) elsewhere,

# suggesting that most instances are correctly classified. Darker colors away from the

# diagonal indicate areas where the model is making more errors.

#######

confu\_matrix = confusion\_matrix(labels, predictions.round())

plt.figure(figsize=(4, 3))

sns.heatmap(confu\_matrix, annot=True, fmt="d", cmap="Dark2")

plt.xlabel("Predicted Labels")

plt.ylabel("Actual Labels")

plt.title("Confusion Matrix Heatmap, Noise Removed")

plt.show()

# Cross verify, how many columns shows disimilarity

filtered\_df = prediction\_rf.filter(col("label") != col("prediction"))

print("Count of un Matched Predictions")

un\_matched = filtered\_df.count()

print("Un Matched Data...", un\_matched)

# Count of un Matched Predictions

# Un Matched Data... 37

# In the Confusion matrix the Correct predictions from test data is calculated as follows :-

# Tot Rows in Test Data = 4559

# Diagonal Total from Confusion Matrix = 4538

# False predictions = 4559 - 4524 = 35, Tallied with Un Matched Data....= 21

# Display unmatched data

print("Display Un Matched Data....")

filtered\_df.show(5, False)

# Display Un Matched Data....

# +------+---------+-----------+------------+--------------+---------+---------------+----------------+--------------+-------------------+-------------+----------------------+---------+----------+------------+-------+-------------+--------------+------------+-----------------+-----------+--------------------+------------+-------------+---------------+----------+----------------+-----------------+---------------+--------------------+--------------+-----------------------+----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+-----+-------------+-----------+----------+

# |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|features |label|rawPrediction|probability|prediction|

# +------+---------+-----------+------------+--------------+---------+---------------+----------------+--------------+-------------------+-------------+----------------------+---------+----------+------------+-------+-------------+--------------+------------+-----------------+-----------+--------------------+------------+-------------+---------------+----------+----------------+-----------------+---------------+--------------------+--------------+-----------------------+----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+-----+-------------+-----------+----------+

# |858479|M |14.25 |16.43 |96.42 |645.7 |0.08957 |0.1036 |0.1112 |0.0415 |0.161 |0.05594 |0.7036 |0.7874 |5.373 |60.78 |0.0112 |0.01891 |0.03638 |0.01367 |0.01718 |0.002851 |17.67 |19.66 |119.1 |959.5 |0.1109 |0.2403 |0.2863 |0.07164 |0.2506 |0.07628 |[14.25,16.43,96.42,645.7,0.08957,0.1036,0.1112,0.0415,0.161,0.05594,0.7036,0.7874,5.373,60.78,0.0112,0.01891,0.03638,0.01367,0.01718,0.002851,17.67,19.66,119.1,959.5,0.1109,0.2403,0.2863,0.07164,0.2506,0.07628] |1.0 |[11.0,9.0] |[0.55,0.45]|0.0 |

# |864033|B |13.48 |16.99 |88.4 |552.2 |0.1037 |0.1131 |0.09799 |0.07785 |0.1584 |0.07065 |0.213 |1.424 |1.545 |18.52 |0.01385 |0.02932 |0.02722 |0.01023 |0.03281 |0.004638 |14.34 |21.47 |106.0 |729.5 |0.1467 |0.2376 |0.2702 |0.1765 |0.2533 |0.08468 |[13.48,16.99,88.4,552.2,0.1037,0.1131,0.09799,0.07785,0.1584,0.07065,0.213,1.424,1.545,18.52,0.01385,0.02932,0.02722,0.01023,0.03281,0.004638,14.34,21.47,106.0,729.5,0.1467,0.2376,0.2702,0.1765,0.2533,0.08468] |0.0 |[0.0,20.0] |[0.0,1.0] |1.0 |

# |896864|B |16.84 |19.35 |108.4 |880.2 |0.09579 |0.04788 |0.03564 |0.02771 |0.1761 |0.0654 |0.4789 |0.5664 |3.479 |46.61 |0.005727 |0.03255 |0.04393 |0.009811 |0.02751 |0.004572 |18.22 |21.95 |120.3 |1032.0 |0.1288 |0.117 |0.22 |0.08268 |0.3596 |0.09166 |[16.84,19.35,108.4,880.2,0.09579,0.04788,0.03564,0.02771,0.1761,0.0654,0.4789,0.5664,3.479,46.61,0.005727,0.03255,0.04393,0.009811,0.02751,0.004572,18.22,21.95,120.3,1032.0,0.1288,0.117,0.22,0.08268,0.3596,0.09166]|0.0 |[3.0,17.0] |[0.15,0.85]|1.0 |

# |904457|M |14.22 |17.26 |94.37 |609.9 |0.09087 |0.1043 |0.09627 |0.05602 |0.1847 |0.06019 |0.286 |1.14 |2.112 |31.72 |0.005463 |0.01964 |0.02079 |0.005398 |0.01477 |0.003071 |15.75 |24.49 |107.4 |762.4 |0.1244 |0.2419 |0.201 |0.1452 |0.2779 |0.08121 |[14.22,17.26,94.37,609.9,0.09087,0.1043,0.09627,0.05602,0.1847,0.06019,0.286,1.14,2.112,31.72,0.005463,0.01964,0.02079,0.005398,0.01477,0.003071,15.75,24.49,107.4,762.4,0.1244,0.2419,0.201,0.1452,0.2779,0.08121] |1.0 |[13.0,7.0] |[0.65,0.35]|0.0 |

# |915296|B |13.3 |13.13 |85.24 |541.2 |0.1007 |0.1863 |0.2254 |0.06017 |0.1881 |0.06662 |0.2621 |1.339 |2.028 |20.98 |0.005733 |0.01777 |0.02101 |0.01164 |0.01801 |0.003721 |15.73 |18.24 |107.3 |840.7 |0.1316 |0.4462 |0.4897 |0.1508 |0.2591 |0.08709 |[13.3,13.13,85.24,541.2,0.1007,0.1863,0.2254,0.06017,0.1881,0.06662,0.2621,1.339,2.028,20.98,0.005733,0.01777,0.02101,0.01164,0.01801,0.003721,15.73,18.24,107.3,840.7,0.1316,0.4462,0.4897,0.1508,0.2591,0.08709] |0.0 |[5.0,15.0] |[0.25,0.75]|1.0 |

# +------+---------+-----------+------------+--------------+---------+---------------+----------------+--------------+-------------------+-------------+----------------------+---------+----------+------------+-------+-------------+--------------+------------+-----------------+-----------+--------------------+------------+-------------+---------------+----------+----------------+-----------------+---------------+--------------------+--------------+-----------------------+----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+-----+-------------+-----------+----------+

######## Save your rf\_model as 'Model\_ **Breast Cancer** \_RF', since the random forest shows greater

# Accuracy level

rf\_model.write().overwrite() \

.save("/content/drive/MyDrive/Model\_Breast\_Cancer\_RF")

print()

display(HTML("<h3 style='color:magenta; text-align:left;'>Model Building Finished...</h3>"))

# Record the end time after the process has finished

end\_time = datetime.datetime.now()

print("End Time H:M:S :", end\_time.strftime("%I:%M:%S %p")) # Format the datetime

# Calculate the time difference

time\_difference = end\_time - current\_time

# Print the time difference

formatted\_time\_difference = str(time\_difference).split('.')[0] # Remove microseconds

print("Time Difference :", formatted\_time\_difference)

# End Time H:M:S : 05:46:23 AM

# Time Difference : 0:00:41 41 Seconds to finish Build Operation

main\_function()

def single\_input\_and\_predict(event):

print()

clear\_result\_grid()

filename = '/content/drive/MyDrive/Single\_BCancer.csv'

headers = ['Patient\_Name',"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"]

css = """

<style>

.red-text .widget-label {

color: red !important;

font-size: 18px;

}

</style>

"""

# Define the input box or frame style

####### background-colors are

# "lightgray', "whitesmoke", "lavender", "mintcream", "aliceblue", "linen"

# "ivory", "honeydew", "floralwhite", "ghostwhite"

########### Define the CSS class with the desired style for frame

# In the code given below, the f-string is used to define the custom CSS (Cascading Style Sheets)

# for an input box. The variable input\_box\_style is expected to contain the CSS styling properties

# for the input box. The f-string allows the variable's value to be inserted directly into the string,

# resulting in a dynamic CSS style applied to the input box.

# The following display(HTML(css)), activate all css declarations

display(HTML(css))

# display(HTML("<h2 style='color:green; text-align:center;'>Input your Patient details</h2>"))

# Actual GUI descriptions for each required entry parameters declared here

Patient\_Name = widgets.Text(description =

'Patient Name>>>>>>>>',

style={'description\_width': 'initial'},

layout=widgets.Layout(

width='450px',

border='2px solid blue', # Blue, Add border style here

border\_radius='5px', # Add border-radius style here

padding='1px'))

# use placeholder to show an input example

radius\_mean\_ph = "Min=6.00, Max=28.112"

radius\_mean = widgets.Text(description =

'Tumour Radius Mean>>>>',

style={'description\_width': 'initial'},

layout=widgets.Layout(

width='400px',

border='2px solid blue',

border\_radius='5px',

padding='1px'),

placeholder=radius\_mean\_ph)

texture\_mean\_ph = "Min=9.50, Max=38.50"

texture\_mean = widgets.Text(description =

'Texture Mean>>>>>>>>>>>',

style={'description\_width': 'initial'},

layout=widgets.Layout(

width='400px',

border='2px solid blue',

border\_radius='5px',

padding='1px'),

placeholder=texture\_mean\_ph)

perimeter\_mean\_ph = "Min=43.50, Max=190.50"

perimeter\_mean = widgets.Text(description =

'Perimeter Mean of Tumour>',

style={'description\_width': 'initial'},

layout=widgets.Layout(

width='400px',

border='2px solid blue',

border\_radius='5px',

padding='1px'),

placeholder=perimeter\_mean\_ph)

area\_mean\_ph = "Min=140.50, Max=2500.50"

area\_mean = widgets.Text(description =

'Area Mean of Tumour>>>>',

style={'description\_width': 'initial'},

layout=widgets.Layout(

width='400px',

border='2px solid blue',

border\_radius='5px',

padding='1px'),

placeholder=area\_mean\_ph)

smoothness\_mean\_ph = "Min=0.06251, Max=0.1634"

smoothness\_mean = widgets.Text(description =

'Smoothness Tumour Boundry>',

style={'description\_width': 'initial'},

layout=widgets.Layout(

width='400px',

border='2px solid blue',

border\_radius='5px',

padding='1px'),

placeholder=smoothness\_mean\_ph)

# Data Arranged in Drop Down list,

# First value M, second value B, third value M, fourth value B,

# fifth value M, Sixth value B

compactness\_mean\_options = [0.187,0.04234,0.2022,0.06492,0.1436,0.07125]

compactness\_mean = widgets.Dropdown(description =

'Area & Perim. Distance>>>>',

style={'description\_width': 'initial'},

options=compactness\_mean\_options,

layout=widgets.Layout(

width='400px',

border='2px solid blue',

border\_radius='5px',

padding='1px'))

concavity\_mean\_options = [0.203,0.03383,0.1722,0.02956,0.09847,0.02451]

concavity\_mean = widgets.Dropdown(description =

'Area & Perim. Distance>>>>',

style={'description\_width': 'initial'},

options=concavity\_mean\_options,

layout=widgets.Layout(

width='400px',

border='2px solid blue',

border\_radius='5px',

padding='1px'))

concave\_points\_mean\_options = [0.0852,0.02292,0.1028,0.02076,0.06152,0.01708]

concave\_points\_mean = widgets.Dropdown(description =

'Avg.Num of concave points>',

style={'description\_width': 'initial'},

options=concave\_points\_mean\_options,

layout=widgets.Layout(

width='400px',

border='2px solid blue',

border\_radius='5px',

padding='1px'))

symmetry\_mean\_options = [0.1931,0.1337,0.1879,0.2128,0.2128,0.2095]

symmetry\_mean = widgets.Dropdown(description =

'Symmetry Mean of Tumour>>>',

style={'description\_width': 'initial'},

options=symmetry\_mean\_options,

layout=widgets.Layout(

width='400px',

border='2px solid blue',

border\_radius='5px',

padding='1px'))

fractal\_dimension\_mean\_options = [0.05796,0.05581,0.05852,0.06767,0.06777,0.05649]

fractal\_dimension\_mean = widgets.Dropdown(description =

'Tumour Complexity>>>>>>>',

style={'description\_width': 'initial'},

options=fractal\_dimension\_mean\_options,

layout=widgets.Layout(

width='400px',

border='2px solid blue',

border\_radius='5px',

padding='1px'))

radius\_se\_options = [0.3331,0.3661,0.5692,0.2773,0.7036,0.3118]

radius\_se = widgets.Dropdown(description =

'Radius Stdandard Error>>>>',

style={'description\_width': 'initial'},

options=radius\_se\_options,

layout=widgets.Layout(

width='400px',

border='2px solid blue',

border\_radius='5px',

padding='1px'))

texture\_se\_options = [0.7859,0.469,0.948,0.8937,0.8937,1.509]

texture\_se = widgets.Dropdown(description =

'Texture Stdandard Error>>>',

style={'description\_width': 'initial'},

options=texture\_se\_options,

layout=widgets.Layout(

width='400px',

border='2px solid blue',

border\_radius='5px',

padding='1px'))

perimeter\_se\_options = [2.937,2.41,3.854,1.909,5.373,2.0]

perimeter\_se = widgets.Dropdown(description =

'Perimeter Stdandard Error>>',

style={'description\_width': 'initial'},

options=perimeter\_se\_options,

layout=widgets.Layout(

width='400px',

border='2px solid blue',

border\_radius='5px',

padding='1px'))

area\_se\_options = [32.52,24.44,54.18,15.7,60.78,24.79]

area\_se = widgets.Dropdown(description =

'Area Stdandard Error>>>>>',

style={'description\_width': 'initial'},

options=area\_se\_options,

layout=widgets.Layout(

width='400px',

border='2px solid blue',

border\_radius='5px',

padding='1px'))

smoothness\_se\_options = [0.00624,0.004731,0.005332,0.006532,0.006532,0.006016]

smoothness\_se = widgets.Dropdown(description =

'Smoothness Stdandard Error>',

style={'description\_width': 'initial'},

options=smoothness\_se\_options,

layout=widgets.Layout(

width='400px',

border='2px solid blue',

border\_radius='5px',

padding='1px'))

compactness\_se\_options = [0.01484,0.01345,0.02115,0.02347,0.02336,0.03482]

compactness\_se = widgets.Dropdown(description =

'Compactness Std.Error>>>',

style={'description\_width': 'initial'},

options=compactness\_se\_options,

layout=widgets.Layout(

width='400px',

border='2px solid blue',

border\_radius='5px',

padding='1px'))

concavity\_se\_options = [0.02813,0.01652,0.01536,0.02645,0.02905,0.04232]

concavity\_se = widgets.Dropdown(description =

'Concavity Std.Error>>>>>>>>',

style={'description\_width': 'initial'},

options=concavity\_se\_options,

layout=widgets.Layout(

width='400px',

border='2px solid blue',

border\_radius='5px',

padding='1px'))

concave\_points\_se\_options = [0.01093,0.005905,0.01187,0.01243,0.01215,0.01269]

concave\_points\_se = widgets.Dropdown(description =

'Concave Points Std.Error>>>',

style={'description\_width': 'initial'},

options=concave\_points\_se\_options,

layout=widgets.Layout(

width='400px',

border='2px solid blue',

border\_radius='5px',

padding='1px'))

symmetry\_se\_options = [0.01397,0.01619,0.01522,0.01743,0.01743,0.02657]

symmetry\_se = widgets.Dropdown(description =

'Symmetry Std.Error>>>>>>',

style={'description\_width': 'initial'},

options=symmetry\_se\_options,

layout=widgets.Layout(

width='400px',

border='2px solid blue',

border\_radius='5px',

padding='1px'))

fractal\_dimension\_se\_options = [0.002461,0.002081,0.002815,0.003641,0.003643,0.004411]

fractal\_dimension\_se = widgets.Dropdown(description =

'Fractal Dimension Std.Err>>',

style={'description\_width': 'initial'},

options=fractal\_dimension\_se\_options,

layout=widgets.Layout(

width='400px',

border='2px solid blue',

border\_radius='5px',

padding='1px'))

radius\_worst\_options = [17.11,12.83,20.96,10.23,17.67,13.67]

radius\_worst = widgets.Dropdown(description =

'Radius Large>>>>>>>>>>>>>>>',

style={'description\_width': 'initial'},

options=radius\_worst\_options,

layout=widgets.Layout(

width='400px',

border='2px solid blue',

border\_radius='5px',

padding='1px'))

texture\_worst\_options = [26.0,15.54,26.19,27.95,27.95,31.59]

texture\_worst = widgets.Dropdown(description =

'Texture Large>>>>>>>>>>>>>>',

style={'description\_width': 'initial'},

options=texture\_worst\_options,

layout=widgets.Layout(

width='400px',

border='2px solid blue',

border\_radius='5px',

padding='1px'))

perimeter\_worst\_options = [117.7,82.14,136.8,65.13,119.1,87.54]

perimeter\_worst = widgets.Dropdown(description =

'Perimeter Large>>>>>>>>>',

style={'description\_width': 'initial'},

options=perimeter\_worst\_options,

layout=widgets.Layout(

width='400px',

border='2px solid blue',

border\_radius='5px',

padding='1px'))

area\_worst\_options = [909.4,495.2,1315.0,314.9,959.5,583.0]

area\_worst = widgets.Dropdown(description =

'Area Large>>>>>>>>>>>>>',

style={'description\_width': 'initial'},

options=area\_worst\_options,

layout=widgets.Layout(

width='400px',

border='2px solid blue',

border\_radius='5px',

padding='1px'))

smoothness\_worst\_options = [0.1546,0.09616,0.1313,0.1786,0.1786,0.119]

smoothness\_worst = widgets.Dropdown(description =

'Smoothness Large>>>>>>>',

style={'description\_width': 'initial'},

options=smoothness\_worst\_options,

layout=widgets.Layout(

width='400px',

border='2px solid blue',

border\_radius='5px',

padding='1px'))

compactness\_worst\_options = [0.4967,0.07866,0.4233,0.1148,0.3724,0.2308]

compactness\_worst = widgets.Dropdown(description =

'Compactness Large>>>>>>>>>>>',

style={'description\_width': 'initial'},

options=compactness\_worst\_options,

layout=widgets.Layout(

width='400px',

border='2px solid blue',

border\_radius='5px',

padding='1px'))

concavity\_worst\_options = [0.5911,0.1147,0.4784,0.08867,0.3664,0.1659]

concavity\_worst = widgets.Dropdown(description =

'Concavity Large>>>>>>>>>',

style={'description\_width': 'initial'},

options=concavity\_worst\_options,

layout=widgets.Layout(

width='400px',

border='2px solid blue',

border\_radius='5px',

padding='1px'))

concave\_points\_worst\_options = [0.2163,0.05104,0.2073,0.06227,0.1491,0.07432]

concave\_points\_worst = widgets.Dropdown(description =

'Concave Points Large>>>>>>>>',

style={'description\_width': 'initial'},

options=concave\_points\_worst\_options,

layout=widgets.Layout(

width='400px',

border='2px solid blue',

border\_radius='5px',

padding='1px'))

symmetry\_worst\_options = [0.2837,0.2309,0.2962,0.39,0.39,0.3689]

symmetry\_worst = widgets.Dropdown(description =

'Symmetry Large>>>>>>>>>',

style={'description\_width': 'initial'},

options=symmetry\_worst\_options,

layout=widgets.Layout(

width='400px',

border='2px solid blue',

border\_radius='5px',

padding='1px'))

fractal\_dimension\_worst\_options = [0.08019,0.06915,0.08472,0.1179,0.1179,0.08368]

fractal\_dimension\_worst = widgets.Dropdown(description =

'Fractal Dimension Large>>>>>',

style={'description\_width': 'initial'},

options=fractal\_dimension\_worst\_options,

layout=widgets.Layout(

width='400px',

border='2px solid blue',

border\_radius='5px',

padding='1px'))

# Add CSS classes declared above to the input widgets

Patient\_Name.add\_class('red-text')

radius\_mean.add\_class('red-text')

texture\_mean.add\_class('red-text')

perimeter\_mean.add\_class('red-text')

area\_mean.add\_class('red-text')

smoothness\_mean.add\_class('red-text')

compactness\_mean.add\_class('red-text')

concavity\_mean.add\_class('red-text')

concave\_points\_mean.add\_class('red-text')

symmetry\_mean.add\_class('red-text')

fractal\_dimension\_mean.add\_class('red-text')

radius\_se.add\_class('red-text')

texture\_se.add\_class('red-text')

perimeter\_se.add\_class('red-text')

area\_se.add\_class('red-text')

smoothness\_se.add\_class('red-text')

compactness\_se.add\_class('red-text')

concavity\_se.add\_class('red-text')

concave\_points\_se.add\_class('red-text')

symmetry\_se.add\_class('red-text')

fractal\_dimension\_se.add\_class('red-text')

radius\_worst.add\_class('red-text')

texture\_worst.add\_class('red-text')

perimeter\_worst.add\_class('red-text')

area\_worst.add\_class('red-text')

smoothness\_worst.add\_class('red-text')

compactness\_worst.add\_class('red-text')

concavity\_worst.add\_class('red-text')

concave\_points\_worst.add\_class('red-text')

symmetry\_worst.add\_class('red-text')

fractal\_dimension\_worst.add\_class('red-text')

# Display Data Entry GUI, The input columns are arranged pairwise

Line1\_container = widgets.HBox(children=[Patient\_Name, radius\_mean])

Line2\_container = widgets.HBox(children=[texture\_mean, perimeter\_mean])

Line3\_container = widgets.HBox(children=[area\_mean, smoothness\_mean])

Line4\_container = widgets.HBox(children=[compactness\_mean, concavity\_mean])

Line5\_container = widgets.HBox(children=[concave\_points\_mean, symmetry\_mean])

Line6\_container = widgets.HBox(children=[fractal\_dimension\_mean, radius\_se])

Line7\_container = widgets.HBox(children=[texture\_se, perimeter\_se])

Line8\_container = widgets.HBox(children=[area\_se, smoothness\_se])

Line9\_container = widgets.HBox(children=[compactness\_se, concavity\_se])

Line10\_container = widgets.HBox(children=[concave\_points\_se, symmetry\_se])

Line11\_container = widgets.HBox(children=[fractal\_dimension\_se, radius\_worst])

Line12\_container = widgets.HBox(children=[texture\_worst, perimeter\_worst])

Line13\_container = widgets.HBox(children=[area\_worst, smoothness\_worst])

Line14\_container = widgets.HBox(children=[compactness\_worst, concavity\_worst])

Line15\_container = widgets.HBox(children=[concave\_points\_worst, symmetry\_worst])

Line16\_container = widgets.HBox(children=[fractal\_dimension\_worst])

# Display the above data entry fields

display(Line1\_container), display(Line2\_container)

display(Line3\_container), display(Line4\_container)

display(Line5\_container), display(Line6\_container)

display(Line7\_container), display(Line8\_container)

display(Line9\_container), display(Line10\_container)

display(Line11\_container), display(Line12\_container)

display(Line13\_container), display(Line14\_container)

display(Line15\_container), display(Line16\_container)

# Display Main Heading

# display(HTML("<h2 style='color:green; text-align:center;'>Prediction using Single Input</h2>"))

# print()

# Declare your buttons

save\_button\_single = widgets.Button(description="Save Data")

predict\_button\_single = widgets.Button(description="Predict Result")

exit\_button\_single = widgets.Button(description="Go Back to Main Menu")

# Apply HTML/CSS for the above buttons

custom\_css\_single = """

<style>

.custom-button-single {

width: 250px; /\* Change this value for width \*/

height: 50px; /\* Change this value for height \*/

color: blue; /\* Text color \*/

background-color: yellow; /\* Background color \*/

border: 2px solid black;

}

/\* Add a mouse hover effect \*/

.custom-button-single:hover {

background-color: white; /\* Change background color on mouse hover \*/

}

</style>

"""

# Attach the above style sheet to each buttons

save\_button\_single.add\_class("custom-button-single")

predict\_button\_single.add\_class("custom-button-single")

exit\_button\_single.add\_class("custom-button-single")

# Display the custom buttons horizontally

display(HTML(custom\_css\_single))

buttons\_horizontal\_layout\_single = widgets.HBox([save\_button\_single,predict\_button\_single,

exit\_button\_single])

# # Align the buttons using the following

align\_layout\_single = widgets.Layout(justify\_content='center')

# Arrange buttons horizontally, in center of the output area

left\_container\_single = widgets.Box([buttons\_horizontal\_layout\_single],

layout=align\_layout\_single)

display(left\_container\_single)

print()

def on\_button\_clicked(save\_button\_single):

# # Assign values from text boxes to variables

entered\_Patient\_Name = Patient\_Name.value

# Since radius\_mean is a text input convert it to float

entered\_radius\_mean = radius\_mean.value

entered\_radius\_mean = float(entered\_radius\_mean)

entered\_texture\_mean = texture\_mean.value

entered\_texture\_mean = float(entered\_texture\_mean)

entered\_perimeter\_mean = perimeter\_mean.value

entered\_perimeter\_mean = float(entered\_perimeter\_mean)

entered\_area\_mean = area\_mean.value

entered\_area\_mean = float(entered\_area\_mean)

entered\_smoothness\_mean = smoothness\_mean.value

entered\_smoothness\_mean = float(entered\_smoothness\_mean)

# The following entries came directly from dropdown box, so no need to convert

# it as float

entered\_compactness\_mean = compactness\_mean.value

entered\_concavity\_mean = concavity\_mean.value

entered\_concave\_points\_mean = concave\_points\_mean.value

entered\_symmetry\_mean = symmetry\_mean.value

entered\_fractal\_dimension\_mean = fractal\_dimension\_mean.value

entered\_radius\_se = radius\_se.value

entered\_texture\_se = texture\_se.value

entered\_perimeter\_se = perimeter\_se.value

entered\_area\_se = area\_se.value

entered\_smoothness\_se = smoothness\_se.value

entered\_compactness\_se = compactness\_se.value

entered\_concavity\_se = concavity\_se.value

entered\_concave\_points\_se = concave\_points\_se.value

entered\_symmetry\_se = symmetry\_se.value

entered\_fractal\_dimension\_se = fractal\_dimension\_se.value

entered\_radius\_worst = radius\_worst.value

entered\_texture\_worst = texture\_worst.value

entered\_perimeter\_worst = perimeter\_worst.value

entered\_area\_worst = area\_worst.value

entered\_smoothness\_worst = smoothness\_worst.value

entered\_compactness\_worst = compactness\_worst.value

entered\_concavity\_worst = concavity\_worst.value

entered\_concave\_points\_worst = concave\_points\_worst.value

entered\_symmetry\_worst = symmetry\_worst.value

entered\_fractal\_dimension\_worst = fractal\_dimension\_worst.value

# # Save your data as csv file 'Single\_BCancer.csv'

with open(filename, 'w', newline='') as f:

writer = csv.writer(f)

writer.writerow(headers)

writer.writerow([entered\_Patient\_Name, entered\_radius\_mean,entered\_texture\_mean,

entered\_perimeter\_mean, entered\_area\_mean, entered\_smoothness\_mean,

entered\_compactness\_mean, entered\_concavity\_mean, entered\_concave\_points\_mean,

entered\_symmetry\_mean, entered\_fractal\_dimension\_mean, entered\_radius\_se,

entered\_texture\_se, entered\_perimeter\_se, entered\_area\_se, entered\_smoothness\_se,

entered\_compactness\_se, entered\_concavity\_se, entered\_concave\_points\_se,

entered\_symmetry\_se, entered\_fractal\_dimension\_se, entered\_radius\_worst,

entered\_texture\_worst, entered\_perimeter\_worst, entered\_area\_worst,

entered\_smoothness\_worst, entered\_compactness\_worst, entered\_concavity\_worst,

entered\_concave\_points\_worst, entered\_symmetry\_worst,

entered\_fractal\_dimension\_worst

])

# Cross verify

print('Data has been written to', filename)

save\_button\_single.on\_click(on\_button\_clicked)

predict\_button\_single.on\_click(predict\_single)

exit\_button\_single.on\_click(go\_back\_function)

def predict\_single(event):

print("Control in single input predict...")

clear\_result\_grid()

print()

display(HTML("<h3 style='color:orange; text-align:left;'>Single Input Data Prediction Started Please...wait !!!</h3>"))

# # Read Single Input Data for prediction

single\_df\_from\_csv = my\_spark.read \

.format("csv") \

.option("header", "True") \

.option("inferSchema", "True") \

.load("/content/drive/MyDrive/Single\_BCancer.csv")

# print("Schema of single\_df\_from\_csv")

# single\_df\_from\_csv.printSchema()

# Schema of single\_df\_from\_csv

# root

# |-- Patient\_Name: string (nullable = true)

# |-- radius\_mean: double (nullable = true)

# |-- texture\_mean: double (nullable = true)

# |-- perimeter\_mean: double (nullable = true)

# |-- area\_mean: double (nullable = true)

# |-- smoothness\_mean: double (nullable = true)

# |-- compactness\_mean: double (nullable = true)

# |-- concavity\_mean: double (nullable = true)

# |-- concave\_points\_mean: double (nullable = true)

# |-- symmetry\_mean: double (nullable = true)

# |-- fractal\_dimension\_mean: double (nullable = true)

# |-- radius\_se: double (nullable = true)

# |-- texture\_se: double (nullable = true)

# |-- perimeter\_se: double (nullable = true)

# |-- area\_se: double (nullable = true)

# |-- smoothness\_se: double (nullable = true)

# |-- compactness\_se: double (nullable = true)

# |-- concavity\_se: double (nullable = true)

# |-- concave\_points\_se: double (nullable = true)

# |-- symmetry\_se: double (nullable = true)

# |-- fractal\_dimension\_se: double (nullable = true)

# |-- radius\_worst: double (nullable = true)

# |-- texture\_worst: double (nullable = true)

# |-- perimeter\_worst: double (nullable = true)

# |-- area\_worst: double (nullable = true)

# |-- smoothness\_worst: double (nullable = true)

# |-- compactness\_worst: double (nullable = true)

# |-- concavity\_worst: double (nullable = true)

# |-- concave\_points\_worst: double (nullable = true)

# |-- symmetry\_worst: double (nullable = true)

# |-- fractal\_dimension\_worst: double (nullable = true)

# print("Data in single\_df\_from\_csv")

# single\_df\_from\_csv.show()

# Data in single\_df\_from\_csv

# +------------+-----------+------------+--------------+---------+---------------+----------------+--------------+-------------------+-------------+----------------------+---------+----------+------------+-------+-------------+--------------+------------+-----------------+-----------+--------------------+------------+-------------+---------------+----------+----------------+-----------------+---------------+--------------------+--------------+-----------------------+

# |Patient\_Name|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|

# +------------+-----------+------------+--------------+---------+---------------+----------------+--------------+-------------------+-------------+----------------------+---------+----------+------------+-------+-------------+--------------+------------+-----------------+-----------+--------------------+------------+-------------+---------------+----------+----------------+-----------------+---------------+--------------------+--------------+-----------------------+

# | JIYA| 10.0| 15.0| 100.0| 1000.0| 0.00625| 0.04234| 0.03383| 0.02292| 0.1337| 0.05581| 0.3661| 0.469| 2.41| 24.44| 0.004731| 0.01345| 0.01652| 0.005905| 0.01619| 0.002081| 12.83| 15.54| 82.14| 495.2| 0.09616| 0.07866| 0.1147| 0.05104| 0.2309| 0.06915|

# +------------+-----------+------------+--------------+---------+---------------+----------------+--------------+-------------------+-------------+----------------------+---------+----------+------------+-------+-------------+--------------+------------+-----------------+-----------+--------------------+------------+-------------+---------------+----------+----------------+-----------------+---------------+--------------------+--------------+-----------------------+

# Select your feature columns for prediction

single\_feature\_cols = ["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"]

single\_output\_col = "features"

# Vectorize the above feature\_cols and add it as a list in df\_final\_transformed

single\_assembler = VectorAssembler(inputCols=single\_feature\_cols, outputCol=single\_output\_col)

single\_df\_final\_vectors = single\_assembler.transform(single\_df\_from\_csv)

# Cross Verify features data

# print("Display few Data in single\_df\_final\_vectors")

# single\_df\_final\_vectors.show(5, False)

# Display few Data in single\_df\_final\_vectors

# +------------+-----------+------------+--------------+---------+---------------+----------------+--------------+-------------------+-------------+----------------------+---------+----------+------------+-------+-------------+--------------+------------+-----------------+-----------+--------------------+------------+-------------+---------------+----------+----------------+-----------------+---------------+--------------------+--------------+-----------------------+------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+

# |Patient\_Name|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|features |

# +------------+-----------+------------+--------------+---------+---------------+----------------+--------------+-------------------+-------------+----------------------+---------+----------+------------+-------+-------------+--------------+------------+-----------------+-----------+--------------------+------------+-------------+---------------+----------+----------------+-----------------+---------------+--------------------+--------------+-----------------------+------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+

# |JIYA |10.0 |15.0 |100.0 |1000.0 |0.00625 |0.04234 |0.03383 |0.02292 |0.1337 |0.05581 |0.3661 |0.469 |2.41 |24.44 |0.004731 |0.01345 |0.01652 |0.005905 |0.01619 |0.002081 |12.83 |15.54 |82.14 |495.2 |0.09616 |0.07866 |0.1147 |0.05104 |0.2309 |0.06915 |[10.0,15.0,100.0,1000.0,0.00625,0.04234,0.03383,0.02292,0.1337,0.05581,0.3661,0.469,2.41,24.44,0.004731,0.01345,0.01652,0.005905,0.01619,0.002081,12.83,15.54,82.14,495.2,0.09616,0.07866,0.1147,0.05104,0.2309,0.06915]|

# +------------+-----------+------------+--------------+---------+---------------+----------------+--------------+-------------------+-------------+----------------------+---------+----------+------------+-------+-------------+--------------+------------+-----------------+-----------+--------------------+------------+-------------+---------------+----------+----------------+-----------------+---------------+--------------------+--------------+-----------------------+------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+

# Load your saved model from "/content/drive/MyDrive/Model\_Breast\_Cancer\_RF"

loaded\_model = RandomForestClassificationModel \

.load("/content/drive/MyDrive/Model\_Breast\_Cancer\_RF")

# predict breast cancer detected or not, prediction = 1 Melignant,prediction = 0 Benign

single\_input\_prediction = loaded\_model.transform(single\_df\_final\_vectors)

# print("Prediction Output using Single user input...")

# single\_input\_prediction.show()

# Prediction Output using Single user input... No Tumour Detected

# +------------+-----------+------------+--------------+---------+---------------+----------------+--------------+-------------------+-------------+----------------------+---------+----------+------------+-------+-------------+--------------+------------+-----------------+-----------+--------------------+------------+-------------+---------------+----------+----------------+-----------------+---------------+--------------------+--------------+-----------------------+--------------------+-------------+-----------+----------+

# |Patient\_Name|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| features|rawPrediction|probability|prediction|

# +------------+-----------+------------+--------------+---------+---------------+----------------+--------------+-------------------+-------------+----------------------+---------+----------+------------+-------+-------------+--------------+------------+-----------------+-----------+--------------------+------------+-------------+---------------+----------+----------------+-----------------+---------------+--------------------+--------------+-----------------------+--------------------+-------------+-----------+----------+

# | JIYA| 10.0| 15.0| 100.0| 1000.0| 0.00625| 0.04234| 0.03383| 0.02292| 0.1337| 0.05581| 0.3661| 0.469| 2.41| 24.44| 0.004731| 0.01345| 0.01652| 0.005905| 0.01619| 0.002081| 12.83| 15.54| 82.14| 495.2| 0.09616| 0.07866| 0.1147| 0.05104| 0.2309| 0.06915|[10.0,15.0,100.0,...| [19.0,1.0]|[0.95,0.05]| 0.0|

# +------------+-----------+------------+--------------+---------+---------------+----------------+--------------+-------------------+-------------+----------------------+---------+----------+------------+-------+-------------+--------------+------------+-----------------+-----------+--------------------+------------+-------------+---------------+----------+----------------+-----------------+---------------+--------------------+--------------+-----------------------+--------------------+-------------+-----------+----------+

# Display Prediction Result as 'Affected / Not Affected'

selected\_df = single\_input\_prediction.withColumn("Final\_Output",

when(single\_input\_prediction["prediction"] == 1, "Tumour Detected")

.otherwise("No Tumour Detected"))

selected\_df\_final = selected\_df.select("Patient\_Name","Final\_Output")

########## Display prediction results in color frame

# Convert final\_prediction to Pandas DataFrame

pandas\_df = selected\_df\_final.toPandas()

# Apply color to the DataFrame output

# The applymap method is used to apply a function to each element of the DataFrame

# the lambda function in this code snippet is used to assign the CSS style 'color: red'

# to the cells in the 'Price' column of the DataFrame when generating its styled representation.

# background colors, 'red','green','blue','yellow','orange','purple','pink'

styled\_df = pandas\_df.style.applymap(lambda x: 'background-color: aliceblue',

subset=["Patient\_Name","Final\_Output"])

# Generate an HTML table(html\_table) with customized CSS

# 'th' = Table Heading, 'td' = Table Data

html\_table = styled\_df.hide(axis="index").set\_table\_styles(

[{'selector': 'th', 'props':[('color', 'red'), ('padding', '5px')]},

{'selector': 'td', 'props': [('vertical-align', 'middle'),('padding', '5px')]}

]).to\_html()

# Draw a frame around your output and Display the contents

# the f prefix enables you to embed expressions within a string by enclosing them in curly braces {}

# Within the string, you have embedded an expression {html\_table}.

# The value of the html\_table variable is dynamically inserted into the string at that location, that

# values are nothing but your prediction results with headings.

output\_frame = f"""

<div style="border: 2px solid black; padding: 10px;width: 50%; background-color: lightgrey;">

<h3 style='color:blue; text-align:left;'>Final Prediction as per your Input</h3>

{html\_table}

</div>

"""

display(HTML(output\_frame))

# Go Back to Sub Menu

button = widgets.Button(description="Click Here to Go Back")

button.on\_click(on\_button\_click\_goback)

# Display the button

display(widgets.HBox([button]))

def on\_button\_click\_goback(event):

# Go back to the Sub menu

single\_input\_and\_predict(event)

def bulk\_data\_load(event):

print()

clear\_result\_grid()

set\_focus\_cell2()

# Declare your upload buttons

browse\_button = widgets.Button(description="Click this and select Choose Files")

predict\_button = widgets.Button(description="Predict Result")

exit\_button = widgets.Button(description="Go Back to Main Menu")

# Apply HTML/CSS for the above buttons

custom\_css\_bulk = """

<style>

.custom-button-bulk {

width: 250px; /\* Change this value for width \*/

height: 50px; /\* Change this value for height \*/

color: blue; /\* Text color \*/

background-color: yellow; /\* Background color \*/

border: 2px solid black;

}

/\* Add a mouse hover effect \*/

.custom-button-bulk:hover {

background-color: white; /\* Change background color on mouse hover \*/

}

</style>

"""

# Attach the above style sheet to each buttons

browse\_button.add\_class("custom-button-bulk")

predict\_button.add\_class("custom-button-bulk")

exit\_button.add\_class("custom-button-bulk")

# Display the custom button

display(HTML(custom\_css\_bulk))

buttons\_horizontal\_layout\_1 = widgets.HBox([browse\_button,predict\_button,exit\_button])

# # Align the buttons using the following

align\_layout = widgets.Layout(justify\_content='center')

# Arrange buttons horizontally, in center of the output area

left\_container\_1 = widgets.Box([buttons\_horizontal\_layout\_1], layout=align\_layout)

print()

print()

print()

print()

display(HTML("<h3 style='color:magenta; text-align:center;'>Bulk Input Data Prediction Sub Menu</h3>"))

print()

display(left\_container\_1)

print()

print()

print()

print()

# The !mv command is used to copy dataset from 'sample\_data' folder inside

# google drive to /content/drive/MyDive. Other wise after 48 hrs

# the data in sample\_data will be removed.

# The dollar sign $ is used in the ! shell command execution within

# Google Colab to indicate that you're referencing a variable within

# a string. When you use $variable\_name within a shell command

# executed in a code cell, it is replaced with the value of the variable.

# The 'selected\_file' after 'files.upload' will contain a key,value pair

# To extract the file name, for loop is

# used to get the key, in other words actual file name.

def upload\_file\_fn(event):

selected\_file = files.upload()

for dataset\_name in selected\_file.keys():

source\_path = f"/content/{dataset\_name}"

destination\_path = f"/content/drive/MyDrive/{dataset\_name}"

!mv "$source\_path" "$destination\_path"

browse\_button.on\_click(upload\_file\_fn)

predict\_button.on\_click(bulk\_predict\_fn)

exit\_button.on\_click(go\_back\_function)

# 'exit\_function' is used separately because you cannot call 'main\_function()' directly

# in on\_click event. '0 argument expected but 1 is given' error will be flagged

def bulk\_predict\_fn(event):

# Read Bulk Input Data for prediction

clear\_result\_grid()

print()

display(HTML("<h3 style='color:orange; text-align:left;'>Bulk Input Data Prediction Started Please...wait !!!</h3>"))

bulk\_df\_from\_csv = my\_spark.read \

.format("csv") \

.option("header", "True") \

.option("inferSchema", "True") \

.load("/content/drive/MyDrive/Bulk\_BCancer.csv")

# print("Schema of df\_from\_CSV")

# bulk\_df\_from\_csv.printSchema()

# Schema of df\_from\_CSV

# root

# |-- Patient\_Name: string (nullable = true)

# |-- radius\_mean: double (nullable = true)

# |-- texture\_mean: double (nullable = true)

# |-- perimeter\_mean: double (nullable = true)

# |-- area\_mean: double (nullable = true)

# |-- smoothness\_mean: double (nullable = true)

# |-- compactness\_mean: double (nullable = true)

# |-- concavity\_mean: double (nullable = true)

# |-- concave\_points\_mean: double (nullable = true)

# |-- symmetry\_mean: double (nullable = true)

# |-- fractal\_dimension\_mean: double (nullable = true)

# |-- radius\_se: double (nullable = true)

# |-- texture\_se: double (nullable = true)

# |-- perimeter\_se: double (nullable = true)

# |-- area\_se: double (nullable = true)

# |-- smoothness\_se: double (nullable = true)

# |-- compactness\_se: double (nullable = true)

# |-- concavity\_se: double (nullable = true)

# |-- concave\_points\_se: double (nullable = true)

# |-- symmetry\_se: double (nullable = true)

# |-- fractal\_dimension\_se: double (nullable = true)

# |-- radius\_worst: double (nullable = true)

# |-- texture\_worst: double (nullable = true)

# |-- perimeter\_worst: double (nullable = true)

# |-- area\_worst: double (nullable = true)

# |-- smoothness\_worst: double (nullable = true)

# |-- compactness\_worst: double (nullable = true)

# |-- concavity\_worst: double (nullable = true)

# |-- concave\_points\_worst: double (nullable = true)

# |-- symmetry\_worst: double (nullable = true)

# |-- fractal\_dimension\_worst: double (nullable = true)

# print("Data in bulk\_df\_from")

# bulk\_df\_from\_csv.show(5, False)

# Data in bulk\_df\_from - along with the Patient\_Name, 'M' = Melignant, B=Benign

# +------------+-----------+------------+--------------+---------+---------------+----------------+--------------+-------------------+-------------+----------------------+---------+----------+------------+-------+-------------+--------------+------------+-----------------+-----------+--------------------+------------+-------------+---------------+----------+----------------+-----------------+---------------+--------------------+--------------+-----------------------+

# |Patient\_Name|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|

# +------------+-----------+------------+--------------+---------+---------------+----------------+--------------+-------------------+-------------+----------------------+---------+----------+------------+-------+-------------+--------------+------------+-----------------+-----------+--------------------+------------+-------------+---------------+----------+----------------+-----------------+---------------+--------------------+--------------+-----------------------+

# |SHEENA-M |15.46 |19.48 |103.8 |731.3 |0.1092 |0.187 |0.203 |0.0852 |0.1931 |0.05796 |0.3331 |0.7859 |2.937 |32.52 |0.00624 |0.01484 |0.02813 |0.01093 |0.01397 |0.002461 |17.11 |26.0 |117.7 |909.4 |0.1546 |0.4967 |0.5911 |0.2163 |0.2837 |0.08019 |

# |ROSAMMA-B |12.18 |13.12 |77.79 |451.1 |0.06955 |0.04234 |0.03383 |0.02292 |0.1337 |0.05581 |0.3661 |0.469 |2.41 |24.44 |0.004731 |0.01345 |0.01652 |0.005905 |0.01619 |0.002081 |12.83 |15.54 |82.14 |495.2 |0.09616 |0.07866 |0.1147 |0.05104 |0.2309 |0.06915 |

# |ANUSREE-M |16.13 |19.1 |108.1 |798.8 |0.08992 |0.2022 |0.1722 |0.1028 |0.1879 |0.05852 |0.5692 |0.948 |3.854 |54.18 |0.005332 |0.02115 |0.01536 |0.01187 |0.01522 |0.002815 |20.96 |26.19 |136.8 |1315.0 |0.1313 |0.4233 |0.4784 |0.2073 |0.2962 |0.08472 |

# |AKHILA-B |9.504 |18.66 |60.34 |273.9 |0.1158 |0.06492 |0.02956 |0.02076 |0.2128 |0.06767 |0.2773 |0.8937 |1.909 |15.7 |0.006532 |0.02347 |0.02645 |0.01243 |0.01743 |0.003641 |10.23 |27.95 |65.13 |314.9 |0.1786 |0.1148 |0.08867 |0.06227 |0.39 |0.1179 |

# |ATHIRA-M |14.25 |18.66 |96.42 |645.7 |0.1158 |0.1436 |0.09847 |0.06152 |0.2128 |0.06777 |0.7036 |0.8937 |5.373 |60.78 |0.006532 |0.02336 |0.02905 |0.01215 |0.01743 |0.003643 |17.67 |27.95 |119.1 |959.5 |0.1786 |0.3724 |0.3664 |0.1491 |0.39 |0.1179 |

# +------------+-----------+------------+--------------+---------+---------------+----------------+--------------+-------------------+-------------+----------------------+---------+----------+------------+-------+-------------+--------------+------------+-----------------+-----------+--------------------+------------+-------------+---------------+----------+----------------+-----------------+---------------+--------------------+--------------+-----------------------+

# Select your feature columns for prediction

bulk\_feature\_cols = ["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"]

bulk\_output\_col = "features"

# Vectorize the above feature\_cols and add it as a list in df\_final\_transformed

bulk\_assembler = VectorAssembler(inputCols=bulk\_feature\_cols, outputCol=bulk\_output\_col)

bulk\_df\_final\_vectors = bulk\_assembler.transform(bulk\_df\_from\_csv)

# Cross Verify features data

# print("Display few Data in bulk\_df\_final\_vectors")

# bulk\_df\_final\_vectors.show(5, False)

# Display few Data in bulk\_df\_final\_vectors

# +------------+-----------+------------+--------------+---------+---------------+----------------+--------------+-------------------+-------------+----------------------+---------+----------+------------+-------+-------------+--------------+------------+-----------------+-----------+--------------------+------------+-------------+---------------+----------+----------------+-----------------+---------------+--------------------+--------------+-----------------------+-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+

# |Patient\_Name|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|features |

# +------------+-----------+------------+--------------+---------+---------------+----------------+--------------+-------------------+-------------+----------------------+---------+----------+------------+-------+-------------+--------------+------------+-----------------+-----------+--------------------+------------+-------------+---------------+----------+----------------+-----------------+---------------+--------------------+--------------+-----------------------+-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+

# |SHEENA-M |15.46 |19.48 |103.8 |731.3 |0.1092 |0.187 |0.203 |0.0852 |0.1931 |0.05796 |0.3331 |0.7859 |2.937 |32.52 |0.00624 |0.01484 |0.02813 |0.01093 |0.01397 |0.002461 |17.11 |26.0 |117.7 |909.4 |0.1546 |0.4967 |0.5911 |0.2163 |0.2837 |0.08019 |[15.46,19.48,103.8,731.3,0.1092,0.187,0.203,0.0852,0.1931,0.05796,0.3331,0.7859,2.937,32.52,0.00624,0.01484,0.02813,0.01093,0.01397,0.002461,17.11,26.0,117.7,909.4,0.1546,0.4967,0.5911,0.2163,0.2837,0.08019] |

# |ROSAMMA-B |12.18 |13.12 |77.79 |451.1 |0.06955 |0.04234 |0.03383 |0.02292 |0.1337 |0.05581 |0.3661 |0.469 |2.41 |24.44 |0.004731 |0.01345 |0.01652 |0.005905 |0.01619 |0.002081 |12.83 |15.54 |82.14 |495.2 |0.09616 |0.07866 |0.1147 |0.05104 |0.2309 |0.06915 |[12.18,13.12,77.79,451.1,0.06955,0.04234,0.03383,0.02292,0.1337,0.05581,0.3661,0.469,2.41,24.44,0.004731,0.01345,0.01652,0.005905,0.01619,0.002081,12.83,15.54,82.14,495.2,0.09616,0.07866,0.1147,0.05104,0.2309,0.06915]|

# |ANUSREE-M |16.13 |19.1 |108.1 |798.8 |0.08992 |0.2022 |0.1722 |0.1028 |0.1879 |0.05852 |0.5692 |0.948 |3.854 |54.18 |0.005332 |0.02115 |0.01536 |0.01187 |0.01522 |0.002815 |20.96 |26.19 |136.8 |1315.0 |0.1313 |0.4233 |0.4784 |0.2073 |0.2962 |0.08472 |[16.13,19.1,108.1,798.8,0.08992,0.2022,0.1722,0.1028,0.1879,0.05852,0.5692,0.948,3.854,54.18,0.005332,0.02115,0.01536,0.01187,0.01522,0.002815,20.96,26.19,136.8,1315.0,0.1313,0.4233,0.4784,0.2073,0.2962,0.08472] |

# |AKHILA-B |9.504 |18.66 |60.34 |273.9 |0.1158 |0.06492 |0.02956 |0.02076 |0.2128 |0.06767 |0.2773 |0.8937 |1.909 |15.7 |0.006532 |0.02347 |0.02645 |0.01243 |0.01743 |0.003641 |10.23 |27.95 |65.13 |314.9 |0.1786 |0.1148 |0.08867 |0.06227 |0.39 |0.1179 |[9.504,18.66,60.34,273.9,0.1158,0.06492,0.02956,0.02076,0.2128,0.06767,0.2773,0.8937,1.909,15.7,0.006532,0.02347,0.02645,0.01243,0.01743,0.003641,10.23,27.95,65.13,314.9,0.1786,0.1148,0.08867,0.06227,0.39,0.1179] |

# |ATHIRA-M |14.25 |18.66 |96.42 |645.7 |0.1158 |0.1436 |0.09847 |0.06152 |0.2128 |0.06777 |0.7036 |0.8937 |5.373 |60.78 |0.006532 |0.02336 |0.02905 |0.01215 |0.01743 |0.003643 |17.67 |27.95 |119.1 |959.5 |0.1786 |0.3724 |0.3664 |0.1491 |0.39 |0.1179 |[14.25,18.66,96.42,645.7,0.1158,0.1436,0.09847,0.06152,0.2128,0.06777,0.7036,0.8937,5.373,60.78,0.006532,0.02336,0.02905,0.01215,0.01743,0.003643,17.67,27.95,119.1,959.5,0.1786,0.3724,0.3664,0.1491,0.39,0.1179] |

# +------------+-----------+------------+--------------+---------+---------------+----------------+--------------+-------------------+-------------+----------------------+---------+----------+------------+-------+-------------+--------------+------------+-----------------+-----------+--------------------+------------+-------------+---------------+----------+----------------+-----------------+---------------+--------------------+--------------+-----------------------+-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+

# Load your saved model from "/content/drive/MyDrive/Model\_Breast\_Cancer\_RF"

loaded\_model = RandomForestClassificationModel \

.load("/content/drive/MyDrive/Model\_Breast\_Cancer\_RF")

# predict Breast Cancer detected or not, prediction = 1 Malignant,prediction = 0 Benign

bulk\_input\_prediction = loaded\_model.transform(bulk\_df\_final\_vectors)

# print("Prediction Output using Bulk user input...")

# bulk\_input\_prediction.show(5, False)

# Prediction Output using Bulk user input...

# +------------+-----------+------------+--------------+---------+---------------+----------------+--------------+-------------------+-------------+----------------------+---------+----------+------------+-------+-------------+--------------+------------+-----------------+-----------+--------------------+------------+-------------+---------------+----------+----------------+-----------------+---------------+--------------------+--------------+-----------------------+-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+-------------+-----------+----------+

# |Patient\_Name|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|features |rawPrediction|probability|prediction|

# +------------+-----------+------------+--------------+---------+---------------+----------------+--------------+-------------------+-------------+----------------------+---------+----------+------------+-------+-------------+--------------+------------+-----------------+-----------+--------------------+------------+-------------+---------------+----------+----------------+-----------------+---------------+--------------------+--------------+-----------------------+-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+-------------+-----------+----------+

# |SHEENA-M |15.46 |19.48 |103.8 |731.3 |0.1092 |0.187 |0.203 |0.0852 |0.1931 |0.05796 |0.3331 |0.7859 |2.937 |32.52 |0.00624 |0.01484 |0.02813 |0.01093 |0.01397 |0.002461 |17.11 |26.0 |117.7 |909.4 |0.1546 |0.4967 |0.5911 |0.2163 |0.2837 |0.08019 |[15.46,19.48,103.8,731.3,0.1092,0.187,0.203,0.0852,0.1931,0.05796,0.3331,0.7859,2.937,32.52,0.00624,0.01484,0.02813,0.01093,0.01397,0.002461,17.11,26.0,117.7,909.4,0.1546,0.4967,0.5911,0.2163,0.2837,0.08019] |[0.0,20.0] |[0.0,1.0] |1.0 |

# |ROSAMMA-B |12.18 |13.12 |77.79 |451.1 |0.06955 |0.04234 |0.03383 |0.02292 |0.1337 |0.05581 |0.3661 |0.469 |2.41 |24.44 |0.004731 |0.01345 |0.01652 |0.005905 |0.01619 |0.002081 |12.83 |15.54 |82.14 |495.2 |0.09616 |0.07866 |0.1147 |0.05104 |0.2309 |0.06915 |[12.18,13.12,77.79,451.1,0.06955,0.04234,0.03383,0.02292,0.1337,0.05581,0.3661,0.469,2.41,24.44,0.004731,0.01345,0.01652,0.005905,0.01619,0.002081,12.83,15.54,82.14,495.2,0.09616,0.07866,0.1147,0.05104,0.2309,0.06915]|[20.0,0.0] |[1.0,0.0] |0.0 |

# |ANUSREE-M |16.13 |19.1 |108.1 |798.8 |0.08992 |0.2022 |0.1722 |0.1028 |0.1879 |0.05852 |0.5692 |0.948 |3.854 |54.18 |0.005332 |0.02115 |0.01536 |0.01187 |0.01522 |0.002815 |20.96 |26.19 |136.8 |1315.0 |0.1313 |0.4233 |0.4784 |0.2073 |0.2962 |0.08472 |[16.13,19.1,108.1,798.8,0.08992,0.2022,0.1722,0.1028,0.1879,0.05852,0.5692,0.948,3.854,54.18,0.005332,0.02115,0.01536,0.01187,0.01522,0.002815,20.96,26.19,136.8,1315.0,0.1313,0.4233,0.4784,0.2073,0.2962,0.08472] |[0.0,20.0] |[0.0,1.0] |1.0 |

# |AKHILA-B |9.504 |18.66 |60.34 |273.9 |0.1158 |0.06492 |0.02956 |0.02076 |0.2128 |0.06767 |0.2773 |0.8937 |1.909 |15.7 |0.006532 |0.02347 |0.02645 |0.01243 |0.01743 |0.003641 |10.23 |27.95 |65.13 |314.9 |0.1786 |0.1148 |0.08867 |0.06227 |0.39 |0.1179 |[9.504,18.66,60.34,273.9,0.1158,0.06492,0.02956,0.02076,0.2128,0.06767,0.2773,0.8937,1.909,15.7,0.006532,0.02347,0.02645,0.01243,0.01743,0.003641,10.23,27.95,65.13,314.9,0.1786,0.1148,0.08867,0.06227,0.39,0.1179] |[20.0,0.0] |[1.0,0.0] |0.0 |

# |ATHIRA-M |14.25 |18.66 |96.42 |645.7 |0.1158 |0.1436 |0.09847 |0.06152 |0.2128 |0.06777 |0.7036 |0.8937 |5.373 |60.78 |0.006532 |0.02336 |0.02905 |0.01215 |0.01743 |0.003643 |17.67 |27.95 |119.1 |959.5 |0.1786 |0.3724 |0.3664 |0.1491 |0.39 |0.1179 |[14.25,18.66,96.42,645.7,0.1158,0.1436,0.09847,0.06152,0.2128,0.06777,0.7036,0.8937,5.373,60.78,0.006532,0.02336,0.02905,0.01215,0.01743,0.003643,17.67,27.95,119.1,959.5,0.1786,0.3724,0.3664,0.1491,0.39,0.1179] |[0.0,20.0] |[0.0,1.0] |1.0 |

# +------------+-----------+------------+--------------+---------+---------------+----------------+--------------+-------------------+-------------+----------------------+---------+----------+------------+-------+-------------+--------------+------------+-----------------+-----------+--------------------+------------+-------------+---------------+----------+----------------+-----------------+---------------+--------------------+--------------+-----------------------+-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+-------------+-----------+----------+

# Extract selected columns from bulk\_input\_prediction

selected\_df = bulk\_input\_prediction.select("Patient\_Name","prediction")

# Cross Verify

# print("Display Selected columns from selected\_df...")

# selected\_df.show(5, False)

# Display Selected columns from selected\_df...

# +------------+----------+

# |Patient\_Name|prediction|

# +------------+----------+

# |SHEENA-M |1.0 | Tumour Found

# |ROSAMMA-B |0.0 | No Tumour

# |ANUSREE-M |1.0 | Tumour Found

# |AKHILA-B |0.0 | No Tumour

# |ATHIRA-M |1.0 | Tumour Found

# +------------+----------+

# Add the "Final\_Output" column based on the condition

selected\_df = selected\_df.withColumn("Final\_Output",

when(selected\_df["prediction"] == 1, "Tumour Detected")

.otherwise("No Tumour Detected"))

selected\_df\_final = selected\_df.select("Patient\_Name","Final\_Output")

########## Display prediction results in color frame

# Convert final\_prediction to Pandas DataFrame

pandas\_df = selected\_df\_final.toPandas()

# # Display the top 5 rows

# print(pandas\_df.head())

# Apply color to the DataFrame output

# The applymap method is used to apply a function to each element of the DataFrame

# the lambda function in this code snippet is used to assign the CSS style 'color: red'

# to the cells in the 'Price' column of the DataFrame when generating its styled representation.

# background colors, 'red','green','blue','yellow','orange','purple','pink'

styled\_df = pandas\_df.style.applymap(lambda x: 'background-color: aliceblue',

subset=["Patient\_Name","Final\_Output"])

# Generate an HTML table(html\_table) with customized CSS

# 'th' = Table Heading, 'td' = Table Data

html\_table = styled\_df.hide(axis="index").set\_table\_styles(

[{'selector': 'th', 'props':[('color', 'red'), ('padding', '5px')]},

{'selector': 'td', 'props': [('vertical-align', 'middle'),('padding', '5px')]}

]).to\_html()

# Draw a frame around your output and Display the contents

# the f prefix enables you to embed expressions within a string by enclosing them in curly braces {}

# Within the string, you have embedded an expression {html\_table}.

# The value of the html\_table variable is dynamically inserted into the string at that location, that

# values are nothing but your prediction results with headings.

output\_frame = f"""

<div style="border: 2px solid black; padding: 10px;width: 50%; background-color: lightgrey;">

<h3 style='color:blue; text-align:left;'>Final Prediction as per your Input</h3>

{html\_table}

</div>

"""

display(HTML(output\_frame))

############ Output of Bulk Input Prediction

# Final Prediction as per your Input

# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# Patient\_Name Final\_Output

#----------------------------

# SHEENA-M Tumour Detected

# ROSAMMA-B No Tumour Detected

# ANUSREE-M Tumour Detected

# AKHILA-B No Tumour Detected

# ATHIRA-M Tumour Detected

# TANUSHREE-B No Tumour Detected

# AMALA-B No Tumour Detected

# ANJANA-M Tumour Detected

# SAVITHRI-M Tumour Detected

# RENUKA-B No Tumour Detected

# SNEHA-B No Tumour Detected

# MEGHA-B No Tumour Detected

# HRIDAYA-M Tumour Detected

# NEETHU-M Tumour Detected

# KUNJAMMA-B No Tumour Detected

# ALEYAMMA-B No Tumour Detected

# PRINCY-B No Tumour Detected

# MARIYA-M Tumour Detected

# GINJU-M Tumour Detected

# JESSY-B No Tumour Detected

print()

display(HTML("<h3 style='color:orange; text-align:left;'>Bulk Input Data Prediction Finished !!!</h3>"))

main\_function()

def go\_back\_function(event):

clear\_result\_grid()

main\_function()

print()

def main\_function():

# clear\_result\_grid()

# Set focus to result grid

set\_focus\_cell2()

print()

print()

print()

print()

display(HTML("<h3 style='color:magenta; text-align:center;'>Main Menu - Breast Cancer Prediction</h3>"))

print()

# Create your buttons for each events

load\_data\_button = widgets.Button(description="1. Load Data")

eda\_analysis\_button = widgets.Button(description="2. EDA Analysis")

pre\_process\_button = widgets.Button(description="3. Pre Process Data")

transform\_data\_button = widgets.Button(description="4. Transform Data")

build\_model\_and\_save\_button = widgets.Button(description="5. Build Model & Save")

single\_mail\_predict\_button = widgets.Button(description="6. Prediction(Single Input)")

bulk\_mail\_predict\_button = widgets.Button(description="7. Prediction(Bulk Input)")

# Assign python functions as for each Menu events

load\_data\_button.on\_click(load\_data)

eda\_analysis\_button.on\_click(EDA\_start)

pre\_process\_button.on\_click(pre\_process\_data)

transform\_data\_button.on\_click(transformation)

build\_model\_and\_save\_button.on\_click(build\_model\_save)

single\_mail\_predict\_button.on\_click(single\_input\_and\_predict)

bulk\_mail\_predict\_button.on\_click(bulk\_data\_load)

# Define custom CSS to adjust button dimensions, backcolor and textcolor

# Here is a list of some popular CSS colors:

# red,green,blue,yellow,black,white,gray,purple,orange,pink,brown,cyan,magenta

# You can also use more specific colors using HEXADECIMAL coded, such as:

# For more hexa color codes go to " https://www.color-hex.com/"

# #FF0000 (bright red), #00FF00 (bright green), #0000FF (bright blue), #FFFFFF (pure white)

# #000000 (pure black). #808080 (gray)

########################

# For more details regarding Background color and foreground text go to

# https://www.uxmatters.com/mt/archives/2007/01/applying-color-theory-to-digital-displays.php

########################

custom\_css = """

<style>

.custom-button {

width: 200px; /\* Change this value for width \*/

height: 50px; /\* Change this value for height \*/

color: blue; /\* Text color \*/

background-color: lightgreen; /\* Background color \*/

border: 2px solid black;

}

/\* Add a mouse hover effect \*/

.custom-button:hover {

background-color: white; /\* Change background color on mouse hover \*/

}

</style>

"""

# Apply CSS class properties to buttons

load\_data\_button.add\_class("custom-button")

eda\_analysis\_button.add\_class("custom-button")

pre\_process\_button.add\_class("custom-button")

transform\_data\_button.add\_class("custom-button")

build\_model\_and\_save\_button.add\_class("custom-button")

single\_mail\_predict\_button.add\_class("custom-button")

bulk\_mail\_predict\_button.add\_class("custom-button")

# Display the custom CSS and buttons

# ensuring that the CSS rules are applied to the button's appearance when

# they are displayed in the notebook.

display(HTML(custom\_css))

# Arrange buttons horizontally, HBox is used to display buttons horizontally

# VBox is used to display buttons vertically

buttons\_horizontal\_layout\_1 = widgets.HBox([load\_data\_button,

eda\_analysis\_button, pre\_process\_button])

buttons\_horizontal\_layout\_2 = widgets.HBox([transform\_data\_button,

build\_model\_and\_save\_button,

single\_mail\_predict\_button])

buttons\_horizontal\_layout\_3 = widgets.HBox([bulk\_mail\_predict\_button])

# Center the buttons using a centered container in result grid

centered\_layout = widgets.Layout(justify\_content='center')

centered\_container\_2 = widgets.Box([buttons\_horizontal\_layout\_2], layout=centered\_layout)

# Arrange buttons horizontally, in center of the output area

centered\_container\_1 = widgets.Box([buttons\_horizontal\_layout\_1], layout=centered\_layout)

display(centered\_container\_1)

centered\_container\_3 = widgets.Box([buttons\_horizontal\_layout\_3], layout=centered\_layout)

# Display the centered buttons with one line gap between each row

print()

display(centered\_container\_2)

print()

display(centered\_container\_3)

print()

# Declare a main function as follows to invoke all other function

if \_\_name\_\_ == "\_\_main\_\_":

# Call main function

main\_function()

############################ End Of Main Program Listing #######################

**Login Code Program Listing**

##### login module for 'Breast\_Cancer\_Prediction.ipynb'

# As part of the deployment only authenticated or subscribed persons can

# access the Recommendation program. Two Step Verifications are required.

# How to subscribe

# 1. When a user subscribed to use ''Breast\_Cancer \_Prediction' program, the developer

# send a link to the subscriber to copy and paste the entire program

# 2. After copying the program, the developer sends this login program to the

# subscriber.

# 3. With a valid gmail id the user can log in to 'Breast\_Cancer \_Prediction model using his/her

# gmail-id and password.

# 4. The subscriber can use Password recovery and email-id recovery protocols laid down

# by Google

from IPython.display import display, HTML,Javascript

# Define a function to scroll your code cell directly to output grid, other wise you will

# have to do it manually

def set\_focus\_cell1():

display(Javascript("""

document.getElementById('output-area').scrollIntoView({behavior: 'smooth'});

"""))

# Call set\_focus\_cell1 to scroll down automatically this cell

set\_focus\_cell1()

# Import all Google Gmail packages

from google.colab import drive

from google.colab import output

import os

from google.colab import auth

from IPython.display import HTML

# Transfer control to gmail login protocols

auth.authenticate\_user()

display(HTML("<h3 style='color:red'>Logging Started Please Wait, may take 1 to 2 Minutes !!!</h3>"))

# Connect to **Breast Cancer**\_Prediction Model

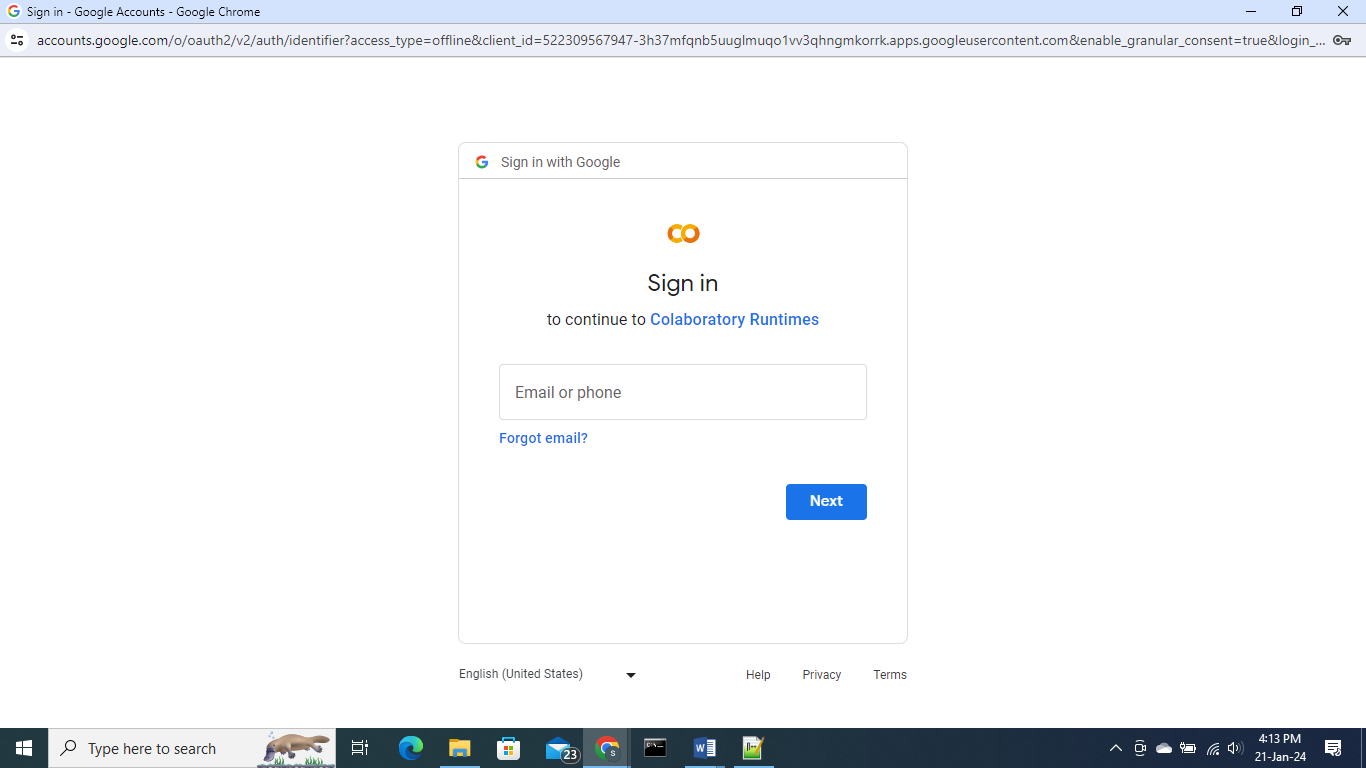
drive.mount('/content/drive')

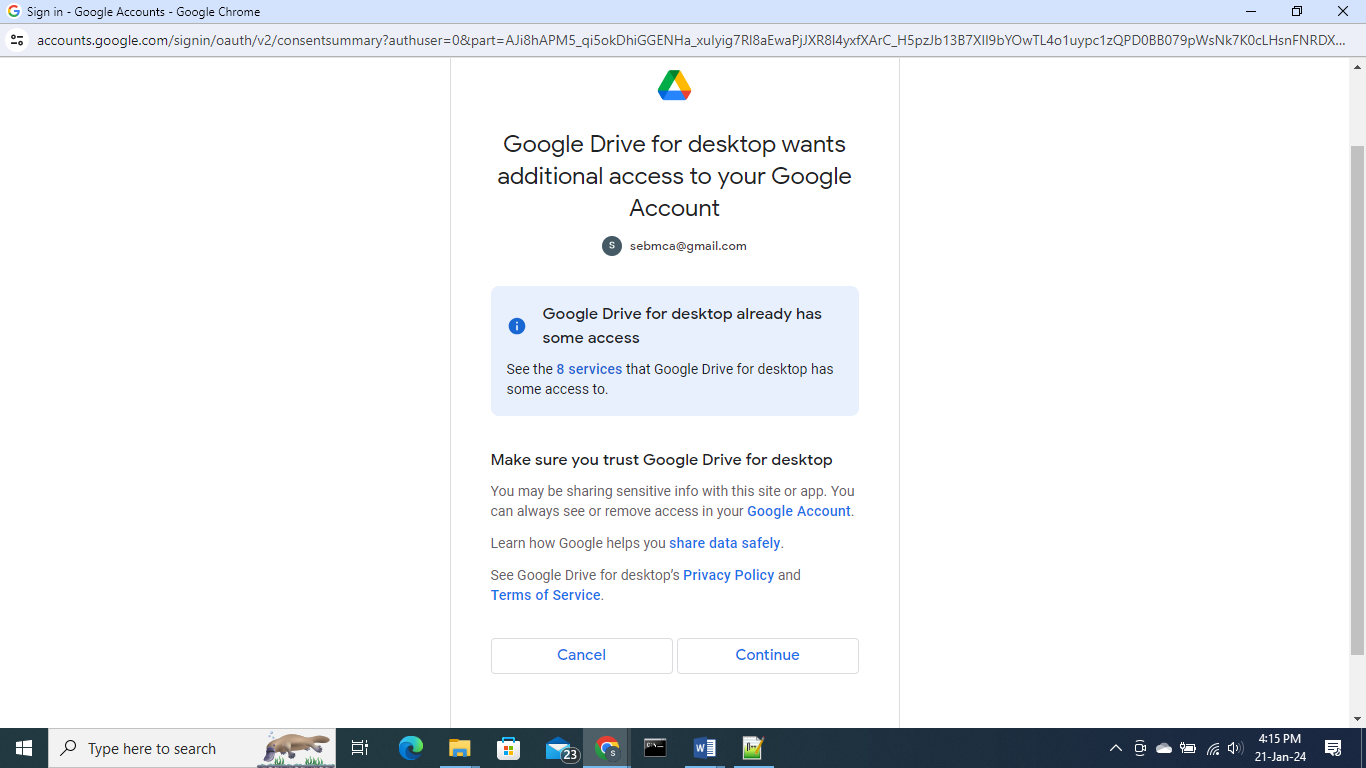
display(HTML("<h3 style='color:green'>Connected ... wait !!!</h3>"))

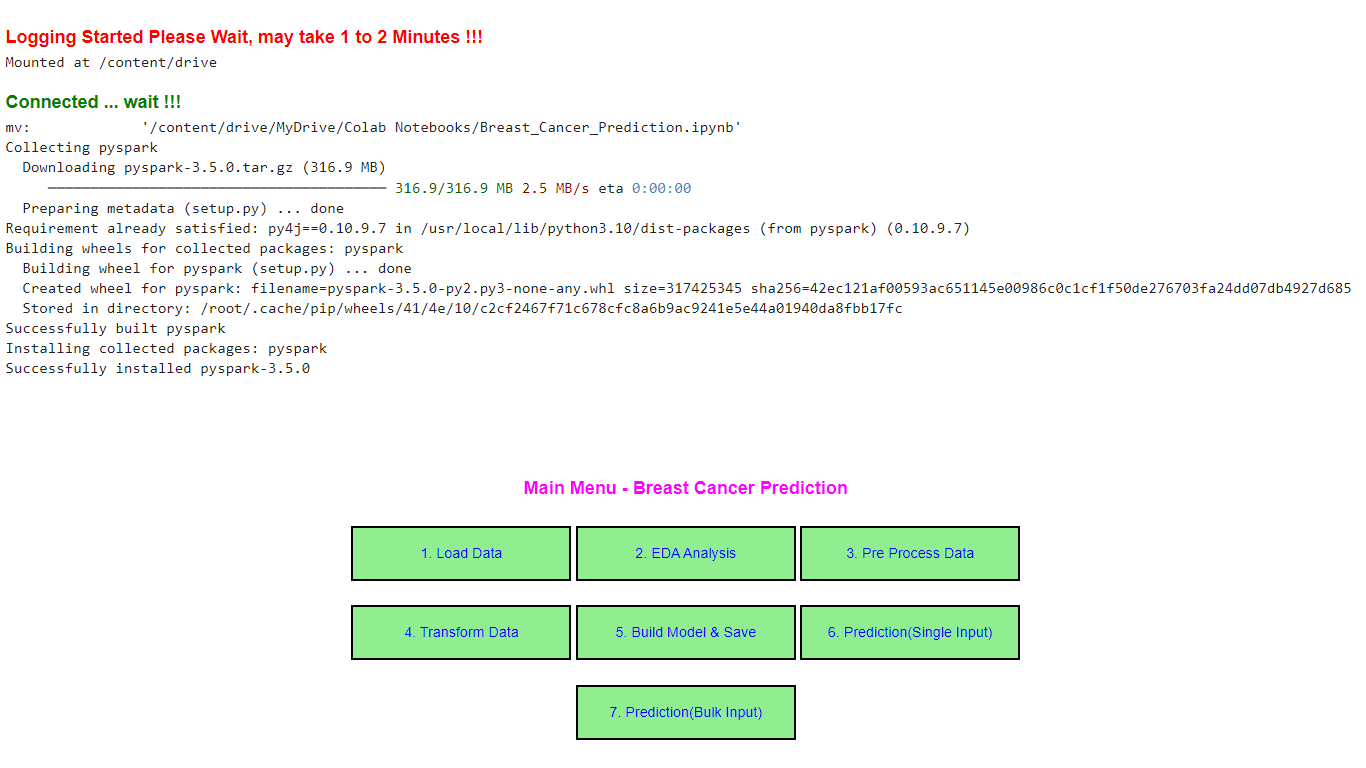
%run '/content/drive/MyDrive/'breast\_bancer \_Prediction.ipynb'

############# End of Login Protocols ####################

**Output Generated by Login Program – 2 Step Verification**

****

****

****

**Additional Features**

1. The commands used in this program are compatible with operating systems like **Widnows, Linux/Unix,**

**Android and iOS.**

2. Using mobile phone open your Gmail Account

3. Open Google Colab and Click the **connect** option on top right hand side of Colab

4. Select the ‘login’ program, then run the code cell on the left panel of Colab

5. After One or Two minutes you will get the Button driven Menu of the Software