BREAST CANCER PREDICTION USING PYTHON WITH MACHINE LEARNING

An internship report submitted in partial fulfillment of the requirements for the

Award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

ACHANTA SATYANARAYANA VINAY -1215316501

Under the esteemed guidance of

V.S.L.Prasad

Python developer in Grepthor Software Solutions Pvt.Ltd

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

GITAM

(Deemed to be University)

VISHAKAPATNAM

MAY-JUNE 2019

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GITAM INSTITUTE OF TECHNOLOGY

GITAM

(Deemed to be University)

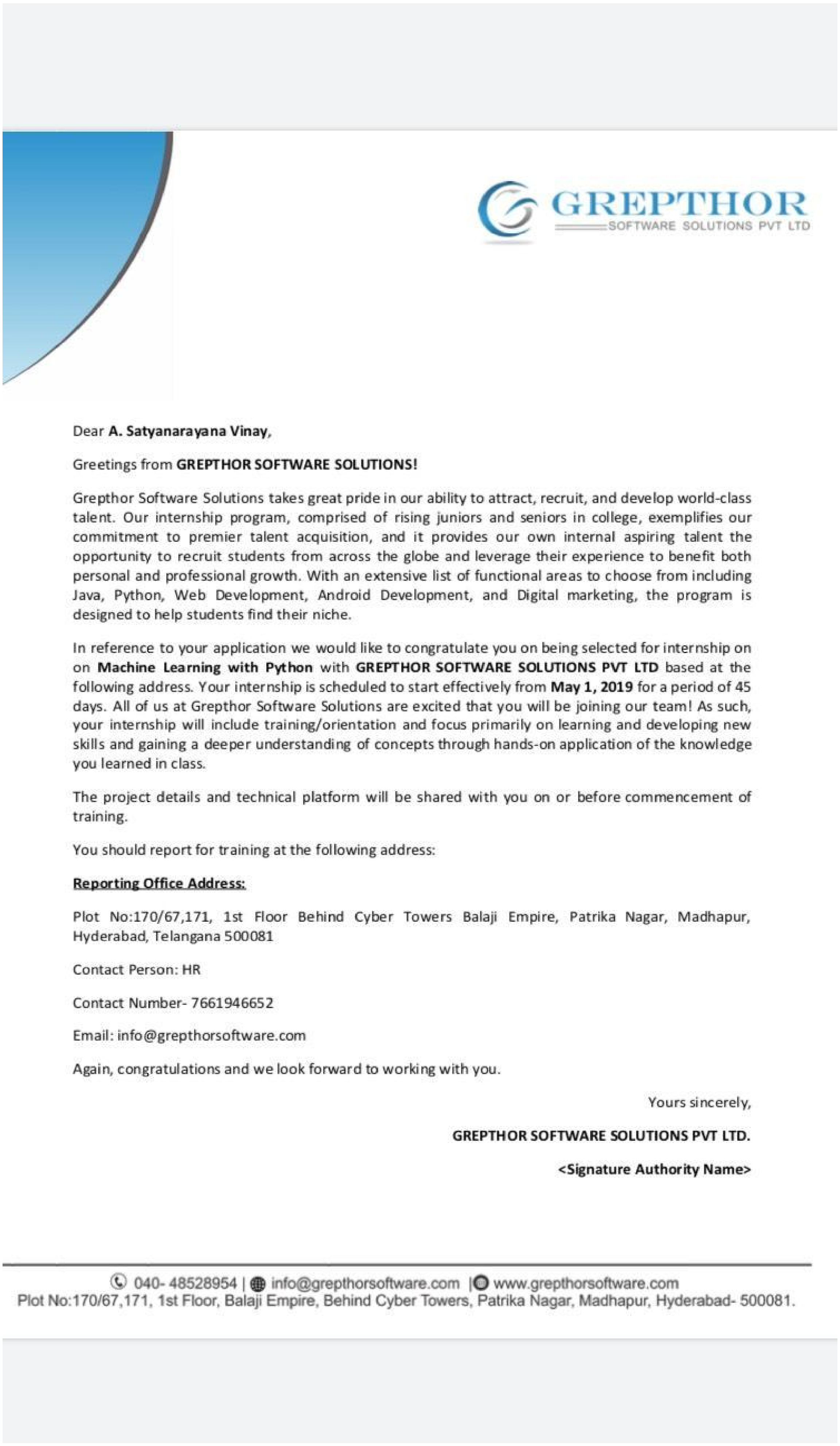
DECLARATION

We, hereby declare that the internship review entitled "**BREAST CANCER PREDICTION USING PYTHON WITH MACHINE LEARNING**". is an original work done in the Department of Computer Science and Engineering, GITAM Institute of Technology, GITAM (Deemed to be University) submitted in partial fulfillment of the requirements for the award of the degree of B.Tech. in Computer Science and Engineering.

The work has not been submitted to any other college or University for the award of any degree or diploma.

Date:

1215316501 **A.S.VINAY**



GREPTHOR
SOFTWARE SOLUTIONS PVT LTD

Dear **A. Satyanarayana Vinay**,

Greetings from **GREPTHOR SOFTWARE SOLUTIONS!**

Grepthor Software Solutions takes great pride in our ability to attract, recruit, and develop world-class talent. Our internship program, comprised of rising juniors and seniors in college, exemplifies our commitment to premier talent acquisition, and it provides our own internal aspiring talent the opportunity to recruit students from across the globe and leverage their experience to benefit both personal and professional growth. With an extensive list of functional areas to choose from including Java, Python, Web Development, Android Development, and Digital marketing, the program is designed to help students find their niche.

In reference to your application we would like to congratulate you on being selected for internship on on **Machine Learning with Python** with **GREPTHOR SOFTWARE SOLUTIONS PVT LTD** based at the following address. Your internship is scheduled to start effectively from **May 1, 2019** for a period of 45 days. All of us at Grepthor Software Solutions are excited that you will be joining our team! As such, your internship will include training/orientation and focus primarily on learning and developing new skills and gaining a deeper understanding of concepts through hands-on application of the knowledge you learned in class.

The project details and technical platform will be shared with you on or before commencement of training.

You should report for training at the following address:

Reporting Office Address:

Plot No:170/67,171, 1st Floor Behind Cyber Towers Balaji Empire, Patrika Nagar, Madhapur, Hyderabad, Telangana 500081

Contact Person: HR

Contact Number- 7661946652

Email: info@grepthorsoftware.com

Again, congratulations and we look forward to working with you.

Yours sincerely,

GREPTHOR SOFTWARE SOLUTIONS PVT LTD.

<Signature Authority Name>

040- 48528954 | info@grepthorsoftware.com | www.grepthorsoftware.com
Plot No:170/67,171, 1st Floor, Balaji Empire, Behind Cyber Towers, Patrika Nagar, Madhapur, Hyderabad- 500081.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GITAM INSTITUTE OF TECHNOLOGY

GITAM

(Deemed to be University)

CERTIFICATE

This is to certify that the internship report entitled " **BREAST CANCER PREDICTION USING PYTHON WITH MACHINE LEARNING**" is a bonafide record of work carried out by **A.S.VINAY(1215316501)**students submitted in partial fulfillment of requirement for the award of degree of Bachelors of Technology in Computer Science and Engineering.

SUPERVISOR	**INTERNSHIP REVIEWER**
Prasad	**Asst Prof. P.Radhika**

ACKNOWLEDGEMENTS:

The internship opportunity I had with Grepthor Software Solutions Pvt.Ltd was a great chance for learning and professional development. Therefore, I consider myself a very lucky individual as I was provided with an opportunity to be a part of it. Bearing in mind previous I am using this opportunity to express my deepest gratitude and special thanks to Prasad at Grepthor Software Solutions Pvt.Ltd who in spite of being extraordinarily busy with his duties, took time out to hear, guide and keep me on the correct path and allowing me to carry out my work at their esteemed organization.

I express my deepest thanks to Mr. Prasad for taking part in useful decisions & giving necessary advice and guidance, as mentors. I choose this moment to acknowledge their contribution gratefully.

I express my deepest thanks to Dr. Konala Thammi Reddy sir, HOD of C.S.E department, GITAM Institute Technology for giving me a great opportunity to complete my internship in the company. I choose this moment to acknowledge his contribution to gratefully.

I would also like to thank Asst Prof. P.Radhika mam, A.M.C who helped us a lot in the successful completion of our internship and internship report.

I perceive as this opportunity as a big milestone in my career development. I will strive to use gained skills and knowledge in the best possible way, and I will continue to work on their improvement, in order to attain desired career objectives. Hope to continue cooperation with all of you in the future.

Sincerely,

A.S.VINAY

TABLE OF CONTENTS

1. ABSTRACT:

Cancer is the second cause of death in the world. 8.8 million patients died due to cancer in 2015. Breast cancer is the leading cause of death among women. Several types of research have been done on early detection of breast cancer to start treatment and increase the chance of survival. Most of the studies concentrated on mammogram images. However, mammogram images sometimes have a risk of false detection that may endanger the patient's health. It is vital to find alternative methods which are easier to implement and work with different data sets, cheaper and safer, that can produce a more reliable prediction.

This paper proposes a hybrid model combined of several Machine Learning (ML) algorithms including Support Vector Machine (SVM), Artificial Neural Network (ANN), K-Nearest Neighbor (KNN), Decision Tree (DT) for effective breast cancer detection. This study also discusses the datasets used for breast cancer detection and diagnosis. The proposed model can be used with different data types such as image, blood, etc. Index Terms—Breast Cancer; Breast Cancer Detection; Medical Images; Machine Learning.

2. ABOUT ORGANIZATION:

Grepthor software solutions was incorporated early 2013 with a key initiative of developing business solutions. The company's main objective had been to be a one stop solution provider for all IT needs and IT enables services. Grepthor as a company is built on the core believes of business ethics, strong process, uncompromising attitude towards quality delivered. With the help of strong individuals, core professionals and highly qualified man power to achieve objectives of growth, keeping up the pace with the trends and technological upgradations on par with the industry.

Grepthor software solutions is a leading web and mobile applications development company. We are a team of 200+ who are dedicated to serve the customers and have been functionalize in the IT industry since 2013.

3. SCHEDULE OF INTERNSHIP AND TRAINING:

The project assigned to me is based on Machine Learning Using Python. The joining date for the internship is 2/05/2019 and is scheduled to 15/06/2019.

Week 1:

- Introduction to Python Scripting
- Platforms Used in Python Scripting
- Python Software Installation
- Python Applications in IDLE
- Python Datatypes
- String Handling
- Python Modules
- Exception Handling in Python Scripting
- Functions In Python Scripting

Week 2:

- Oops Application Configurations in Python Scripting
- Constructors
- Python Scripting With Database Configurations
- Python Variables and Methods in Class and Objects
- Working with Dir function and help Function
- Introduction Python with Machine learning

Implementations

Week 3:

- Anaconda Jupiter Software Installation
- Working With DataSets

Week 4:

- Working With Pandas Module
- Pandas Properties
- Workking With Numpy Module
- Working With Matplotlib Module

Week 5:

- Navibayes algorithm
- Decision Tree Techniques

Week 6:

- logistic regression
- linear regression

4. INTERNSHIP ACTIVITES

4.2. TRAINING:

INTRODUCTION:

Breast cancer (BC) is one of the most common cancers among women worldwide, representing the majority of new cancer cases and cancer-related deaths according to global statistics, making it a significant public health problem in today's society.The early diagnosis of BC can improve the prognosis and chance of survival significantly, as it can promote timely clinical treatment to patients. Further accurate classification of benign tumors can prevent patients undergoing unnecessary treatments. Thus, the correct diagnosis of BC and classification of patients into malignant or benign groups is the subject of much research. Because of its unique advantages in critical features detection from complex BC datasets, machine learning (ML) is widely recognized as the methodology of choice in BC pattern classification and forecast modelling.

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

There are some guidelines:

Mammography. The most important screening test for breast cancer is the mammogram. A mammogram is an X-ray of the breast. It can detect breast cancer up to two years before the tumor can be felt by you or your doctor. **Women age 40–45 or older** who are at average risk of breast cancer should have a mammogram once a year.**Women at high risk** should have yearly mammograms along with an MRI starting at age 30.

PROBLEM STATEMENT:

the chances of an individual of getting a disease, such as breast cancer. Presence of risk factors isn't a complete affirmative that a woman will develop breast cancer. There have been cases where many
women with breast cancer have no apparent risk factors. Some factors need to be known by women so that they canower their risk of breast cancer. Since causes of breast cancer are not fully known. Researchers believe that these risk factors increase (or decrease) the changes of developing breast cancer. Since breast cancer is a complex disease it is likely to be caused by a combination of risk factors. Some of the factors associated with breast cancer – can't be changed (Non-preventable) like age, genetic factor, heredity. While making choices can change other factors (Preventable) like overweight, lack of exercises, smoking,smoking,harmone replacement therapy.

REVIEW OF LITERATURE:

A literature review showed that there have been several studies on the survival prediction problem using statistical approaches and artificial neural networks. However, we could only find a few studies related to medical diagnosis and recurrence using data mining approaches such as decision trees . Delen et al. used artificial neural networks, decision trees and logistic regression to develop prediction models for breast cancer survival by analyzing a large dataset, the SEER cancer incidence database . Lundin et al. used ANN and logistic regression models to predict 5, 10, and 15 -year breast cancer survival. They studied 951 breast cancer patients and used tumor size, axillary nodal status, histological type, mitotic count, nuclear pleomorphism, tubule formation, tumor necrosis, and age as input variables . Pendharker et al. used several data mining techniques for exploring interesting patterns in breast cancer. In this study, they showed that data mining could be a valuable tool in identifying similarities (patterns) in breast cancer cases, which can be used for diagnosis, prognosis, and treatment purposes . These studies are some examples of researches that apply data mining to medical fields for prediction of diseases.

DATA COLLECTION

Data collection is the process of gathering and measuring information from the patient in the organization or from any other organization, in an established systematic fashion that enables one to answer stated research questions, test hypotheses, and evaluate outcomes. The data collection component of research is common to all fields of study including physical and social sciences, humanities, business, etc. While methods vary by discipline, the emphasis on ensuring accurate and honest collection remains the same. The data collection for cancer patients can be determined by considering the id type.

4.2 ROLE IN THE INTERNSHIP:

As a Machine Learning Trainee-cum-Intern, I was initially trained by our guide Prasad, in different concepts of Python for the first 2 weeks using the 'IDLE' or python version platform. In the third week, I was given access to Anaconda Jupiter installation , to deploy the programs and for predicting data. In the same week, all the interns were divided into teams of three. We were assigned with the project - "Breast Cancer Prediction Using Machine Learning".

I was an active member of the group and participated in all the phases of the project, from research to deployment. All of us were instrumental in the completion of the project with an equal contribution at every stage . On the last day of my internship, our group had done a presentation with respect to our project. Our presentation was met with immense applause which concluded the sixth-week journey on a very happy note.

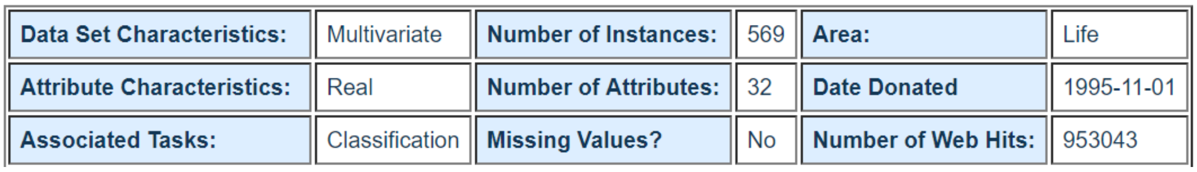
4.3. METHODOLOGIES AND FUNCTIONALTIES:

Methodology:

In order to predict the 2-year recurrence rate of breast cancer, we used ICBC dataset in the National Cancer Institute of Tehran for the years 1997-2008. The ICBC is responsible for collecting incidence and survival data from the participating registries, and disseminating these datasets for the purpose of conducting analytical research projects. This dataset contained population characteristics and included 22 input variables. Our cases were collected from the total number of 1189 women that were diagnosed breast cancer. We preprocessed the data to remove unsuitable cases. After using data cleansing and data preparation strategies, the final dataset was constructed. Finally, 547 cases were analyzed after 642 records were excluded because of missing data. Patients with breast cancer recurrence were followed-up years. The independent variables that we used are shown in below. The dataset was cleaned by handling missing values, noise, identifying and correcting inconsistencies. Some fields, such as Her2, age of menarche, and Npositive, contained missing values. Since these variables are important in predicting recurrence, the records containing the missing data were removed from the dataset. Missing values for continuous variables were substituted using EM method.

Phase 0—Data Preparation:

We will use the UCI Machine Learning Repository for breast cancer dataset**.**



Data Set Characteristics:	Multivariate	**Number of Instances:**	569	**Area:**	Life
Attribute Characteristics:	Real	**Number of Attributes:**	32	**Date Donated**	1995-11-01
Associated Tasks:	Classification	**Missing Values?**	No	**Number of Web Hits:**	953043

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

Attribute Information:

1. ID number 2) Diagnosis (M = malignant, B = benign) 3–32)

Ten real-valued features are computed for each cell nucleus:

1. radius (mean of distances from center to points on the perimeter)
2. texture (standard deviation of gray-scale values)
3. perimeter
4. area
5. smoothness (local variation in radius lengths)
6. compactness (perimeter^2 / area—1.0)
7. concavity (severity of concave portions of the contour)
8. concave points (number of concave portions of the contour)
9. symmetry
10. fractal dimension (“coastline approximation”—1)

The mean, standard error and “worst” or largest (mean of the three largest values) of these features were computed for each image, resulting in 30 features. For instance, field 3 is Mean Radius, field 13 is Radius SE, field 23 is Worst Radius.

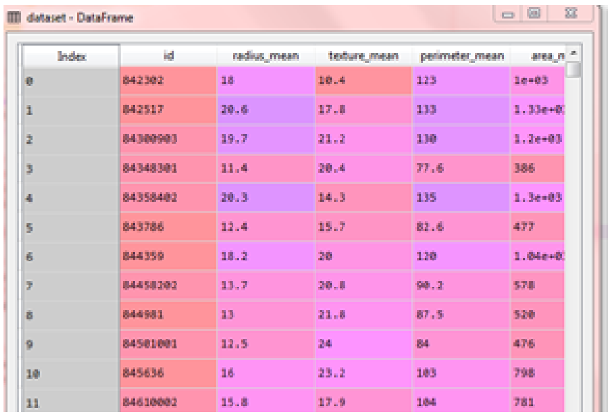
Objectives

This analysis aims to observe which features are most helpful in predicting malignant or benign cancer and to see general trends that may aid us in model selection and hyper parameter selection. The goal is to classify whether the breast cancer is benign or malignant. To achieve this i have used machine learning classification methods to fit a function that can predict the discrete class of new input**.**

Phase 1—Data Exploration:

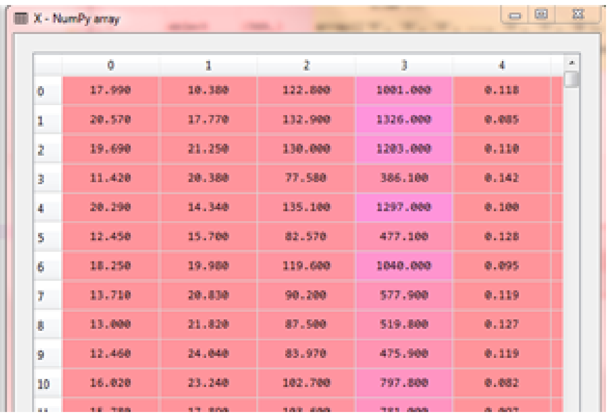
We will be using ***Spyder*** to work on this dataset. We will first go with importing the necessary libraries and import our dataset to Spyder :

```
#importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
#importing our cancer dataset
dataset = pd.read_csv(‘cancer.csv')
X = dataset.iloc[:, 1:31].values
Y = dataset.iloc[:, 31].values
```



dataset - DataFrame

Index	id	radius_mean	texture_mean	perimeter_mean	area_n
0	842302	18	10.4	123	1e+03
1	842517	20.6	17.8	133	1.33e+0
2	84300903	19.7	21.2	130	1.2e+03
3	84348301	11.4	20.4	77.6	386
4	84358402	20.3	14.3	135	1.3e+03
5	843786	12.4	15.7	82.6	477
6	844359	18.2	20	120	1.04e+0
7	84458202	13.7	20.8	90.2	578
8	844981	13	21.8	87.5	520
9	84501001	12.5	24	84	476
10	845636	16	23.2	103	798
11	84610002	15.8	17.9	104	781



X - NumPy array

	0	1	2	3	4
0	17.990	10.380	122.800	1001.000	0.118
1	20.570	17.770	132.900	1326.000	0.085
2	19.690	21.250	130.000	1203.000	0.110
3	11.420	20.380	77.580	386.100	0.142
4	20.290	14.340	135.100	1297.000	0.100
5	12.450	15.700	82.570	477.100	0.128
6	18.250	19.980	119.600	1040.000	0.095
7	13.710	20.830	90.200	577.900	0.119
8	13.000	21.820	87.500	519.800	0.127
9	12.460	24.040	83.970	475.900	0.119
10	16.020	23.240	102.700	797.800	0.082

We can examine the data set using the pandas' **head()** method.

dataset.head()

```
         id  radius_mean   ...     fractal_dimension_worst  diagnosis
0    842302        17.99   ...                     0.11890          M
1    842517        20.57   ...                     0.08902          M
2  84300903        19.69   ...                     0.08758          M
3  84348301        11.42   ...                     0.17300          M
4  84358402        20.29   ...                     0.07678          M
```

We can find the dimensions of the data set using the panda dataset 'shape' attribute.

print("Cancer data set dimensions : {}".format(dataset.shape))

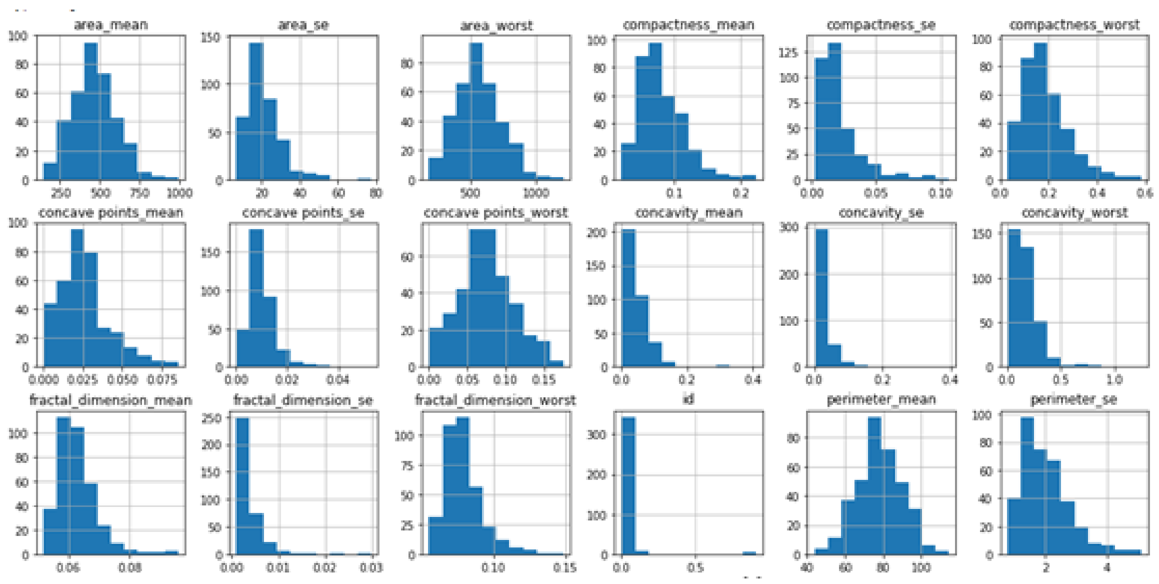
Cancer data set dimensions : (569, 32)

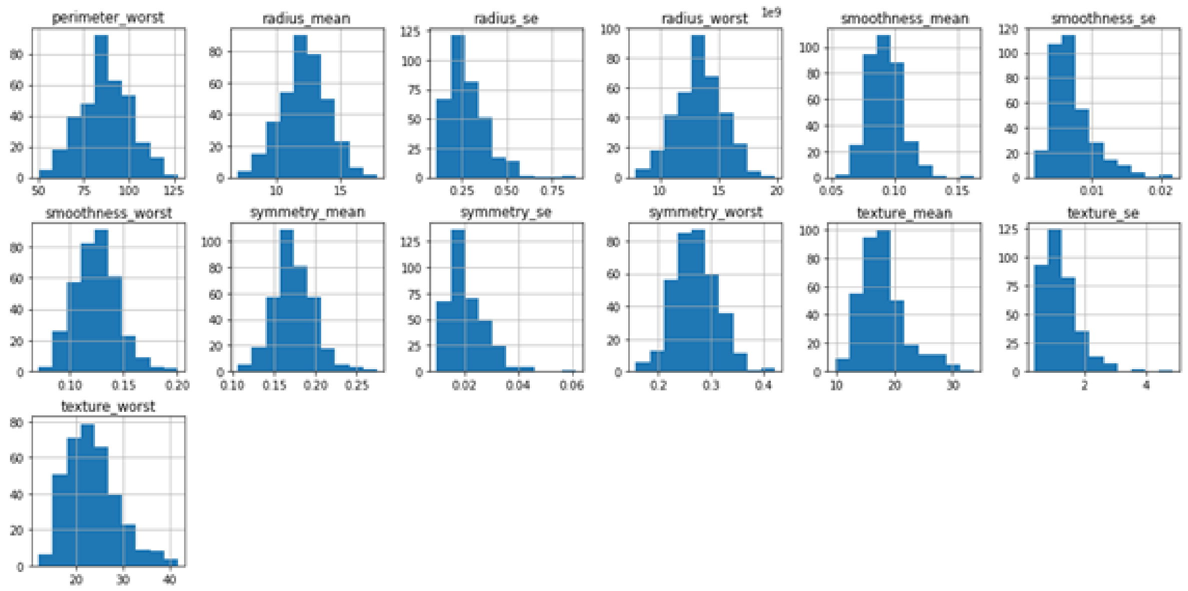
We can observe that the data set contain 569 rows and 32 columns. '*Diagnosis*' is the column which we are going to predict , which says if the cancer is M = malignant or B = benign. 1 means the cancer is malignant and 0 means benign. We can identify that out of the 569 persons, 357 are labeled as B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

```
diagnosis
B    357
M    212
dtype: int64
```

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

In this tutorial we will use pandas' visualization which is built on top of matplotlib, to find the data distribution of the features.


area_mean
area_se
area_worst
compactness_mean
compactness_se
compactness_worst
concave points_mean
concave points_se
concave points_worst
concavity_mean
concavity_se
concavity_worst
fractal_dimension_mean
fractal_dimension_se
fractal_dimension_worst
id
perimeter_mean
perimeter_se


1e9
perimeter_worst
radius_mean
radius_se
radius_worst
smoothness_mean
smoothness_se
smoothness_worst
symmetry_mean
symmetry_se
symmetry_worst
texture_mean
texture_se
texture_worst

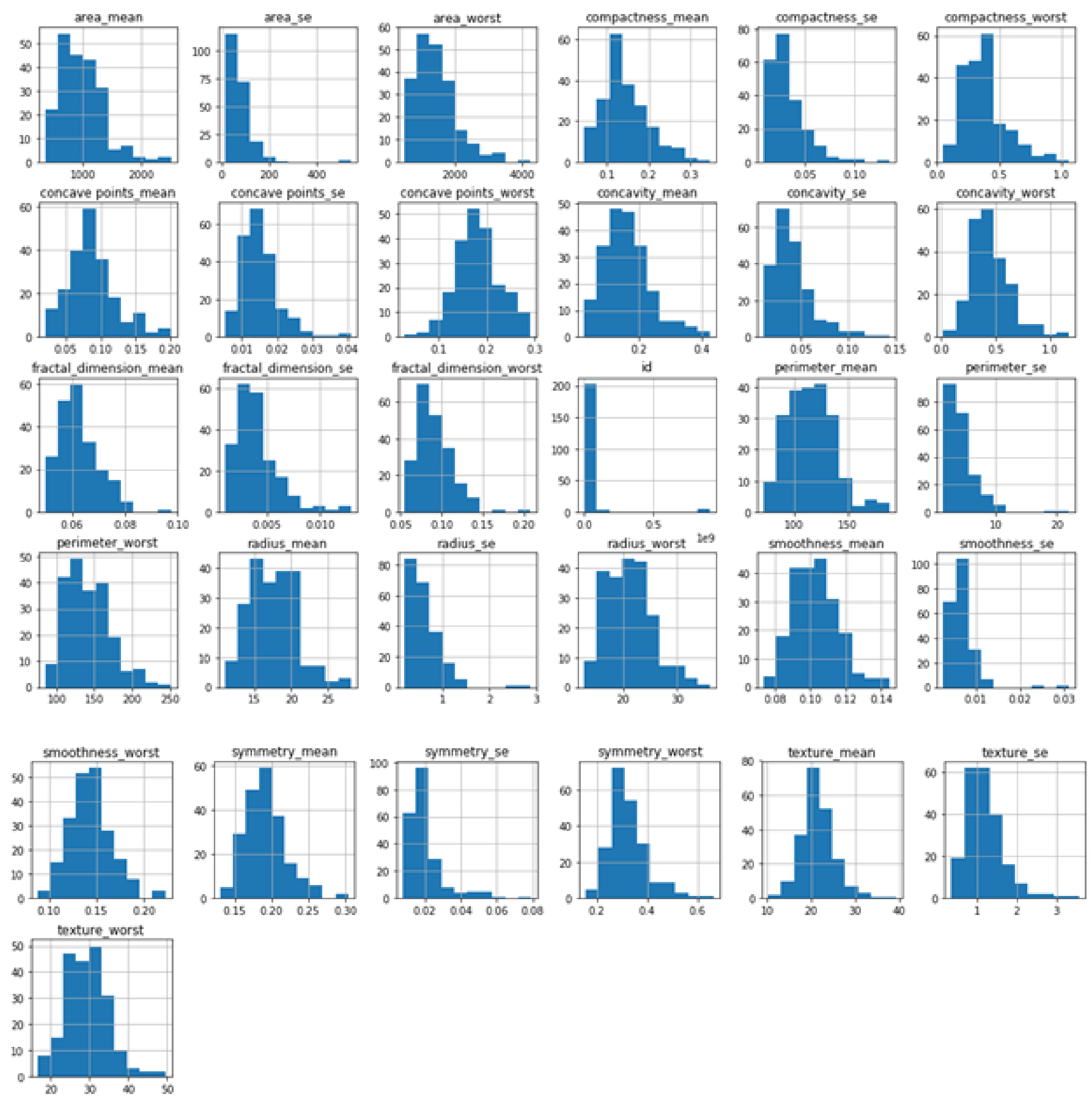
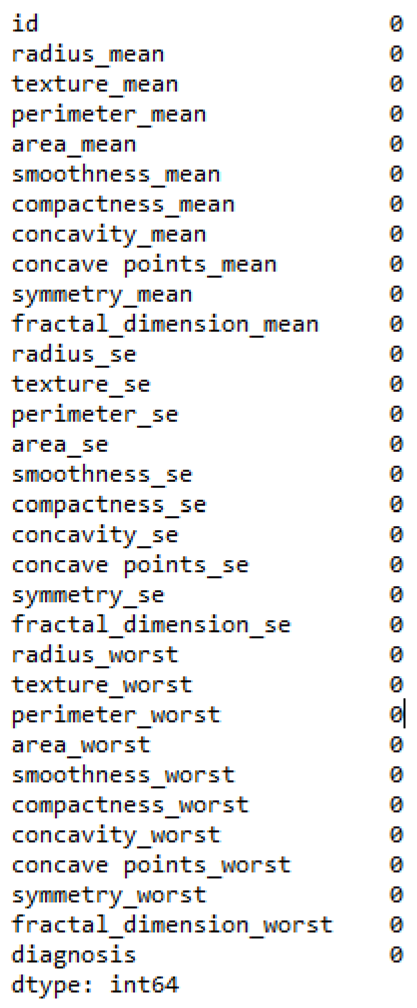


Fig : Visualization of Dataset

Missing or Null Data points:

We can find any missing or null data points of the data set (if there is any) using the following pandas function.

```
dataset.isnull().sum()
dataset.isna().sum()
```



```
id                         0
radius_mean                0
texture_mean               0
perimeter_mean             0
area_mean                  0
smoothness_mean            0
compactness_mean           0
concavity_mean             0
concave points_mean        0
symmetry_mean              0
fractal_dimension_mean     0
radius_se                  0
texture_se                 0
perimeter_se               0
area_se                    0
smoothness_se              0
compactness_se             0
concavity_se               0
concave points_se          0
symmetry_se                0
fractal_dimension_se       0
radius_worst               0
texture_worst              0
perimeter_worst            0
area_worst                 0
smoothness_worst           0
compactness_worst          0
concavity_worst            0
concave points_worst       0
symmetry_worst             0
fractal_dimension_worst    0
diagnosis                  0
dtype: int64
```

Fig : Observe missing data

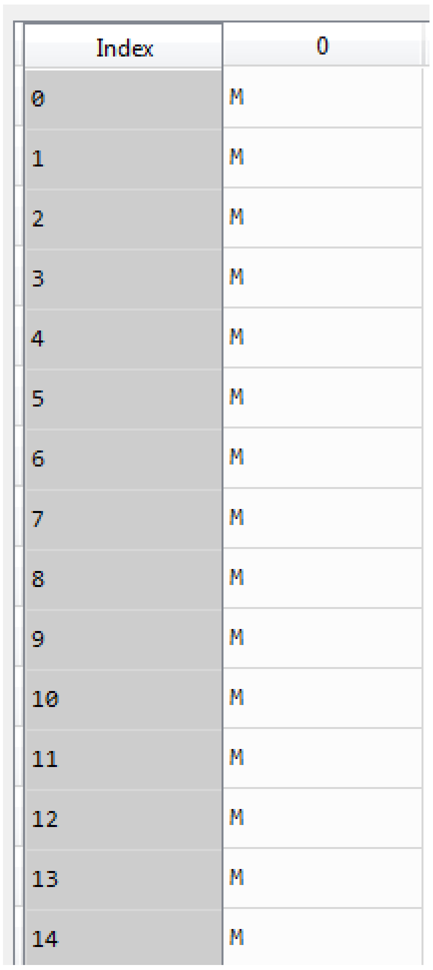
Phase 2—Categorical Data

Categorical data are variables that contain label values rather than numeric values.The number of possible values is often limited to a fixed set.

For example, users are typically described by country, gender, age group etc.

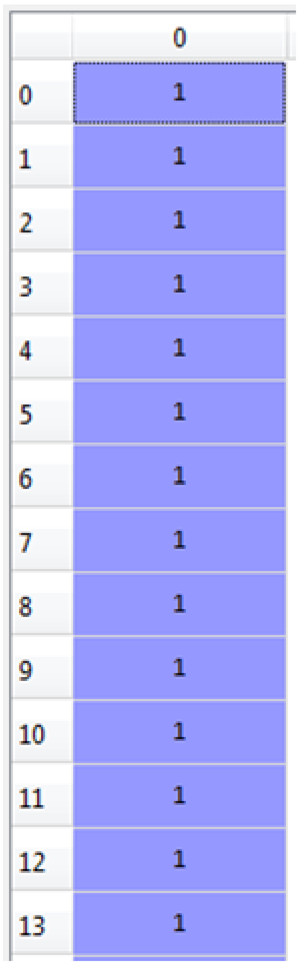
We will use Label Encoder to label the categorical data. Label Encoder is the part of SciKit Learn library in Python and used to convert categorical data, or text data, into numbers, which our predictive models can better understand.

```
#Encoding categorical data values
from sklearn.preprocessing import LabelEncoder
labelencoder_Y = LabelEncoder()
Y = labelencoder_Y.fit_transform(Y)
```



Index	0
0	M
1	M
2	M
3	M
4	M
5	M
6	M
7	M
8	M
9	M
10	M
11	M
12	M
13	M
14	M

Fig: Diagnosis Data without Encoding



	0
0	1
1	1
2	1
3	1
4	1
5	1
6	1
7	1
8	1
9	1
10	1
11	1
12	1
13	1

Fig: Diagnosis Data after Encoding

Splitting the dataset

The data we use is usually split into training data and test data. The training set contains a known output and the model learns on this data in order to be generalized to other data later on. We have the test dataset (or subset) in order to test our model's prediction on this subset.

We will do this using SciKit-Learn library in Python using the train_test_split method.

```
# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
test_size = 0.25, random_state = 0)
```

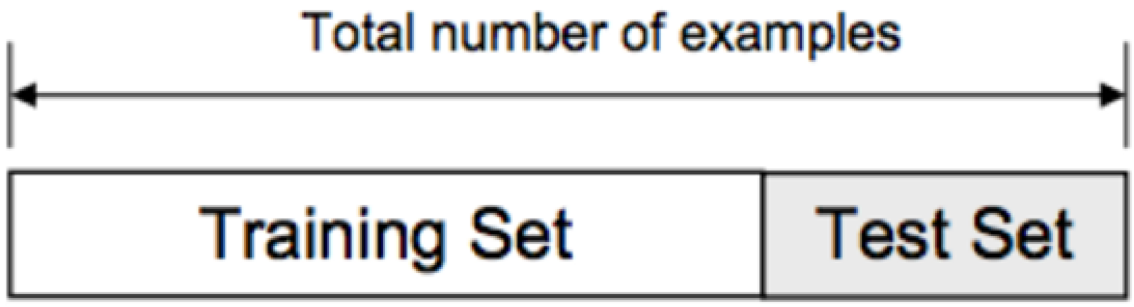


Fig: Training and test set

Phase 3—Feature Scaling

Most of the times, your dataset will contain features highly varying in magnitudes, units and range. But since, most of the machine learning algorithms use Eucledian distance between two data points in their computations. We need to bring all features to the same level of magnitudes. This can be achieved by scaling. This means that you're transforming your data so that it fits within a specific scale, like 0–100 or 0–1.

We will use StandardScaler method from SciKit-Learn library.

```
#Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Phase 4—Model Selection

This is the most exciting phase in Applying Machine Learning to any Dataset. It is also known as Algorithm selection for Predicting the best results.

Usually Data Scientists use different kinds of Machine Learning algorithms to the large data sets. But, at high level all those different algorithms can be classified in two groups : supervised learning and unsupervised learning.

Without wasting much time, I would just give a brief overview about these two types of learnings.

Supervised learning : Supervised learning is a type of system in which both input and desired output data are provided. Input and output data are labelled for classification to provide a learning basis for future data processing. Supervised learning problems can be further grouped into **Regression** and **Classification** problems.

A **regression** problem is when the output variable is a real or continuous value, such as “salary” or “weight”.

A **classification** problem is when the output variable is a category like filtering emails “spam” or “not spam”

Unsupervised Learning : Unsupervised learning is the algorithm using information that is neither classified nor labeled and allowing the algorithm to act on that information without guidance.

In our dataset we have the outcome variable or Dependent variable i.e Y having only two set of values, either M (Malign) or B(Benign). So we will use Classification algorithm of supervised learning.

We have different types of classification algorithms in Machine Learning :-

1. Logistic Regression

2. Nearest Neighbor

3. Support Vector Machines

4. Kernel SVM

5. Naïve Bayes

6. Decision Tree Algorithm

7. Random Forest Classification

Lets start applying the algorithms :

We will use sklearn library to import all the methods of classification algorithms.

1.Artificial Neural Network (ANN):

ANN is a model like human brains nerve system that has a large number of nodes connected to each other. Each node has two states: 0 means active and 1 means active. Also, each node has a positive or negative weight that adjusts the strength of the node and can activate or deactivate it. ANN provides samples of data to train the machine. The trained machine is used to detect the pattern of hidden date. It can search for patterns among patients' healthcare and personal records to identify high-risk lesions.

2.SUPPORT VECTOR MACHINE(SVM):

SVM is a supervised pattern classification model which is used as a training algorithm for learning classification and regression rule from gathered data .The purpose of this method is to separate data until a hyperplane with high minimum distance is found. SVM is used to classify two or more data types. SVM include single or hybrid models such as Standard SVM (St-SVM), Proximal Support Vector Machine (PSVM), Newton Support Vector Machine (NSVM), Lagrangian Support Vector Machines (LSVM),

Linear Programming Support Vector Machines (LPSVM), and Smooth Support Vector Machine (SSVM).

3.K-Nearest Neighbors (KNN) :

KNN is a supervised learning method which is used for diagnosing and classifying cancer . In this method, the computer is trained in a specific field and new data is given to it. Additionally, similar data is used by the machine for detecting (K) hence, the machine starts finding KNN for the unknown data. It is recommended to choose a large dataset for training also K value must be an odd number.

4.Random Forest (RF) Algorithm

RF algorithm is used at the regularization point where the model quality is highest, variance and bias problems are compromised . RF builds numerous numbers of DTs using random samples with a replacement to overcome the problem of DTs. Each tree classifies its observations, and majority votes decision is chosen. RF is used in the unsupervised mode for assessing proximities among the data points Bayes' theorem with robust independence assumptions [16]. In this model, all properties are considered separately todetect any existing relationship between them. It assumes that predictive attributes are conditionally independent given class. Moreover, the values of the numeric attributes arWe will use sklearn library to import all the methods of classification algorithms.

We will use LogisticRegression method of model selection to use Logistic Regression Algorithm,

```
#Using Logistic Regression Algorithm to the Training Set
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0)
classifier.fit(X_train, Y_train)
#Using KNeighborsClassifier Method of neighbors class to use Nearest Neighbor algorithm
```

from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)
classifier.fit(X_train, Y_train)

#Using SVC method of svm class to use Support Vector Machine Algorithm

from sklearn.svm import SVC
classifier = SVC(kernel = 'linear', random_state = 0)
classifier.fit(X_train, Y_train)

#Using SVC method of svm class to use Kernel SVM Algorithm

from sklearn.svm import SVC
classifier = SVC(kernel = 'rbf', random_state = 0)
classifier.fit(X_train, Y_train)

#Using GaussianNB method of naïve_bayes class to use Naïve Bayes Algorithm

from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, Y_train)

#Using DecisionTreeClassifier of tree class to use Decision Tree Algorithm

from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
classifier.fit(X_train, Y_train)

#Using RandomForestClassifier method of ensemble class to use Random Forest Classification algorithm

```
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators = 10, criterion = 'entropy', random_state =
0)
classifier.fit(X_train, Y_train)
```

We will now predict the test set results and check the accuracy with each of our model:

```
Y_pred = classifier.predict(X_test)
```

To check the accuracy we need to import confusion_matrix method of metrics class. The confusion matrix is a way of tabulating the number of mis-classifications, i.e., the number of predicted classes which ended up in a wrong classification bin based on the true classes.

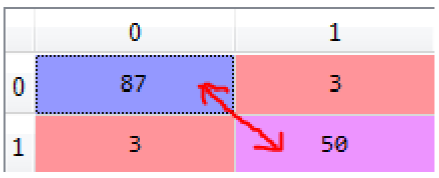
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(Y_test, Y_pred)
```

We will use Classification Accuracy method to find the accuracy of our models. Classification Accuracy is what we usually mean, when we use the term accuracy. It is the ratio of number of correct predictions to the total number of input samples.

$$Accuracy = \frac{Number\ of\ Correct\ predictions}{Total\ number\ of\ predictions\ made}$$

Fig: Accuracy

To check the correct prediction we have to check confusion matrix object and add the predicted results diagonally which will be number of correct prediction and then divide by total number of predictions.



	0	1
0	87	3
1	3	50

Fig: Confusion Matrix

1. Decision Tree for Classification

1.1 Business Problem Statement: predict whether a tumor is *malignant(cancer)* or *benign(non Cancer)* based on two features the mean radius of the tumor (radius_mean) and its mean number of concave points (concave points_mean)

Step 1: Import required Modules

```
import os
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score,confusion_matrix
DecisionTreeClassifier?
```

Step 2: Load data

```
#os.chdir("C:\\Users\\Hi\\Google Drive\\01 Data Science Lab Copy\\02 Lab Data\\Python")
df = pd.read_csv("Wisconsin_Breast_Cancer_Dataset.csv")
df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 33 columns):
id                    569 non-null int64
diagnosis             569 non-null object
```

```
radius_mean              569 non-null float64
texture_mean             569 non-null float64
perimeter_mean           569 non-null float64
area_mean                569 non-null float64
smoothness_mean          569 non-null float64
compactness_mean         569 non-null float64
concavity_mean           569 non-null float64
concave points_mean      569 non-null float64
symmetry_mean            569 non-null float64
fractal_dimension_mean   569 non-null float64
radius_se                569 non-null float64
texture_se               569 non-null float64
perimeter_se             569 non-null float64
area_se                  569 non-null float64
smoothness_se            569 non-null float64
compactness_se           569 non-null float64
concavity_se             569 non-null float64
concave points_se        569 non-null float64
symmetry_se              569 non-null float64
fractal_dimension_se     569 non-null float64
radius_worst             569 non-null float64
texture_worst            569 non-null float64
perimeter_worst          569 non-null float64
area_worst               569 non-null float64
smoothness_worst         569 non-null float64
compactness_worst        569 non-null float64
concavity_worst          569 non-null float64
concave points_worst     569 non-null float64
symmetry_worst           569 non-null float64
fractal_dimension_worst  569 non-null float64
Unnamed: 32              0 non-null float64
dtypes: float64(31), int64(1), object(1)
memory usage: 146.8+ KB
X = df[["radius_mean","concave points_mean"]]
y=df["diagnosis"]
X[15:25]
```

	radius_mean	concave points_mean
15	14.540	0.07364
16	14.680	0.05259
17	16.130	0.10280

18	19.810	0.09498
19	13.540	0.04781
20	13.080	0.03110
21	9.504	0.02076
22	15.340	0.09756
23	21.160	0.08632
24	16.650	0.09170

```
y[15:25]
15    M
16    M
17    M
18    M
19    B
20    B
21    B
22    M
23    M
24    M
Name: diagnosis, dtype: object

y = y.replace('M',1)
y = y.replace('B',0)
SEED = 1 # for reproducing
```

Step 3: Create training and test sets

```
X_train, X_test, y_train, y_test = train_test_split(X,
                                   y,test_size = 0.2,random_state=SEED,stratify=y)

print(X_train.shape) # (455, 2)
print(y_train.shape) # (455,)
print(X_test.shape) # (114, 2)
print(y_test.shape) # (114,)
(455, 2)
(455,)
(114, 2)
(114,)
```

Step 4: Create DecisionTreeClassifier Model with a maximum depth of 6

```
dt = DecisionTreeClassifier(max_depth=6,
                            random_state=SEED,
                            criterion='gini')
# Fit dt to the training set
dt.fit(X_train, y_train)
Out[10]:

DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=6,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=1,
            splitter='best')
```

Step 5: Predict test set labels using model

```
y_pred = dt.predict(X_test)
```

Step 6: Test the Performance

```
# Compute test set accuracy
acc = accuracy_score(y_test, y_pred)
print("Test set accuracy: {:.2f}".format(acc))
Test set accuracy: 0.89
print(confusion_matrix(y_test,y_pred))
[[65  7]
 [ 6 36]]
 (65+36)/114


0.8859649122807017
```

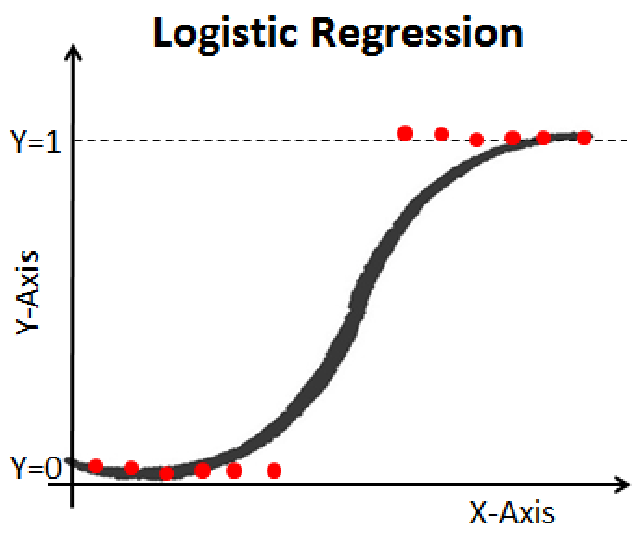
Note: Not bad! Using only two features, your tree was able to achieve an accuracy of 89% :)

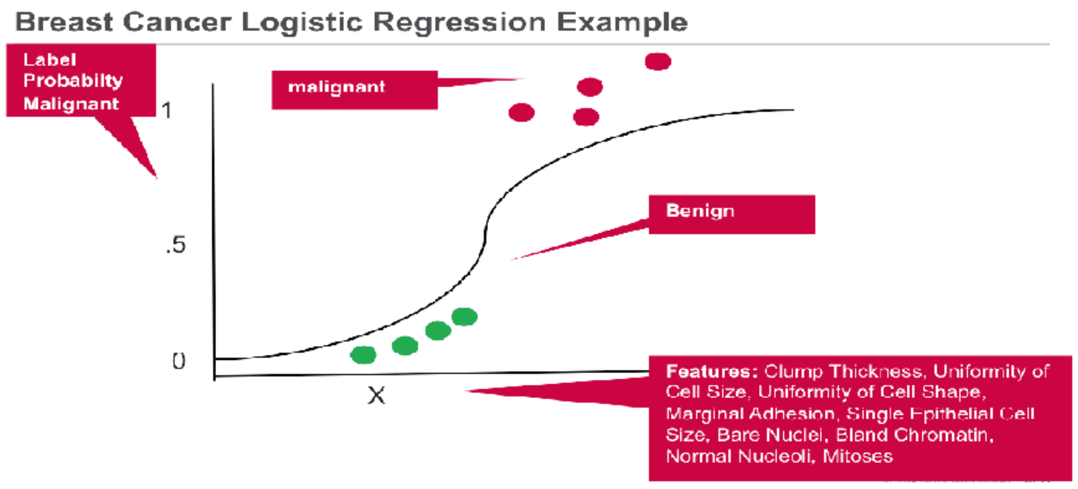
LogisticRegression Vs Decision Tree Classification

```
# Import LogisticRegression from sklearn.linear_model
from sklearn.linear_model import  LogisticRegression
# Instatiate logreg
logreg = LogisticRegression(random_state=1)
# Fit logreg to the training set
logreg.fit(X_train, y_train)

LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=1, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)
# predict
y_pred1 = logreg.predict(X_test)
acc1 = accuracy_score(y_test, y_pred1)
acc1


0.9122807017543859
```

By observing accuracies, On this data set which alogorithm is good ?

DT Classifier [✓ Logistic Regression]




Breast Cancer Logistic Regression Example
Label Probabilty Malignant
1
.5
0
malignant
Benign
X
Features: Clump Thickness, Uniformity of Cell Size, Uniformity of Cell Shape, Marginal Adhesion, Single Epithelial Cell Size, Bare Nuclei, Bland Chromatin, Normal Nucleoli, Mitoses

<u>Coding</u>

Step 1: Import Required Modules

```
import os
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score,confusion_matrix
```

Step 2: Load Data

```
#os.chdir("C:\\Users\\Hi\\Google Drive\\01 Data Science Lab Copy\\02 Lab Data\\Python")
df = pd.read_csv("Wisconsin_Breast_Cancer_Dataset.csv")
df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 33 columns):
id                         569 non-null int64
diagnosis                  569 non-null object
radius_mean                569 non-null float64
texture_mean               569 non-null float64
perimeter_mean             569 non-null float64
area_mean                  569 non-null float64
smoothness_mean            569 non-null float64
compactness_mean           569 non-null float64
concavity_mean             569 non-null float64
concave points_mean        569 non-null float64
symmetry_mean              569 non-null float64
fractal_dimension_mean     569 non-null float64
radius_se                  569 non-null float64
texture_se                 569 non-null float64
perimeter_se               569 non-null float64
area_se                    569 non-null float64
smoothness_se              569 non-null float64
compactness_se             569 non-null float64
concavity_se               569 non-null float64
concave points_se          569 non-null float64
symmetry_se                569 non-null float64
fractal_dimension_se       569 non-null float64
radius_worst               569 non-null float64
texture_worst              569 non-null float64
perimeter_worst            569 non-null float64
area_worst                 569 non-null float64
smoothness_worst           569 non-null float64
compactness_worst          569 non-null float64
concavity_worst            569 non-null float64
concave points_worst       569 non-null float64
symmetry_worst             569 non-null float64
fractal_dimension_worst    569 non-null float64
Unnamed: 32                0 non-null float64
dtypes: float64(31), int64(1), object(1)
memory usage: 146.8+ KB
X = df.iloc[:,2:32]
type(X)


pandas.core.frame.DataFrame
```

```
X.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 30 columns):
radius_mean                569 non-null float64
texture_mean               569 non-null float64
perimeter_mean             569 non-null float64
area_mean                  569 non-null float64
smoothness_mean            569 non-null float64
compactness_mean           569 non-null float64
concavity_mean             569 non-null float64
concave points_mean        569 non-null float64
symmetry_mean              569 non-null float64
fractal_dimension_mean     569 non-null float64
radius_se                  569 non-null float64
texture_se                 569 non-null float64
perimeter_se               569 non-null float64
area_se                    569 non-null float64
smoothness_se              569 non-null float64
compactness_se             569 non-null float64
concavity_se               569 non-null float64
concave points_se          569 non-null float64
symmetry_se                569 non-null float64
fractal_dimension_se       569 non-null float64
radius_worst               569 non-null float64
texture_worst              569 non-null float64
perimeter_worst            569 non-null float64
area_worst                 569 non-null float64
smoothness_worst           569 non-null float64
compactness_worst          569 non-null float64
concavity_worst            569 non-null float64
concave points_worst       569 non-null float64
symmetry_worst             569 non-null float64
fractal_dimension_worst    569 non-null float64
dtypes: float64(30)
memory usage: 133.4 KB
y=df["diagnosis"]
y[15:25]
15    M
16    M
17    M
18    M
19    B
20    B
21    B
22    M
23    M
24    M
```

```
Name: diagnosis, dtype: object
y = y.replace('M',1)
y = y.replace('B',0)
SEED = 1 # for reproducing
```

Step 3: Create Training and Test sets

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
                test_size = 0.2,random_state=SEED,stratify=y)
```

```
X_train.shape # (455, 2)
```

```
(455, 30)
```

```
y_train.shape # (455,)
```

```
(455,)
```

```
X_test.shape # (114, 2)
```

```
(114, 30)
```

```
y_test.shape # (114,)
```

```
(114,)
```

Step 4: Create Model using criterion as *entropy*

```
# Create dt_entropy model, set 'entropy' as the information criterion
dt_entropy = DecisionTreeClassifier(max_depth=8,
                              criterion='entropy',
                              random_state=SEED)
# Fit dt_entropy to the training set
dt_entropy.fit(X_train, y_train)
```

```
DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=8,
          max_features=None, max_leaf_nodes=None,
          min_impurity_decrease=0.0, min_impurity_split=None,
          min_samples_leaf=1, min_samples_split=2,
          min_weight_fraction_leaf=0.0, presort=False, random_state=1,
          splitter='best')
```

```
# Use dt_entropy to predict test set labels
y_pred = dt_entropy.predict(X_test)
# Evaluate accuracy_entropy
accuracy_entropy = accuracy_score(y_test, y_pred)
accuracy_entropy
```

```
0.9298245614035088
```

Step 5: Create Model using criterion as gini

```
# Instantiate dt_gini, set 'gini' as the information criterion
```

```
dt_gini= DecisionTreeClassifier(max_depth=8,
                               criterion='gini',
                               random_state=SEED)
# Fit dt_entropy to the training set
dt_gini.fit(X_train, y_train)
```

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=8,
          max_features=None, max_leaf_nodes=None,
          min_impurity_decrease=0.0, min_impurity_split=None,
          min_samples_leaf=1, min_samples_split=2,
          min_weight_fraction_leaf=0.0, presort=False, random_state=1,
          splitter='best')
```

```
# Use dt_entropy to predict test set labels
y_pred_gini = dt_gini.predict(X_test)
# Evaluate accuracy_gini
accuracy_gini = accuracy_score(y_test, y_pred_gini)
accuracy_gini
```

```
0.9298245614035088
```

Step 6: compare entropy and gini accuracy

```
# Print accuracy_entropy
print('Accuracy achieved by using entropy: ', accuracy_entropy)
# Print accuracy_gini
print('Accuracy achieved by using the gini index: ', accuracy_gini)
```

```
Accuracy achieved by using entropy:  0.9298245614035088
Accuracy achieved by using the gini index:  0.9298245614035088
```

Note: Notice how the two models achieve exactly the same accuracy. Most of the time, the gini index and entropy lead to the same results. The gini index is slightly faster to compute and is the default criterion used in the DecisionTreeClassifier model of scikit-learn.

After applying the different classification models, we have got below accuracies with different models:

1.Logistic Regression—95.8%

2. Nearest Neighbor—95.1%

3. Support Vector Machines—97.2%

4. Kernel SVM—96.5%

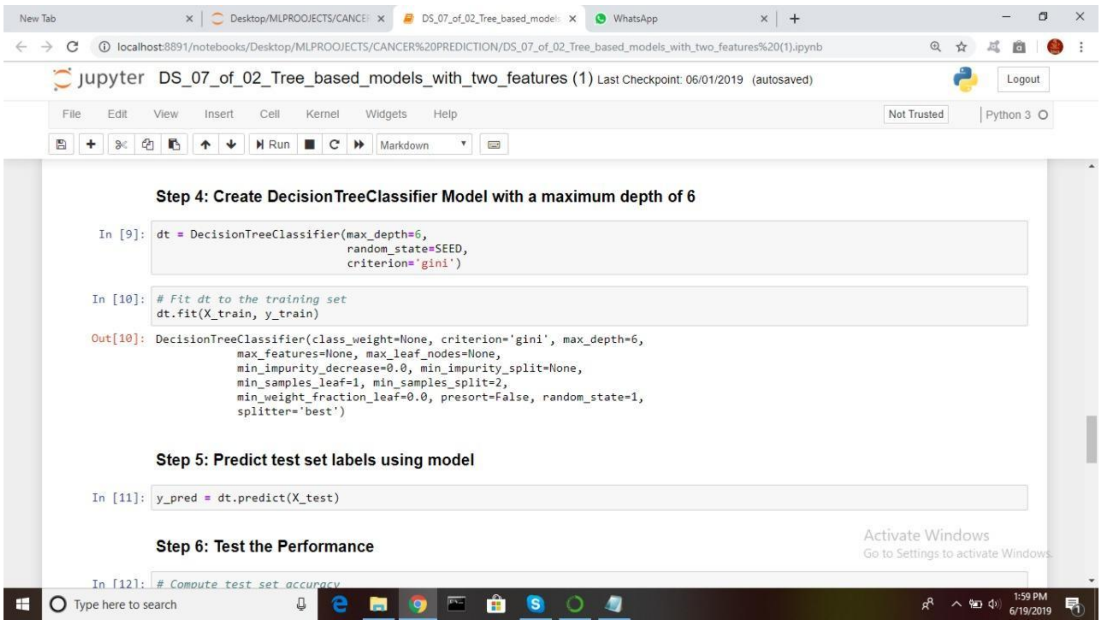
5. Naive Bayes—91.6%

6. Decision Tree Algorithm—95.8%

7. Random Forest Classification—98.6%

So finally we have built our classification model and we can see that Random Forest Classification algorithm gives the best results for our dataset. Well its not always applicable to every dataset. To choose our model we always need to analyze our dataset and then apply our machine learning model.

OUTCOMES:


New Tab
Desktop/MLPROOJECTS/CANCE
DS_07_of_02_Tree_based_model
WhatsApp
localhost:8891/notebooks/Desktop/MLPROOJECTS/CANCER%20PREDICTION/DS_07_of_02_Tree_based_models_with_two_features%20(1).ipynb
jupyter DS_07_of_02_Tree_based_models_with_two_features (1) Last Checkpoint: 06/01/2019 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Markdown
Activate Windows
Go to Settings to activate Windows.
Type here to search
1:59 PM
6/19/2019

Step 4: Create DecisionTreeClassifier Model with a maximum depth of 6

```
In [9]: dt = DecisionTreeClassifier(max_depth=6,
                                    random_state=SEED,
                                    criterion='gini')
```

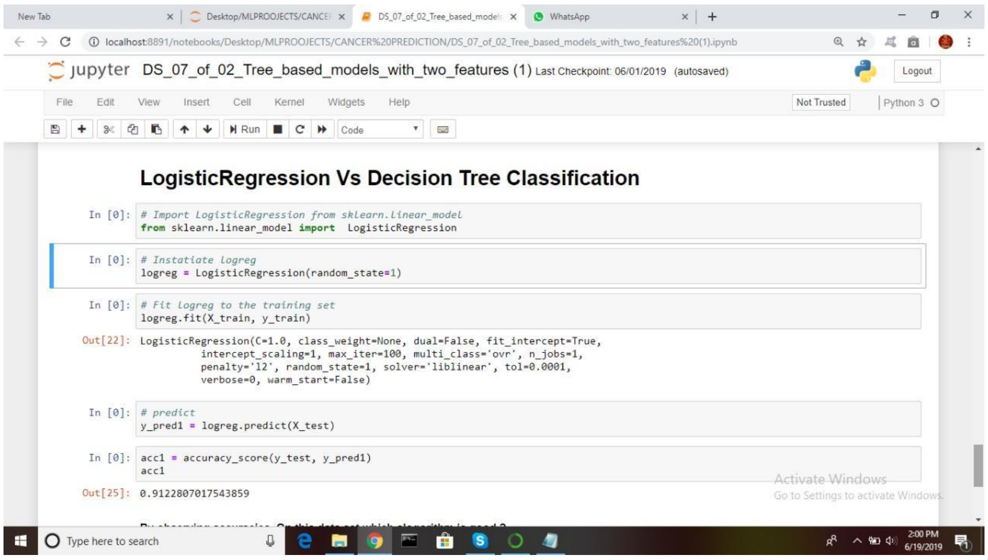
```
In [10]: # Fit dt to the training set
         dt.fit(X_train, y_train)
Out[10]: DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=6,
                     max_features=None, max_leaf_nodes=None,
                     min_impurity_decrease=0.0, min_impurity_split=None,
                     min_samples_leaf=1, min_samples_split=2,
                     min_weight_fraction_leaf=0.0, presort=False, random_state=1,
                     splitter='best')
```

Step 5: Predict test set labels using model

```
In [11]: y_pred = dt.predict(X_test)
```

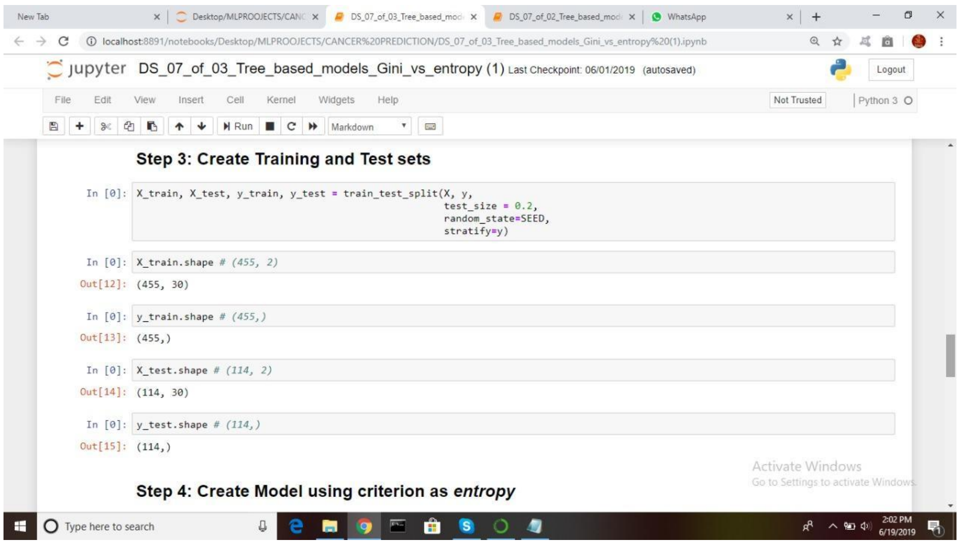
Step 6: Test the Performance

```
In [12]: # Compute test set accuracy
```


localhost:8891/notebooks/Desktop/MLPROJECTS/CANCER%20PREDICTION/DS_07_of_02_Tree_based_models_with_two_features%20(1).ipynb
jupyter DS_07_of_02_Tree_based_models_with_two_features (1) Last Checkpoint: 06/01/2019 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code
Activate Windows
Go to Settings to activate Windows.
Type here to search
2:00 PM
6/19/2019

LogisticRegression Vs Decision Tree Classification

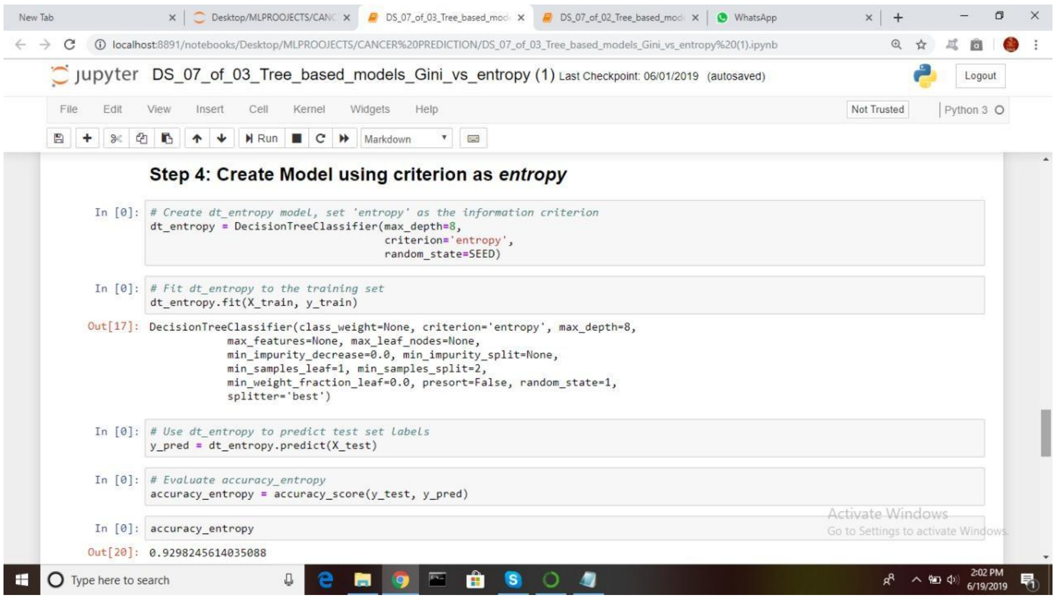
```
In [0]: # Import LogisticRegression from sklearn.linear_model
        from sklearn.linear_model import  LogisticRegression

In [0]: # Instatiate logreg
        logreg = LogisticRegression(random_state=1)

In [0]: # Fit logreg to the training set
        logreg.fit(X_train, y_train)

Out[22]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
                   penalty='l2', random_state=1, solver='liblinear', tol=0.0001,
                   verbose=0, warm_start=False)

In [0]: # predict
        y_pred1 = logreg.predict(X_test)

In [0]: acc1 = accuracy_score(y_test, y_pred1)
        acc1

Out[25]: 0.9122807017543859
```


localhost:8891/notebooks/Desktop/MLPROJECTS/CANCER%20PREDICTION/DS_07_of_03_Tree_based_models_Gini_vs_entropy%20(1).ipynb
jupyter DS_07_of_03_Tree_based_models_Gini_vs_entropy (1) Last Checkpoint: 06/01/2019 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Markdown
Activate Windows
Go to Settings to activate Windows.
Type here to search
2:02 PM
6/19/2019

Step 3: Create Training and Test sets

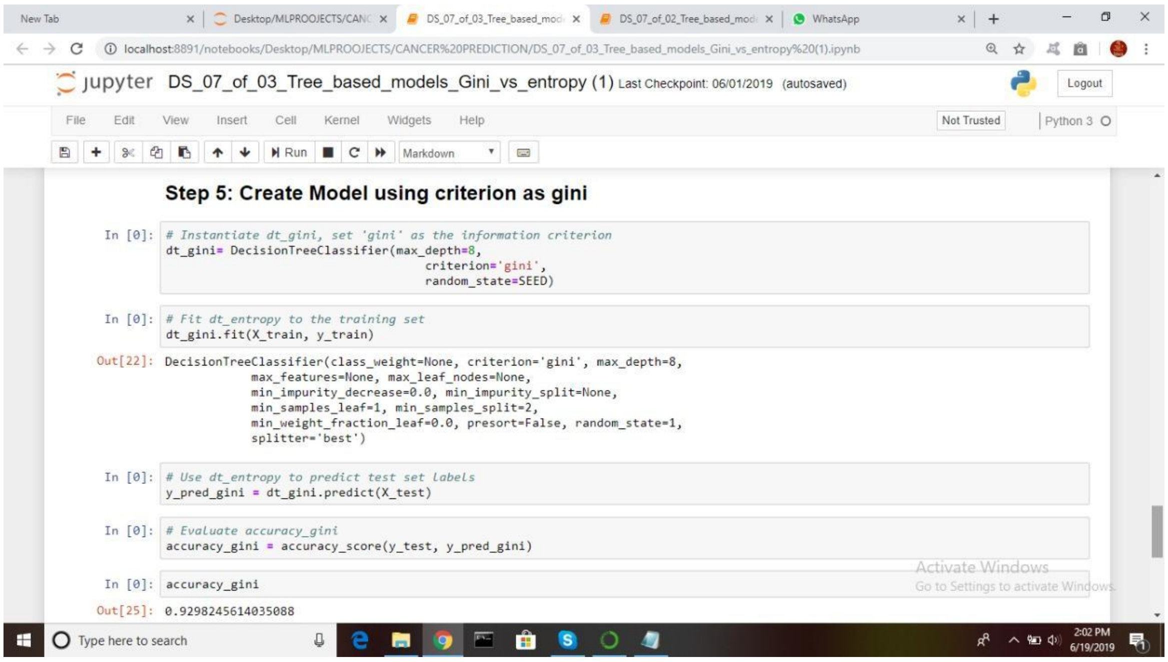
```
In [0]: X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                            test_size = 0.2,
                                                            random_state=SEED,
                                                            stratify=y)

In [0]: X_train.shape # (455, 2)

Out[12]: (455, 30)

In [0]: y_train.shape # (455,)

Out[13]: (455,)

In [0]: X_test.shape # (114, 2)

Out[14]: (114, 30)

In [0]: y_test.shape # (114,)

Out[15]: (114,)
```

Step 4: Create Model using criterion as *entropy*


New Tab
Desktop/MLPROOJECTS/CANC
DS_07_of_03_Tree_based_mod
DS_07_of_02_Tree_based_mod
WhatsApp
localhost:8891/notebooks/Desktop/MLPROOJECTS/CANCER%20PREDICTION/DS_07_of_03_Tree_based_models_Gini_vs_entropy%20(1).ipynb
jupyter DS_07_of_03_Tree_based_models_Gini_vs_entropy (1) Last Checkpoint: 06/01/2019 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Markdown
Activate Windows
Go to Settings to activate Windows.
Type here to search
2:02 PM
6/19/2019

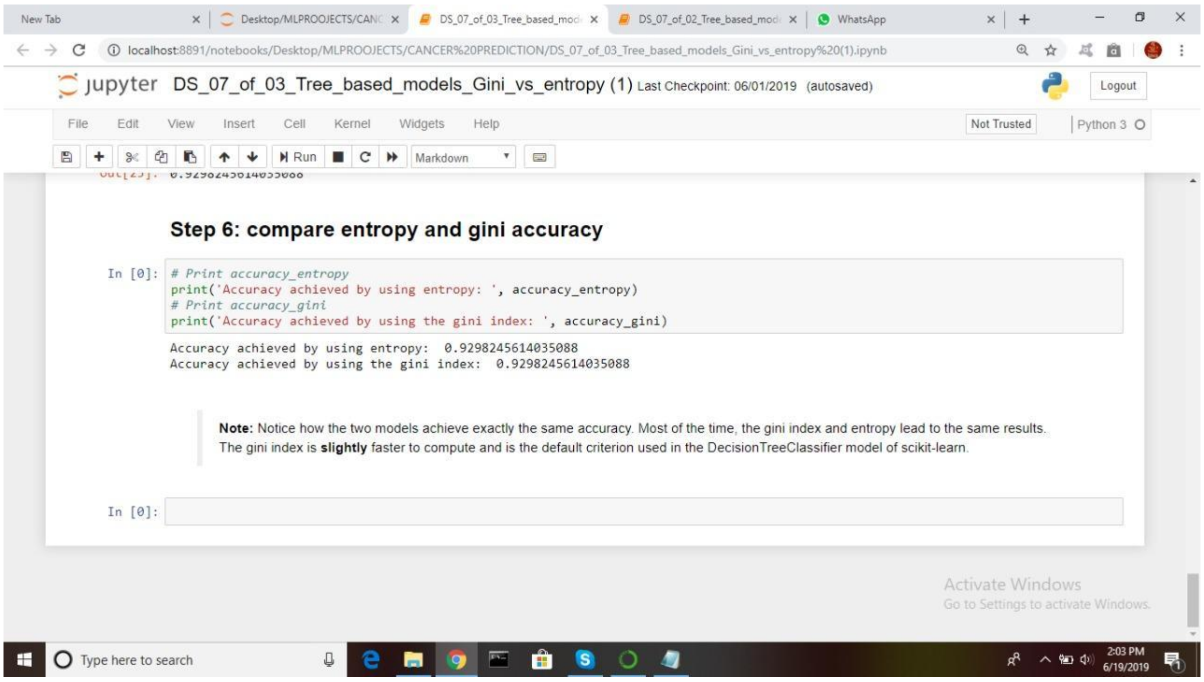
Step 4: Create Model using criterion as *entropy*

```
In [0]: # Create dt_entropy model, set 'entropy' as the information criterion
        dt_entropy = DecisionTreeClassifier(max_depth=8,
                                            criterion='entropy',
                                            random_state=SEED)

In [0]: # Fit dt_entropy to the training set
        dt_entropy.fit(X_train, y_train)

Out[17]: DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=8,
                     max_features=None, max_leaf_nodes=None,
                     min_impurity_decrease=0.0, min_impurity_split=None,
                     min_samples_leaf=1, min_samples_split=2,
                     min_weight_fraction_leaf=0.0, presort=False, random_state=1,
                     splitter='best')

In [0]: # Use dt_entropy to predict test set labels
        y_pred = dt_entropy.predict(X_test)

In [0]: # Evaluate accuracy_entropy
        accuracy_entropy = accuracy_score(y_test, y_pred)

In [0]: accuracy_entropy

Out[20]: 0.9298245614035088
```


New Tab
Desktop/MLPROOJECTS/CANC
DS_07_of_03_Tree_based_mod
DS_07_of_02_Tree_based_mod
WhatsApp
localhost:8891/notebooks/Desktop/MLPROOJECTS/CANCER%20PREDICTION/DS_07_of_03_Tree_based_models_Gini_vs_entropy%20(1).ipynb
jupyter DS_07_of_03_Tree_based_models_Gini_vs_entropy (1) Last Checkpoint: 06/01/2019 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Markdown
Activate Windows
Go to Settings to activate Windows.
Type here to search
2:02 PM
6/19/2019

Step 5: Create Model using criterion as gini

```
In [0]: # Instantiate dt_gini, set 'gini' as the information criterion
        dt_gini= DecisionTreeClassifier(max_depth=8,
                                        criterion='gini',
                                        random_state=SEED)

In [0]: # Fit dt_entropy to the training set
        dt_gini.fit(X_train, y_train)

Out[22]: DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=8,
                     max_features=None, max_leaf_nodes=None,
                     min_impurity_decrease=0.0, min_impurity_split=None,
                     min_samples_leaf=1, min_samples_split=2,
                     min_weight_fraction_leaf=0.0, presort=False, random_state=1,
                     splitter='best')

In [0]: # Use dt_entropy to predict test set labels
        y_pred_gini = dt_gini.predict(X_test)

In [0]: # Evaluate accuracy_gini
        accuracy_gini = accuracy_score(y_test, y_pred_gini)

In [0]: accuracy_gini

Out[25]: 0.9298245614035088
```

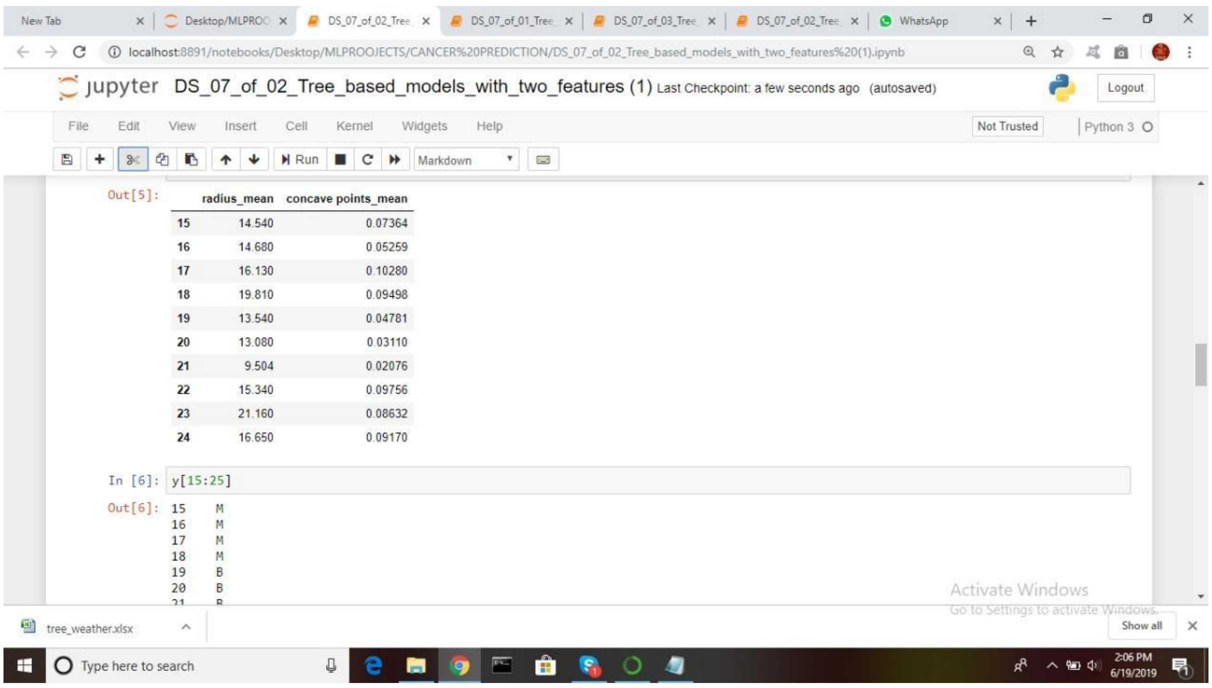

jupyter DS_07_of_03_Tree_based_models_Gini_vs_entropy (1) Last Checkpoint: 06/01/2019 (autosaved)
Logout
Not Trusted
Python 3
Activate Windows
Go to Settings to activate Windows.

Step 6: compare entropy and gini accuracy

```
In [0]: # Print accuracy_entropy
        print('Accuracy achieved by using entropy: ', accuracy_entropy)
        # Print accuracy_gini
        print('Accuracy achieved by using the gini index: ', accuracy_gini)

        Accuracy achieved by using entropy:  0.9298245614035088
        Accuracy achieved by using the gini index:  0.9298245614035088
```

> **Note:** Notice how the two models achieve exactly the same accuracy. Most of the time, the gini index and entropy lead to the same results. The gini index is **slightly** faster to compute and is the default criterion used in the DecisionTreeClassifier model of scikit-learn.

```
In [0]:
```


jupyter DS_07_of_02_Tree_based_models_with_two_features (1) Last Checkpoint: a few seconds ago (autosaved)
Logout
Not Trusted
Python 3
tree_weather.xlsx
Show all
Activate Windows
Go to Settings to activate Windows.

Out[5]:

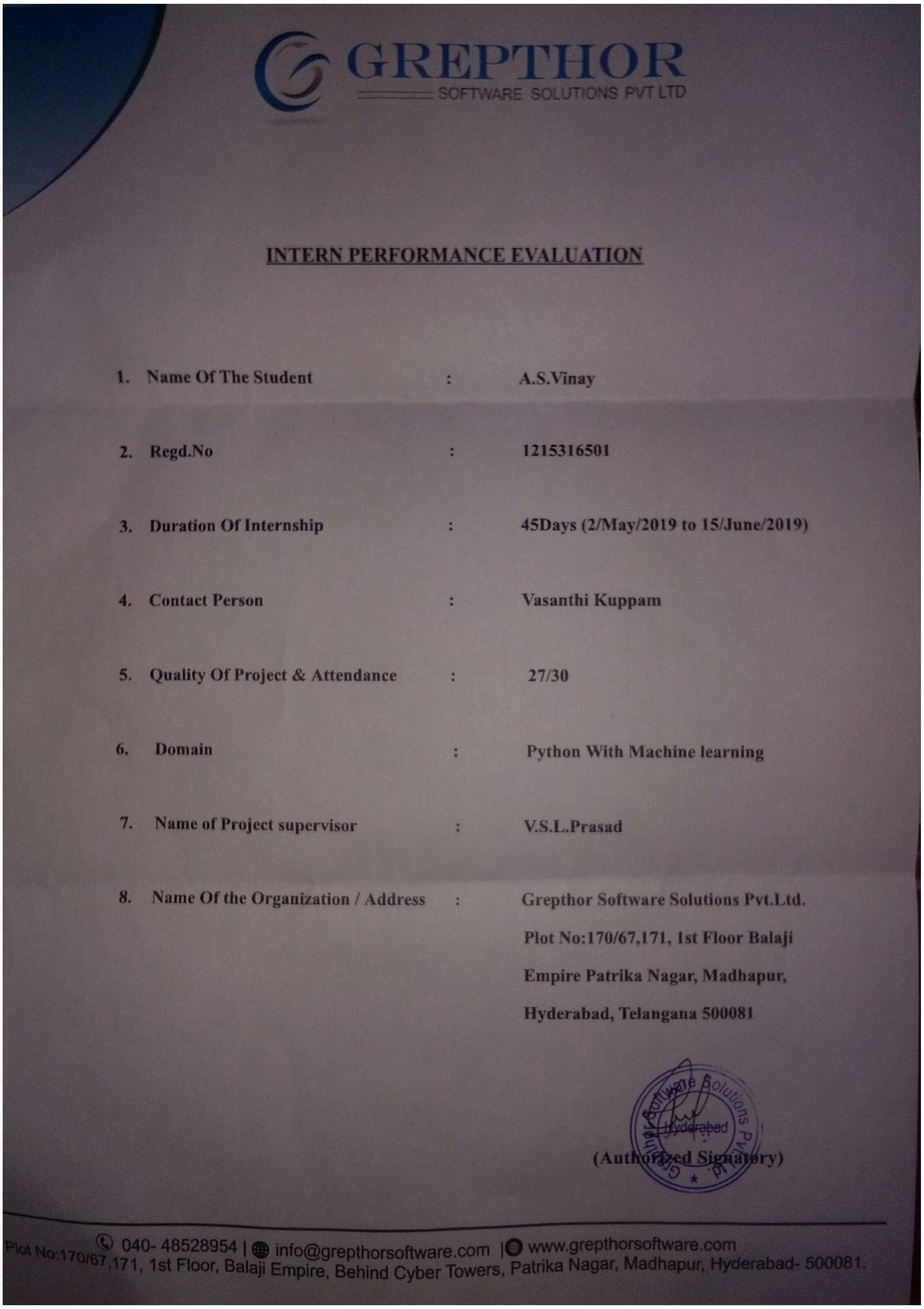
	radius_mean	concave points_mean
15	14.540	0.07364
16	14.680	0.05259
17	16.130	0.10280
18	19.810	0.09498
19	13.540	0.04781
20	13.080	0.03110
21	9.504	0.02076
22	15.340	0.09756
23	21.160	0.08632
24	16.650	0.09170

```
In [6]: y[15:25]

Out[6]: 15    M
        16    M
        17    M
        18    M
        19    B
        20    B
```

CONCLUSION:

In the present paper, breast cancer and ML were introduced as well as an in-depth literature review was performed on existing ML methods used for breast cancer detection. The findings of these researchers suggest that SVM is the most popular method used for cancer detection applications. SVM was used either alone or combined with another method to improve the performance. The maximum achieved accuracy of SVM (single or hybrid) was 99.8% that can be improved to 100%. It was observed from the work of who used optional ANN   on MRI resulted in 100% accuracy in detecting breast cancer. This method can be applied and tested on another dataset like mammogram and ultrasound to check the performance of different data types. The mammogram was the most frequent data set used compared to other types of data such as ultrasound images,boold features.

ASSESSMENT OF INTERNSHIP:



GREPTHOR
SOFTWARE SOLUTIONS PVT LTD

INTERN PERFORMANCE EVALUATION

1.	Name Of The Student	:	A.S.Vinay
2.	Regd.No	:	1215316501
3.	Duration Of Internship	:	45Days (2/May/2019 to 15/June/2019)
4.	Contact Person	:	Vasanthi Kuppam
5.	Quality Of Project & Attendance	:	27/30
6.	Domain	:	Python With Machine learning
7.	Name of Project supervisor	:	V.S.L.Prasad
8.	Name Of the Organization / Address	:	Grepthor Software Solutions Pvt.Ltd. Plot No:170/67,171, 1st Floor Balaji Empire Patrika Nagar, Madhapur, Hyderabad, Telangana 500081

Grepthor Software Solutions Pvt. Ltd. Hyderabad

[illegible]

(Authorized Signatory)

040- 48528954 | info@grepthorsoftware.com | www.grepthorsoftware.com
Plot No:170/67,171, 1st Floor, Balaji Empire, Behind Cyber Towers, Patrika Nagar, Madhapur, Hyderabad- 500081.

REFERENCES:

1.World Health Organization, “Cancer country profiles 2014,” WHO, http://www.who.int/cancer/country-profiles/en.

2.M. Stalin, and R. Kalaimagal, “Breast cancer diagnosis from low-intensity asymmetry thermogram breast images using fast support vector machine,” i-manager's Journal on Image Processing, vol. 3, no.

3, pp. 17–26, 2016.

3.R. Kirubakaran, T. C. Jia, and N. M. Aris, “Awareness of Breast Cancer among Surgical Patients in a Tertiary Hospital in Malaysia,” Asian Pacific Journal of Cancer Prevention, 2017, vol. 18, no. 1, pp. 115-120.

4.B. Stewart and C.P. Wild, World Cancer Report 2014, International Agency for Research on Cancer, WHO, 2014.