VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

FOUNDATIONS OF AI

CSE1014

J COMPONENT REPORT

PROJECT-TITLE: ***Prediction of Diseases using Machine Learning Techniques.***

FACULTY: ***Dr. Priyadarshini J***

COURSE-CODE/SLOT:***CSE1014 / G2 /Winter Semester 2021-2022***

TEAM MEMBERS:

Mandla Sheshi Kiran Reddy - 20BAI1061

Uppanapalli Lakshmi Sowjanya- 20BAI1289

Jahnavi Thondepu - 20BAI1150

ACKNOWLEDGMENT

We as a team are very grateful and is a great pleasure to undertake this project as a J component for CSE1014 subject which is entitled – "***Prediction of Diseases using Machine Learning Techniques***."

We are very grateful to our faculty member ***Dr. Priyadarshini J***. This project would not have been possible without her enormous help and worthy guidance whenever it is needed. Her support and suggestions proved valuable in enabling the successful completion of the project.

-TEAM MEMBERS

Mandla Sheshi Kiran Reddy - 20BAI1061

Uppanapalli Lakshmi Sowjanya- 20BAI1289

Jahnavi Thondepu - 20BAI1150

CONTENTS

INTRODUCTION

Machine Learning is the domain that uses past data for predicting. Machine Learning is the understanding of computer system under which the Machine Learning model learn from data and experience. The machine learning algorithm has two phases: ‘
1) Training & 2) Testing.
To predict the disease from a patient’s symptoms and from the history of the patient, machine learning technology is struggling from past decades. Healthcare issues can be solved efficiently by using Machine Learning Technology. We are applying complete machine learning concepts to keep the track of patient’s health. ML model allows us to build models to get quickly cleaned and processed data and deliver results faster. By using this system doctors will make good decisions related to patient diagnoses and according to that, good treatment will be given to the patient, which increases improvement in patient healthcare services. To introduce machine learning in the medical field, healthcare is the prime example. To improve the accuracy of large data, the existing work will be done on unstructured or textual data. For the prediction of diseases, the existing will be done on Decision trees, logistic regression, SVM and Random Forest.

ABSTRACT

Disease Prediction using Machine Learning is the system that is used to predict the diseases from the symptoms which are given by the patients or any user. The system processes the symptoms/records provided by the user as input and gives the output as the probability of the disease.With an increase in biomedical and healthcare data, accurate analysis of medical data benefits early disease detection and patient care.We have used the classification techniques such as *SVM, Random Forest, Decision Trees, Logistic regression.*and then we used Random Forest machine learning algorithm to predict the diseases “Lung Cancer” and “Breast Cancer” as it reached the maximum accuracy of 98.3%.

KEYWORDS

Lung cancer, breast cancer, SVM, Random Forest, Decision Trees, Logistic regression, accuracy

LITERATURE REVIEW

In the paper [1],

The key objective of this paper is the early diagnosis of lung cancer by examining the performance of classification algorithms

ADVANTAGES:
A comparative analysis of accuracy rates of each classifier are presented. The predictive performance of classifiers are compared quantitatively. In the performance chart, different results are produced for each classifier on the lung cancer dataset. The best result is given by the support vector machine algorithm. SVM algorithm used high dimension to classify the observation so it's performance is the best. More accurate lung cancer detection can be done using this technique.So they have used several classification techniques such as the SVM, decision Tree and Naive Bayes classification algorithms for the prediction.They got the best result for the Support vector machines.

DISADVANTAGES:
The more preprocessing of the dataset , we can achieve more accuracy so there is still more chance to increase the accuracy.Also the SVM accuracy is nearly 99.2% percent where there are chances of overfitting of the data.SVM algorithm is not suitable for large data sets.SVM does not perform very well when the data set has more noise i.e. target classes are overlapping.In cases where the number of features for each data point exceeds the number of training data samples, the SVM will underperform.As the support vector classifier works by putting data points,above and below the classifying hyperplane there is no probabilistic explanation for the classification.So the SVM may work for the smaller datasets but incase it will be used as the model based agent where the knowledge base has to be increased and the dataset also increases and also the accuracy is nearly equal to ~1.0 (0.992) so there is also chance of overfitting.So these are the disadvantages seen in the mentioned research paper.

OVERCOMING LIMITATIONS:
We have used the ensemble learning methods such as the "random forest" which gives the best accuracy for these type of datasets.We also compared the other classification methods such as the logistic regression,decision trees.We got the best output for the random forest.As the random forest has the ,whether you have a regression or classification task, random forest is an applicable model for your needs. It can handle binary features, categorical features, and numerical features. There is very little pre-processing that needs to be done. The data does not need to be rescaled or transformed.They are parallelizable, meaning that we can split the process to multiple machines to run. This results in faster computation time. Boosted models are sequential in contrast, and would take longer to compute.Random forests is great with high dimensional data since we are working with subsets of data.We also done the hyper parameter tuning to achieve the best accuracy for all the algorithms such as we increased the accuracy of the logistic regression from 0.66(in the research paper) to 0.79(in our project) and achieved the better results.
So we used the random forest algorithm to achieve the best results with accuracy of ~[0.98] by hyper parameter tuning and the proper preprocessing steps.

ML Algorithm	Accuracy(%)
Logistic Regression	66.7
Decision Tree	90
Naïve Bayes	87.87
SVM	99.2

In paper [2],
The classification is based on a multi-class logistic regression algorithm. Accuracy of 100% was achieved at 10,000 iterations with learning rate set to 0.09. The above lung cancer dataset was also applied to KNN classifiers. As the number of K neighbours increased from 2 to 30, accuracy degraded from 99.7 to 89.44%

ADVANTAGES:
The lung cancer dataset has been classified and made available for prediction analysis using the two algorithms , which are the KNN classifier and the Logistic regression.They properly pre processed the dataset and they applied the classification techniques and concluded that the logistic regression is working better than the knn classifier.As the logistic regression makes no assumptions about distributions of classes in feature space and it can It can easily extend to multiple classes(multinomial regression) and a natural probabilistic view of class predictions.

DISADVANTAGES:
They have only checked the two classification methods where they got overfitting of the data for the KNN classifier given K=2. As the K increased the accuracy is nearly 89% which is not up to the mark as this is the disease prediction application and needs to be more accurate.When performed the logistic regression the accuracy we got is 95% for 500 iterations and for the 10000 iterations it is again overfitting.The major limitation of Logistic Regression is the assumption of linearity between the dependent variable and the independent variables.
So the two classification techniques used are prone to the overfitting of the data and may not give better results for the predictions if there is a variation from the dataset values.

OVERCOMING THE LIMITATION:
The limitation of the research paper mentioned is the overfitting of the classification techniques used.So we properly pre processed the data and mainly the part lies in the hyper parameter tuning.Even though we did not use the KNN classifier in our project we have used the logistic regression and achieved the accuracy of nearly 0.80 without any overfitting and we avoided the overfitting for logistic regression but we got the better results for the random forest technique which is ~0.98.

Table 1 KNN classifier applied on lung cancer dataset

Classifier KNN(K)	Accuracy high risk class (365)	Accuracy low risk class (303)	Accuracy medium risk class (332)	Overall accuracy
$K = 2$	100	99.67	100	99.89
$K = 5$	100	99	100	99.67
$K = 10$	100	99	100	99.67
$K = 15$	100	99	100	99.67
$K = 20$	94.8	85.8	82.2	87.6
$K = 25$	100	86.1	82.5	89.5
$K = 30$	100	85.8	82.5	89.44

Table 2 Logistic regression applied on lung cancer dataset

Number of iterations	Min cost function in high, low, medium classes	Accuracy high risk class (365)	Accuracy low risk class (303)	Accuracy medium risk class (332)	Overall Acc
500	0.0384, 0.0635, 0.2321,	100	96.7	90.97	95.88
1000	0.0226, 0.0392, 0.2122	100	99.7	90.97	96.87
1500	0.01614, 0.02842, 0.2019	100	100	93.67	97.89
2000	0.0125, 0.0222, 0.1947	100	100	93.67	97.89
10,000	0.0028, 0.0051, 0.146	100	100	100	100

In paper [3],
The methods they used are artificial neural networks (ANNs), support vector machines (SVMs) and decision trees (DTs). While we can understand cancer progression with the use of ML methods, an adequate validity level is needed to take these methods into consideration in clinical practice every day. In this study, the ML & DL approaches used in cancer progression modelling are reviewed. The predictions addressed are mostly linked to specific ML, input, and data samples supervision.

ADVANTAGES:
The whole study explains and compares the findings of various machine learning and in-depth learning implemented to cancer prognosis. Specifically, several trends related to those same kinds of machines techniques to be used, the kinds of training data to be incorporated, the kind of endpoint forecasts to be made, sorts of cancers being investigated, and the overall performance of cancer prediction oroutcome methods have been identified.While the ANNs are common, it is clear that a broader variety of alternative learning approaches is also used to predict at least three different cancer types. ANNs continue to be prevalent. Furthermore, it is clear that machine training methods typically increase the efficiency or predictable accuracy of most pronostics, in particular when matched with conventional statistical or expert systems.

DISADVANTAGES:
Decision trees are simple to interpret, it should be taken as the minimal decision standard of work-relatedness for lung cancer, is the best predictor by attaining high accuracy, and it automatically prunes to a very short three-level depth. However, the running time of training algorithms do not scale well with the size of the training set. SVM is used to build n-hyperlanes and n-features for dividing each different class apart from maximal margin, and it improves the classification power and robustness. Yet, many parameters need to be set accurately for attaining the best results. Gene Expression Programming has the better solution for predicting lung cancer difficulties, and has high accuracy. However there are some disadvantages such as if they are easy to manipulate, they lose in functional complexity.Here the methods they used have several disadvantages mentioned above , so we used the random forest method for the classification which is an ensemble learning method

OVECOMING LIMITATIONS:
Since the disadvantages are described for the decision trees,SVM etc , we used the random forest method which is the ensemble learning method for the prediction analysis.As the random forest has the ,whether you have a regression or classification task, random forest is an applicable model for your needs. It can handle binary features,

categorical features, and numerical features. There is very little pre-processing that needs to be done. The data does not need to be rescaled or transformed.They are parallelizable, meaning that we cansplit the process to multiple machines to run. This results in faster computation time. Boosted models are sequential in contrast, and would take longer to compute.Random forests are great with high dimensional data since we are working with subsets of data.
So we used the random forest algorithm to achieve the best results with accuracy of ~[0.98] by hyper parameter tuning and the proper preprocessing steps.

In paper [4]
They used many algorithms for classification and prediction of breast cancer: Support Vector Machine (SVM), Decision Tree (CART), Naive Bayes (NB) and k Nearest Neighbours (kNN). In this project, Support Vector Machine (SVM) on the Wisconsin Breast Cancer dataset is used. The dataset is also trained with the other algorithms: KNN, Naives Bayes and CART and the accuracy of prediction for each algorithm is compared.

ADVANTAGES:
For overall methodology, KNN technique has given the best results. Naive Bayes and logistic regression have also performed well in diagnosis of breast cancer. SVM is a strong technique for predictive analysis and owing to the above finding, we conclude that SVM using Gaussian kernel is the most suited technique for recurrence/non-recurrence prediction of breast cancer. The SVM that is used in the analysis in this paper is only applicable when the number of class variables is binary i.e. we can't have more than 2 classes.

DISADVANTAGES:
The algorithms used got almost the same accuracy that is 92% but the preprocessing of the data is not done properly.The standardisation of the input is to be done so that the accuracy can still be increased.There is also overfitting of the data which is not a good sign for the prediction analysis as the model will not work efficiently for the new instances.So our challenge is to overcome the overfitting and also search for the other best classification methods which will give us the best prediction model.

OVERCOMING LIMITATIONS:
We properly executed the preprocessing steps and also used the ensemble learning method "random forest" method.We also standardised the data so that the SVM method accuracy can be increased.We used the random forest ensemble learning method because in the ensemble learning methods every base model is trained on a different subset of data and all the results are combined, so the final model is less overfitted and variance is reduced.

When we calculate accuracy we observe the output to be as shown below:
Accuracy score 0.991228

TABLE II. RESULTS

Cancer type	Precision	Recall	F1-score	Support
0	1.00	0.99	0.99	75
1	0.97	1.00	0.99	39
Average / Total	0.99	0.99	0.99	114

F. Confusion Matrix:

```
    M  B
M [[74  1]
B  [0 39]]
```

In paper [5],
The performance of the study is measured with respect to accuracy, sensitivity, specificity, precision, negative predictive value, false-negative rate, false-positive rate, F1 score, and Matthews Correlation Coefficient. Additionally, these techniques were appraised on precision–recall area under curve and receiver operating characteristic curve. The results reveal that the ANNs obtained the highest accuracy, precision, and F1 score of 98.57%, 97.82%, and 0.9890, respectively, whereas 97.14%, 95.65%, and 0.9777 accuracy, precision, and F1 score are obtained by SVM, respectively.

ADVANTAGES:
This paper presented a comparative study of fve machine learning techniques for the prediction of breast cancer, namely support vector machine, K-nearest neighbours, random forests, artificial neural networks, and logistic regression. The basic features and working principle of each of the five machine learning techniques were illus-trated. The highest accuracy obtained by ANNs is 98.57% whereas the lowest accuracy derived from the RFs and LR is 95.7%. The diagnosis procedure in the medical field is very expensive as well as time-consuming. The system proposed that machine learning techniques can act as a clinical assistant for the diagnosis of breast cancer.

DISADVANTAGES:
The disadvantage of the above method is that this takes longer time for the computation and the ANN method is used for the classification which needs high power and the memory consumption.The longer time computation is one disadvantage. Artificial Neural Networks require processors with parallel processing power, by their structure.For this reason, the realisation of the equipment is dependent.Also,problems have to be translated into numerical values before being introduced to ANN.

OVERCOMING LIMITATIONS:
We overcame the disadvantage by using the ensemble learning method “random forest” and achieved the best accuracy of ~0.98 which is greater than the artificial neural networks algorithm.This takes less time to compute and also the efficiency is high than the ANN’s and the specific hardware requirement is not needed.So we overcame the disadvantage by exploring the random forest ensemble learning technique for the faster computation and the better accuracy.

Table 7 Performances of breast cancer prediction system

	SVM	K-NN	RF	ANN	LR
Accuracy (%)	97.14	97.14	95.71	98.57	95.71
Sensitivity (%)	100	97.82	95.65	100	95.74
Specificity (%)	92.3	95.83	95.83	96	95.65
Precision (%)	95.65	0.9782	0.9777	0.9782	0.9782
NPV (%)	100	95.83	92	100	91.66
FPR (%)	7.69	4.16	4.16	4	4.34
FNR (%)	0	2.17	4.34	0	4.25
F1 score	0.9777	0.9782	0.967	0.989	0.9677
MCC	0.9396	0.9365	0.9062	0.969	0.9043

PROPOSED METHODOLOGY

ATTRIBUTE (DATASET) DESCRIPTION:

1. LUNG CANCER

Age	Gender	Air_Pollution	Alcohol_Use	Dust_Allergy	OccuPational_Hazards	Genetic_Risk	Chronic_Lung_Diseas	Balanced_Diet	Obesity
33	Male	AQI INDEX 51-100	7-8 drinks per week	Itchy nose, roof of mouth or throat	dry cleaning jobs	no	stage-I early	Not Following regualy	BMI not Overweight
17	Male	AQI INDEX 101-150	0-2 drinks per week	Itchy nose, roof of mouth or throat	Textile jobs / convininece store jobs	no	stage-I early	Not Following regualy	BMI not Overweight
35	Male	AQI INDEX 151-200	9-10 drinks per week	cough	manufacturing and aerospace jobs	yes	stage-II Moderate	Following regualy	BMI Overweight
37	Male	AQI INDEX 301-350	13 - 14 drinks per week	facial pressure and pain	mining jobs / construction jobs / smelting jobs	yes	stage-IV Very Severe	Following regualy	BMI Overweight
46	Male	AQI INDEX 251-300	>14 drinks per week	facial pressure and pain	mining jobs / construction jobs / smelting jobs	yes	stage-III Severe	Following regualy	BMI Overweight
35	Male	AQI INDEX 151-200	9-10 drinks per week	cough	manufacturing and aerospace jobs	yes	stage-II Moderate	Following regualy	BMI Overweight
52	Female	AQI INDEX 51-100	7-8 drinks per week	Itchy nose, roof of mouth or throat	dry cleaning jobs	no	stage-I early	Not Following regualy	BMI not Overweight
28	Female	AQI INDEX 101-150	0-2 drinks per week	Nasal congestion	Textile jobs / convininece store jobs	no	stage-II Moderate	Not Following regualy	BMI not Overweight
35	Female	AQI INDEX 151-200	9-10 drinks per week	cough	manufacturing and aerospace jobs	yes	stage-III Severe	Following regualy	BMI Overweight
46	Male	AQI INDEX 51-100	5-6 drinks per week	Nasal congestion	Research jobs	no	stage-II Moderate	Not Following regualy	BMI not Overweight
44	Male	AQI INDEX 251-300	13 - 14 drinks per week	facial pressure and pain	mining jobs / construction jobs / smelting jobs	yes	stage-III Severe	Following regualy	BMI Overweight
64	Female	AQI INDEX 251-300	>14 drinks per week	facial pressure and pain	mining jobs / construction jobs / smelting jobs	yes	stage-III Severe	Following regualy	BMI Overweight
39	Female	AQI INDEX 151-200	9-10 drinks per week	cough	chemical industry jobs	yes	stage-II Moderate	Following regualy	BMI Overweight
34	Male	AQI INDEX 251-300	13 - 14 drinks per week	facial pressure and pain	mining jobs / construction jobs / smelting jobs	yes	stage-IV Very Severe	Following regualy	BMI Overweight

Age:
Age of the patient.
Average - 37

Gender:
Gender of the patient.

- Male
- Female

Air_pollution:

- 0-50 = 1
- 51-100 = 2
- 101-150 = 3
- 151-200 = 4
- 201-250 = 5
- 251-300 = 6
- 301-350 = 7
- >350 = 8

Alcohol_Use:

Drinking in Moderation:

According to the "Dietary Guidelines for Americans 2020-2025," U.S. Department of Health and Human Services and U.S. Department of Agriculture, adults of legal drinking age can choose not to drink or to drink in moderation by limiting intake to 2 drinks or less in a day for men and 1 drink or less in a day for women, when alcohol is consumed. Drinking less is better for health than drinking more.

Binge Drinking:

- NIAAA defines binge drinking as a pattern of drinking alcohol that brings blood alcohol concentration (BAC) to 0.08 percent - or 0.08 grams of alcohol per deciliter - or higher. For a typical adult, this pattern corresponds to consuming 5 or more drinks (male), or 4 or more drinks (female), in about 2 hours.

- The Substance Abuse and Mental Health Services Administration (SAMHSA), which conducts the annual National Survey on Drug Use and Health (NSDUH), defines binge drinking as 5 or more alcoholic drinks for males or 4 or more alcoholic drinks for females on the same occasion (i.e., at the same time or within a couple of hours of each other) on at least 1 day in the past month.

Heavy Alcohol Use:

- NIAAA defines heavy drinking as follows

    - For men, consuming more than 4 drinks on any day or more than 14 drinks per week
    - For women, consuming more than 3 drinks on any day or more than 7 drinks per week.

- SAMHSA defines heavy alcohol use as binge drinking on 5 or more days in the past month

- 0-2 drinks per week
- 3-4 drinks per week
- 5-6 drinks per week
- 7-8 drinks per week
- 9-10 drinks per week
- 11-12 drinks per week
- 13-14 drinks per week
- >14 drinks per week

Dust_Allergy:

Dust mite allergy is an allergic reaction to tiny bugs that commonly live in house dust. Signs of dust mite allergy include those common to hay fever, such as sneezing and runny nose. Many people with dust mite allergy also experience signs of asthma, such as wheezing and difficulty breathing.

Dust mites, close relatives of ticks and spiders, are too small to see without a microscope. Dust mites eat skin cells shed by people, and they thrive in warm, humid environments. In most homes, such items as bedding, upholstered furniture and carpeting provide an ideal environment for dust mites.

By taking steps to reduce the number of dust mites in your home, you may get control of dust mite allergy. Medications or other treatments are sometimes necessary to relieve symptoms and manage asthma.

- Sneezing
- Runny nose
- Itchy, red or watery eyes
- Nasal congestion

- Itchy nose, roof of mouth or throat
- Cough
- Facial pressure and pain
- Swollen, blue-colored skin under your eyes

OccuPational_Hazards:

The definition of occupational hazard is the presence of hazardous materials or noises within a workplace that may affect people while performing their job. Such workplace pollutants may affect workers' health, especially if exposure continues over longer periods of time even at low levels. The most common exposure is that to workplace air pollution. This involves workplace hazards from airborne pollution, or, in other words, the presence in the workplace indoor air of hazardous substances either as gases (fumes) or as particulate matter (tiny particles - dust) dispersed in the air. Other types of exposure may occur involving skin contact, ingestion, and/or injection.

- It Jobs / Office Jobs
- Research Jobs
- Textile Jobs / Convenience Store Jobs
- Gas Station Jobs
- Dry Cleaning Jobs
- Manufacturing and Aerospace Jobs
- Chemical Industry Jobs
- Mining Jobs/Construction Jobs/Smelting Jobs

Genetic_Risk:

Genetic risk is the contribution our genes play in the chance we have of developing certain illnesses or diseases. Genes are not the only deciding factor for whether or not we will develop certain diseases and their influence varies depending on the disease.

- no
- yes

Chronic_Lung_Disease:

Chronic obstructive pulmonary disease (COPD) is a chronic inflammatory lung disease that causes obstructed airflow from the lungs. Symptoms include breathing difficulty, cough, mucus (sputum) production and wheezing.

- Stage-I-Early
- Stage-II-Moderate
- Stage-III-Severe
- Stage-IV-Very Severe

Balanced_Diet:

A balanced diet is a diet that contains differing kinds of foods in certain quantities and proportions so that the requirement for calories, proteins, minerals, vitamins and alternative nutrients is adequate and a small provision is reserved for additional nutrients to endure the short length of leanness.

- Following regularly
- Not following regularly

Obesity:

It is a measure that relates body weight to height. BMI is sometimes used to measure total body fat and whether a person is a healthy weight. Excess body fat is linked to an increased risk of some diseases including heart disease and some cancers. Also called body mass index.

- BMI overweight
- BMI not Overweight

Smoking:

It’s no big secret that smoking is bad for your health. Various campaigns, writings and media have made it abundantly clear that there are no health benefits to lighting up a roll of tobacco, though there are many consequences.

Types of smokers:
Social smokers
Anxious smokers
Skinny smokers
Addicted smokers

- No of cigarrates consumed per day-1-3
- No of cigarrates consumed per day-4-6
- No of cigarrates consumed per day-7-9
- No of cigarrates consumed per day-10-12
- No of cigarrates consumed per day-13-15
- No of cigarrates consumed per day-15-18
- No of cigarrates consumed per day-19-20
- No of cigarrates consumed per day- >20

Passive_Smoker:
Passive smoking is when you breathe in the smoke from other people's cigarettes, cigars or pipes. It is a serious health threat — being exposed to tobacco smoke for just a moment can cause harm. Unborn babies, children and people with breathing problems are most at risk.

- No effect
- Coughing
- Headaches
- Sore Throat
- Sore Eye
- Respiratory infections
- Nasal Irritation
- Asthama

Chest_Pain:
A sudden feeling of pressure, squeezing, tightness, or crushing under your breastbone. Chest pain that spreads to your jaw, left arm, or back. Sudden, sharp chest pain with shortness of breath, especially after a long period of inactivity.

- Non Urgent
- Standard
- Non Cardiac Pain
- Urgent(Atypical Chest Pain)
- Very Urgent
- Intense pain
- Immidiate
- Cardiac Pain
- Alteration Level Of consciousness

Coughing_Of_Blood:
Coughing up blood (also called hemoptysis) refers to coughing or spitting up of blood or bloody mucus from your respiratory tract (lungs and throat). It's a common condition that can have many causes. Some of these conditions can be very serious.

- per month-1-2 times
- per month-3-4 times
- per week-1-2 times
- per week-3-4 times
- per week-5-7 times
- per week-7-10 times
- per day-3-4 times
- per day-5-7 times
- per day->7 times

Fatigue:
Fatigue is a term used to describe an overall feeling of tiredness or lack of energy. It isn't the same as simply feeling drowsy or sleepy. When you're fatigued, you have no motivation and no energy. Being sleepy may be a symptom of fatigue, but it's not the same thing.

- Fully Alert
- Less Alert(Okay)
- Dehydrated+Drowsy
- Tired
- Streesed+Struggling
- Normal Emergency
- Extremely Urgent

Weight_Loss:

- 1-2 kg per month
- 3-4 kg per month
- 5-6 kg per month
- 7-8 kg per month
- 9-10 kg per month
- 10-12 kg per month
- 13-15 kg per month
- >15 kg per month

Shortness_Of_Breath:
Shortness of breath (known medically as dyspnea) is often described as an intense tightening in the chest, air hunger, difficulty breathing, breathlessness or a feeling of suffocation. Very strenuous exercise, extreme temperatures, obesity and higher altitude all can cause shortness of breath in a healthy person.

- Feeling Of suffocation
- Difficulty in breathing
- Air Hunger
- Intense Tightening Of the Chest
- Strenuous Exercise & High Temperatures

Wheezing:
Wheezing is the shrill whistle or coarse rattle you hear when your airway is partially blocked. It might be blocked because of an allergic reaction, a cold, bronchitis or allergies. Wheezing is also a symptom of asthma, pneumonia, heart failure and more.

- mild intermittent
- moderate intermittent
- severe intermittent
- persistant mild
- persistant moderate
- persistant severe

Swallowing_Difficulty:
Dysphagia is difficulty swallowing — taking more time and effort to move food or liquid from your mouth to your stomach. Dysphagia can be painful. In some cases, swallowing is impossible.

- Yes
- No

Clubbing_Of_Finger_Nails:
Clubbed fingers is a symptom of disease, often of the heart or lungs which cause chronically low blood levels of oxygen. Diseases which cause malabsorption, such as cystic fibrosis or celiac disease can also cause clubbing. Clubbing may result from chronic low blood-oxygen levels.

- No Visible Clubbing
- Mild Clubbing
- Moderate Clubbing
- Gross Clubbing

Frequent_Cold:
Frequent colds are also common if your immune system is compromised.

- Rare
- Common
- Mild
- Moderate
- Severe

Dry_Cough:
A cough is your body's way of responding when something irritates your throat or airways. An irritant stimulates nerves that send a message to your brain. The brain then tells muscles in your chest and abdomen to push air out of your lungs to force out the irritant.

- Drycough
- Wetcough

Snoring:
Snoring occurs when air flows past relaxed tissues, such as your tongue, soft palate and airway, as you breathe. The sagging tissues narrow your airway, causing these tissues to vibrate.

- No Snoring
- Rarely slight snoring Depends on the sleep Position
- Continuous slight snoring without hyponea
- Sometimes loud snoring without hyponea
- Continuous loud snoring without hyponea or apnea
- Continuous loud snoring with hyponea and apnea
- Snoring with sore throat and pain

Label :

- ***Low***
- ***Medium***
- ***High***

2. BREAST CANCER

Diagnosis	Mean_Radius	Mean_Texture	Mean_Perimeter	Mean_Area	Mean_Smoothness	Mean_Compactness	Mean_Concavity	Mean_Concave_Points	Mean_Symmetry	Mean_Fractal_Dimension	Radius_Error
M	17.99	10.38	122.8	1001	0.1184	0.2776	0.3001	0.1471	0.2419	0.07871	1.095
M	20.57	17.77	132.9	1326	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	0.5435
M	19.69	21.25	130	1203	0.1096	0.1599	0.1974	0.1279	0.2069	0.05999	0.7456
M	11.42	20.38	77.58	386.1	0.1425	0.2839	0.2414	0.1052	0.2597	0.09744	0.4956
M	20.29	14.34	135.1	1297	0.1003	0.1328	0.198	0.1043	0.1809	0.05883	0.7572
M	12.45	15.7	82.57	477.1	0.1278	0.17	0.1578	0.08089	0.2087	0.07613	0.3345
M	18.25	19.98	119.6	1040	0.09463	0.109	0.1127	0.074	0.1794	0.05742	0.4467
M	13.71	20.83	90.2	577.9	0.1189	0.1645	0.09366	0.05985	0.2196	0.07451	0.5835
M	13	21.82	87.5	519.8	0.1273	0.1932	0.1859	0.09353	0.235	0.07389	0.3063
M	12.46	24.04	83.97	475.9	0.1186	0.2396	0.2273	0.08543	0.203	0.08243	0.2976
M	16.02	23.24	102.7	797.8	0.08206	0.06669	0.03299	0.03323	0.1528	0.05697	0.3795
M	15.78	17.89	103.6	781	0.0971	0.1292	0.09954	0.06606	0.1842	0.06082	0.5058
M	19.17	24.8	132.4	1123	0.0974	0.2458	0.2065	0.1118	0.2397	0.078	0.9555

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.

Attributes:
Creation of this dataset included using 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.

- ✓ Mean_Radius
- ✓ Mean_Texture
- ✓ Mean_Perimeter
- ✓ Mean_Area
- ✓ Mean_Smoothness
- ✓ Mean_Compactness
- ✓ Mean_Concavity
- ✓ Mean_Concave_Points
- ✓ Mean_Symmetry
- ✓ Mean_Fractal_Dimension
- ✓ Radius_Error
- ✓ Texture_Error
- ✓ Perimeter_Error
- ✓ Area_Error

- ✓ Smoothness_Error
- ✓ Compactness_Error
- ✓ Concavity_Error
- ✓ Concave_Points_Error
- ✓ Symmetry_Error
- ✓ Fractal_Dimension_Error
- ✓ Worst_Radius
- ✓ Worst_Texture
- ✓ Worst_Perimeter
- ✓ Worst_Area
- ✓ Worst_Smoothness
- ✓ Worst_Compactness
- ✓ Worst_Concavity
- ✓ Worst_Concave_Points
- ✓ Worst_Symmetry
- ✓ Worst_Fractal_Dimension
- ✓ **Diagnosis**
  - ***Malignant***
  - ***Benign***

DATA PREPROCESSING:

1. Lung cancer

Gender

- 1 for male
- 2 for female

air pollution-

- 0-50 = 1
- 51-100 = 2
- 101-150 = 3
- 151-200 = 4
- 201-250 = 5
- 251-300 = 6
- 301-350 = 7
- >350 = 8

alcohol-use

- 0-2 drink per week = 1
- 3-4 drink per week = 2
- 5-6 drink per week = 3
- 7-8 drink per week = 4
- 9-10 drink per week = 5
- 11-12 drink per week = 6
- 13-14 drinks per week = 7
- >14 drinks per week = 8

dust_allergy

- Sneezing = 1
- Runny nose = 2
- Itchy, red or watery eyes = 3
- Nasal congestion = 4
- Itchy nose, roof of mouth or throat = 5
- Cough = 6

- Facial pressure and pain = 7
- Swollen, blue-colored skin under your eyes =8

Occupational_hazard
- It Jobs / Office Jobs = 1
- Research Jobs = 2
- Textile Jobs / Convenience Store Jobs = 3
- Gas Station Jobs = 4
- Dry Cleaning Jobs = 5
- Manufacturing and Aerospace Jobs = 6
- Chemical Industry Jobs = 7
- Mining Jobs/Construction Jobs/Smelting Jobs = 8

Genetic_Risk
- no=1
- yes=2

Chronic_Lung_Disease
- Stage-I-Early = 1
- Stage-II-Moderate = 2
- Stage-III-Severe = 3
- Stage-IV-Very Severe = 4

Balanced_Diet
- Following reguraly-yes-2
- Not following reguraly-no-1

Obesity
- BMI overweight-2
- BMI not Overweight-1

Smoking
- No of cigarrates consumed per day-1-3 = 1
- No of cigarrates consumed per day-4-6 = 2
- No of cigarrates consumed per day-7-9 = 3
- No of cigarrates consumed per day-10-12 = 4
- No of cigarrates consumed per day-13-15 = 5
- No of cigarrates consumed per day-15-18 = 6
- No of cigarrates consumed per day-19-20 = 7
- No of cigarrates consumed per day- >20 = 8

Passive_Smoker
- No effect-1
- Coughing-2
- Headaches-3
- Sore Throat-4
- Sore Eye-5
- Respiratory infections-6
- Nasal Irritation-7
- Asthama-8

Chest_Pain

- Non Urgent-1
- Standard-2
- Non Cardiac Pain-3
- Urgent(Atypical Chest Pain)-4
- Very Urgent-5
- Intense pain-6
- Immidiate-7
- Cardiac Pain-8
- Alteration Level Of consciousness-9

Coughing_Of_Blood

- per month-1-2 times-1
- per month-3-4 times-2
- per week-1-2 times-3
- per week-3-4 times-4
- per week-5-7 times-5
- per week-7-10 times-6
- per day-3-4 times-7
- per day-5-7 times-8
- per day->7 times-9

Fatigue

- Fully Alert-1
- Less Alert(Okay)-2
- Dehydrated+Drowsy-3
- Tired-4
- Streesed+Struggling-5
- Normal Emergency-6
- Extremely Urgent-7

Weight_Loss

- 1-2 kg per month = 1
- 3-4 kg per month = 2
- 5-6 kg per month = 3
- 7-8 kg per month = 4
- 9-10 kg per month = 5
- 10-12 kg per month = 6
- 13-15 kg per month = 7
- >15 kg per month = 8

Shortness_Of_Breath

- Feeling Of suffocation = 1
- Difficulty in breathing = 2
- Air Hunger = 3
- Intense Tightening Of the Chest = 4
- Strenous Exercise & High Temperatures = 5

WheezingAsthama

- mild intermittent-1
- moderate intermittent-2
- severe intermittent-3
- persistant mild-4
- persistant moderate-5
- persistant severe-6

Swallowing_Difficulty

- No = 1
- Yes = 2

Clubbing_Of_Finger_Nails

- No Visible Clubbing-1
- Mild Clubbing-2
- Moderate Clubbing-3
- Gross Clubbing-4

Frequent_Cold- max_value

- Rare-1
- Common-2
- Mild-3
- Moderate-4
- Severe-5

Dry_Cough

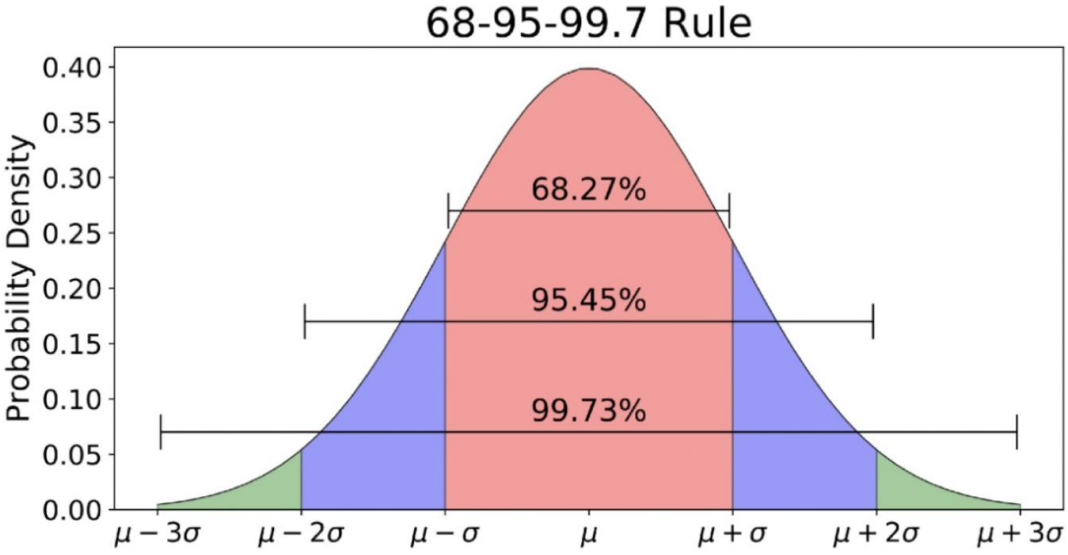
- Drycough-1
- Wetcough-2

Snoring

- No Snoring-1
- Rarely slight snoring Depends on the sleep Position-2
- Continuous slight snoring without hyponea-3
- Sometimes loud snoring without hyponea-4
- Continuous loud snoring without hyponea or apnea-5
- Continuous loud snoring with hyponea and apnea-6
- Snoring with sore throat and pain-7

2. Breast cancer

The values have to be standardized so that the data will be in Normal Distribution for better working of the algorithms used. The data in Breast cancer will be in continuous form. As these values are recorded in LAB REPORT.

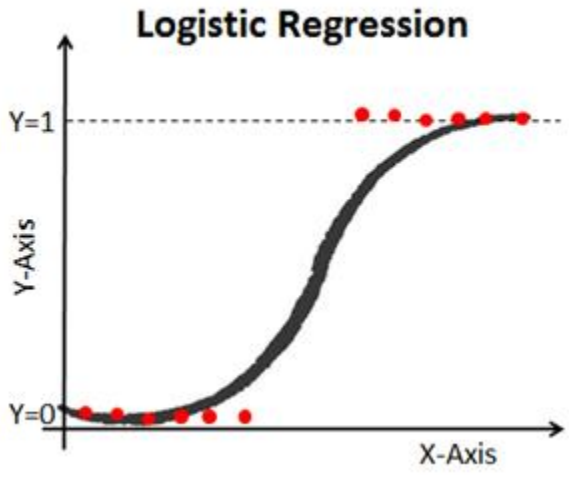


Algorithms techniques we Tested for BEST ACCURACY:

Logistic Regression?

Logistic regression is a supervised learning classification algorithm used to predict the probability of a target variable. The nature of target or dependent variable is dichotomous, which means there would be only two possible classes.

In simple words, the dependent variable is binary in nature having data coded as either 1 (stands for success/yes) or 0 (stands for failure/no)



Types of Logistic Regression:

Generally, logistic regression means binary logistic regression having binary target variables, but there can be two more categories of target variables that can be predicted by it. Based on those number of categories, Logistic regression can be divided into following types:

- ✓ ***Binary or Binomial:*** A dependent variable will have only two possible types either 1 and 0.

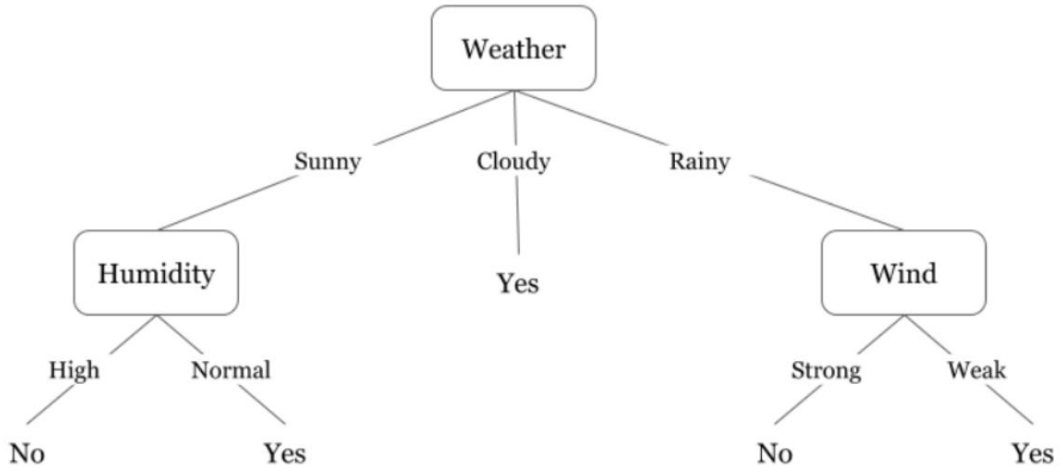
- ✓ ***Multinomial:*** Dependent variable can have 3 or more possible unordered types or the types having no quantitative significance.
- ✓ ***Ordinal:*** Dependent variable can have 3 or more possible ordered types or the types having a quantitative significance.

Decision Tree?

It is a tool that has applications spanning several different areas. Decision trees can be used for classification as well as regression problems. The name itself suggests that it uses a flowchart like a tree structure to show the predictions that result from a series of feature-based splits. It starts with a root node and ends with a decision made by leaves.

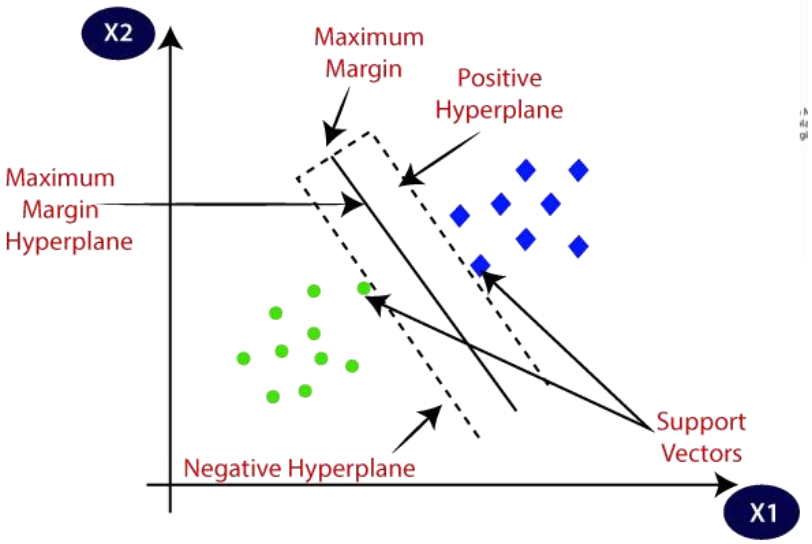
Day	Weather	Temperature	Humidity	Wind	Play?
1	Sunny	Hot	High	Weak	No
2	Cloudy	Hot	High	Weak	Yes
3	Sunny	Mild	Normal	Strong	Yes
4	Cloudy	Mild	High	Strong	Yes
5	Rainy	Mild	High	Strong	No
6	Rainy	Cool	Normal	Strong	No
7	Rainy	Mild	High	Weak	Yes
8	Sunny	Hot	High	Strong	No
9	Cloudy	Hot	Normal	Weak	Yes
10	Rainy	Mild	High	Strong	No

Decision tree for the above dataset :
In the below diagram the tree will first ask what is the weather? Is it sunny, cloudy, or rainy? If yes then it will go to the next feature which is humidity and wind. It will again check if there is a strong wind or weak, if it's a weak wind and it's rainy then the person may go and play.



SVM?

In machine learning, support-vector machines (SVMs, also support-vector networks) are supervised learning models with associated learning algorithms that analyze data for classification and regression analysis.
SVMs are one of the most robust prediction methods, being based on statistical learning frameworks or VC theory proposed by Vapnik (1982, 1995) and Chervonenkis (1974). Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier (although methods such as Platt scaling exist to use SVM in a probabilistic classification setting). SVM maps training examples to points in space so as to maximise the width of the gap between the two categories. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.

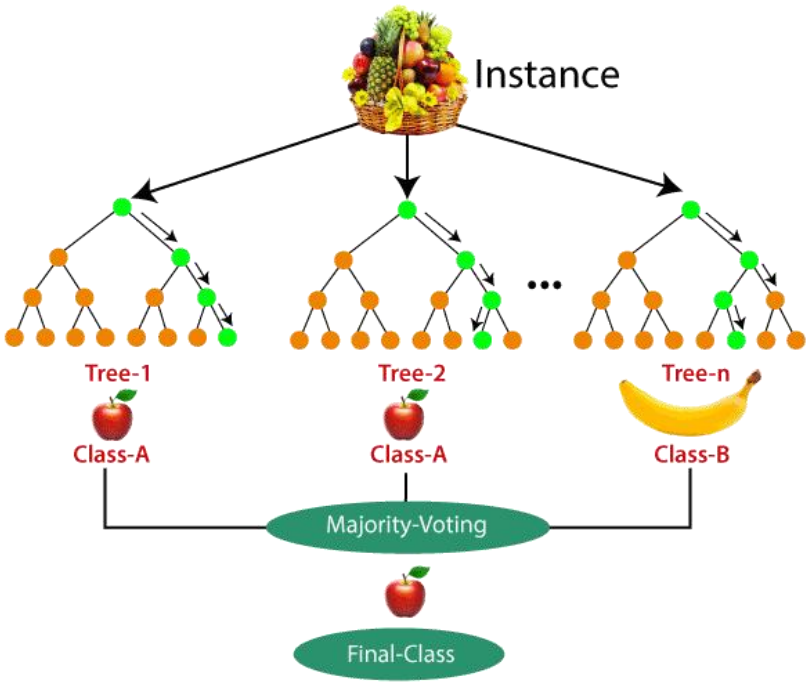


In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces. When data are unlabelled, supervised learning is not possible, and an unsupervised learning approach is required, which attempts to find natural clustering of the data to groups, and then map new data to these formed groups. The support-vector clustering[3] algorithm, created by Hava Siegelmann and Vladimir Vapnik, applies the statistics of support vectors, developed in the support vector machines algorithm, to categorize unlabeled data.

RANDOM FOREST?

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is controlled with the max_samples parameter if bootstrap=True (default), otherwise the whole dataset is used to build each tree.
How random forest works?:

- Select random samples from a given dataset.
- Construct a decision tree for each sample and get a prediction result from each decision tree.
- Perform a vote for each predicted result.
- Select the prediction result with the most votes as the final prediction.



Advantages of random forests

- Works well "out of the box" without tuning any parameters. Other models may have settings that require significant experimentation to find the best values.
- Tend not to overfit. The processes of randomizing the data and variables across many trees means that no single tree sees all the data. This helps to focus on the general patterns within the training data and reduce sensitivity to noise.
- Ability to handle non-linear numeric and categorical predictors and outcomes. Other models may require numeric inputs or assume linearity.
- Accuracy calculated from out-of-bag samples is a proxy for using a separate test data set. The out-of-bag samples are those not used for training a specific tree and as such can be used as an unbiased measure of performance.
- Predictor variable importance can be calculated. For more information, see "How is Variable Importance Calculated for Random Forests?"

PERFORMANCE ANALYSIS

We tested the accuracy for “Cancer prediction” for lung and breast cancer using many existing ML Algorithms (Decision trees, Logistic Regression, Support Vector Machine and Random Forest.)
For Lung cancer prediction, as all the attributes are categorical variables so we used Decision Trees, Logistic regression and Random forest.
For Breast cancer prediction, as the attributes are continuous and we cannot use Logistic regression, we used Decision Trees, Support Vector Machines (SVM) and Random Forest as the algorithms to be tested for the best one.
As RANDOM FOREST gives the **best accuracy** for both the given datasets. We chose that to be the suitable algorithm for our project.

Accuracy prediction with each algorithm:

Lung Cancer Prediction

Decision Trees

In [43]:
```
# Load libraries
import pandas as pd
from sklearn.tree import DecisionTreeClassifier # Import Decision Tree Classifier
from sklearn.model_selection import train_test_split # Import train_test_split function
from sklearn import metrics #Import scikit-learn metrics module for accuracy calculation
```

In [46]:
```
col_names = ['Age', 'Gender', 'Air_pollution', 'Alcohol_Use', 'Dust_Allergy', 'OccuPational_Hazards', 'Genetic_Risk', 'Chronic_Lu
# Load dataset
pima = pd.read_csv(r'/content/lungcancer.csv', header=None, names=col_names)
```

In [47]:
```
pima.head()
```

Out[47]:

	Age	Gender	Air_pollution	Alcohol_Use	Dust_Allergy	OccuPational_Hazards	Genetic_Risk	Chronic_Lung_Disease	Balanced_Diet	Obesity	...	Fatigue	We
0	Age	Gender	Air_Pollution	Alcohol_Use	Dust_Allergy	OccuPational_Hazards	Genetic_Risk	Chronic_Lung_Disease	Balanced_Diet	Obesity	...	Fatigue	W
1	33	1	2	4	5	4	3	1	2	4	...	3	
2	17	1	3	1	5	3	4	1	2	2	...	1	
3	35	1	4	5	6	5	5	2	6	7	...	8	
4	37	1	7	7	7	7	6	4	7	7	...	4	

5 rows × 24 columns

In [48]:
```
#split dataset in features and target variable
feature_cols = ['Age', 'Gender', 'Air_pollution', 'Alcohol_Use', 'Dust_Allergy', 'OccuPational_Hazards', 'Genetic_Risk', 'Chronic
X = pima[feature_cols] # Features
y = pima.label # Target variable
```

In [49]:
```
# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X[1:], y[1:], test_size=0.3, random_state=1) # 70% training and 30% test
```

DECISION TREE

In [50]:
```
# Create Decision Tree classifer object
clf = DecisionTreeClassifier()
```

In [51]:
```
# Train Decision Tree Classifer
clf = clf.fit(X_train,y_train)
```

In [52]:
```
#Predict the response for test dataset
y_pred = clf.predict(X_test)
```

In [1]:
```
# Model Accuracy, how often is the classifier correct?
print("Accuracy: ",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy:  0.91333333334
```

In [54]:
```
pip install graphviz
```

```
Requirement already satisfied: graphviz in /usr/local/lib/python3.7/dist-packages (0.10.1)
```

In [55]:
```
pip install pydotplus
```

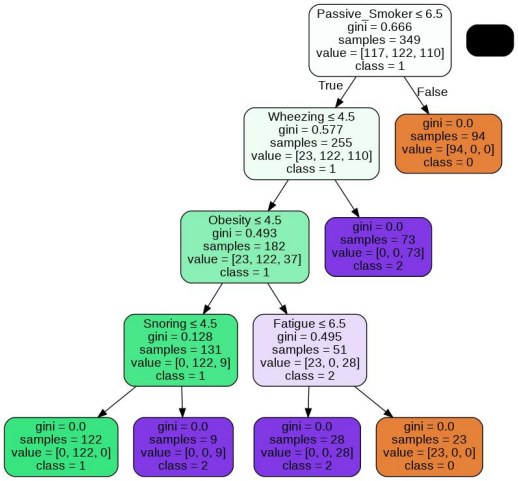
```
Requirement already satisfied: pydotplus in /usr/local/lib/python3.7/dist-packages (2.0.2)
Requirement already satisfied: pyparsing>=2.0.1 in /usr/local/lib/python3.7/dist-packages (from pydotplus) (3.0.8)
```

In [56]:
```
pip install --upgrade scikit-learn==0.20.3
```

```
Requirement already satisfied: scikit-learn==0.20.3 in /usr/local/lib/python3.7/dist-packages (0.20.3)
Requirement already satisfied: numpy>=1.8.2 in /usr/local/lib/python3.7/dist-packages (from scikit-learn==0.20.3) (1.21.6)
Requirement already satisfied: scipy>=0.13.3 in /usr/local/lib/python3.7/dist-packages (from scikit-learn==0.20.3) (1.4.1)
```

In [57]:
```
from sklearn.tree import export_graphviz
from sklearn.externals.six import StringIO
from IPython.display import Image
import pydotplus
```

In [58]:
```
dot_data = StringIO()
export_graphviz(clf, out_file=dot_data,
                filled=True, rounded=True,
                special_characters=True,feature_names = feature_cols,class_names=['0','1','2'])
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('lungcancer.png')
Image(graph.create_png())
```



In [2]:
```
# Model Accuracy, how often is the classifier correct?
print("Accuracy: ",metrics.accuracy_score(y_test, y_pred))
```

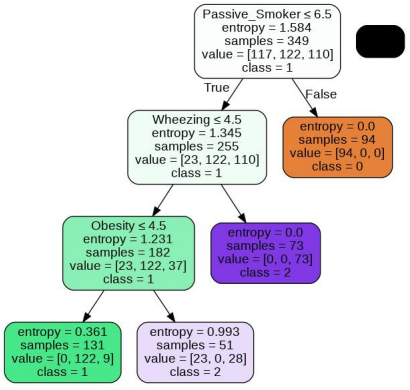
```
Accuracy:  0.9090909090901
```

In [62]:
```
# Create Decision Tree classifer object
clf = DecisionTreeClassifier(criterion="entropy", max_depth=3)

# Train Decision Tree Classifer
clf = clf.fit(X_train,y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)

dot_data = StringIO()
export_graphviz(clf, out_file=dot_data,
                filled=True, rounded=True,
                special_characters=True,feature_names = feature_cols,class_names=['0','1','2'])
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('lungcancer.png')
Image(graph.create_png())
```



In [3]:
```
# Model Accuracy, how often is the classifier correct?
print("Accuracy: ",metrics.accuracy_score(y_test, y_pred))
```

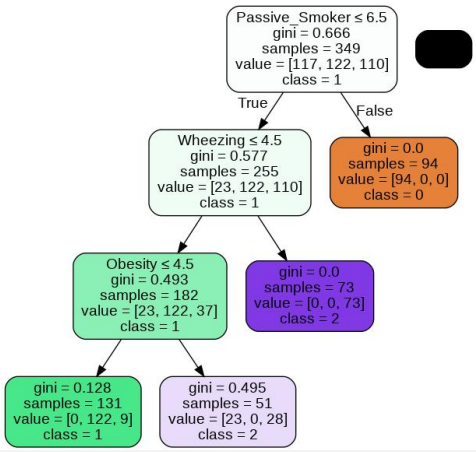
```
Accuracy:  0.916666666667
```

In [64]:
```
# Create Decision Tree classifer object
clf = DecisionTreeClassifier(criterion="gini", max_depth=3)

# Train Decision Tree Classifer
clf = clf.fit(X_train,y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)

dot_data = StringIO()
export_graphviz(clf, out_file=dot_data,
                filled=True, rounded=True,
                special_characters=True,feature_names = feature_cols,class_names=['0','1','2'])
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('lungcancer.png')
Image(graph.create_png())
```



In [4]:
```
# Model Accuracy, how often is the classifier correct?
print("Accuracy: ",metrics.accuracy_score(y_test, y_pred))
```

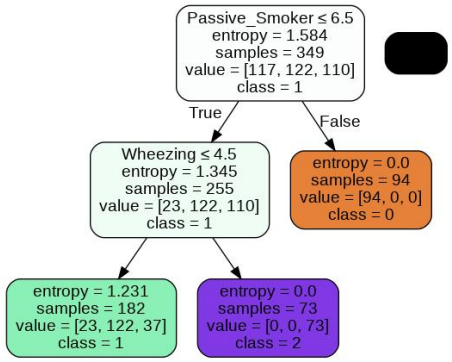
```
Accuracy:  0.916666666667
```

In [66]:
```
# Create Decision Tree classifer object
clf = DecisionTreeClassifier(criterion="entropy", max_depth=2)

# Train Decision Tree Classifer
clf = clf.fit(X_train,y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)

dot_data = StringIO()
export_graphviz(clf, out_file=dot_data,
                filled=True, rounded=True,
                special_characters=True,feature_names = feature_cols,class_names=['0','1','2'])
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('lungcancer.png')
Image(graph.create_png())
```



In [5]:
```
# Model Accuracy, how often is the classifier correct?
print("Accuracy: ",metrics.accuracy_score(y_test, y_pred))
```

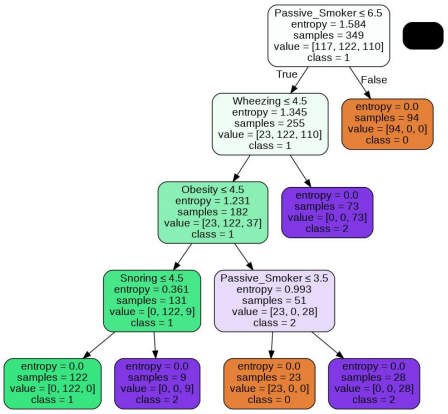
```
Accuracy:  0.91333333334
```

In [69]:
```
# Create Decision Tree classifer object
clf = DecisionTreeClassifier(criterion="entropy", max_depth=4)

# Train Decision Tree Classifer
clf = clf.fit(X_train,y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)

dot_data = StringIO()
export_graphviz(clf, out_file=dot_data,
                filled=True, rounded=True,
                special_characters=True,feature_names = feature_cols,class_names=['0','1','2'])
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('lungcancer.png')
Image(graph.create_png())
```



In [6]:
```
# Model Accuracy, how often is the classifier correct?
print("Accuracy: ",metrics.accuracy_score(y_test, y_pred))
```

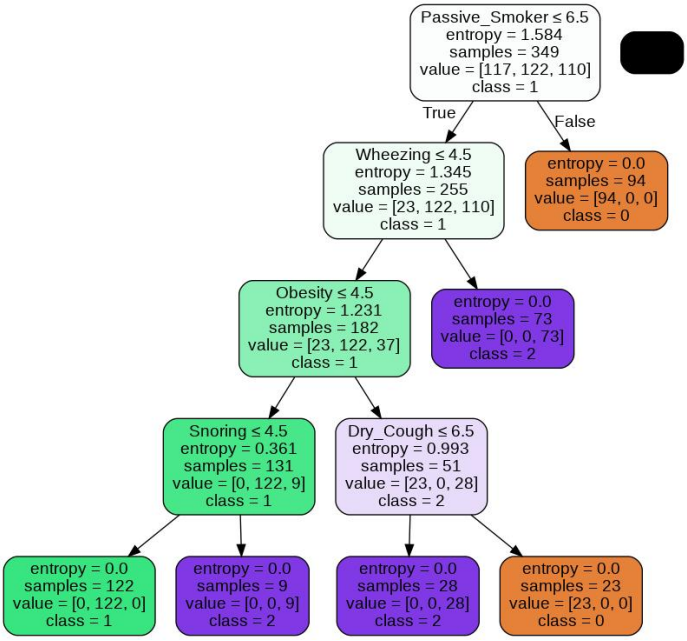
```
Accuracy: 0.916666666667
```

In [72]:
```
# Create Decision Tree classifer object
clf = DecisionTreeClassifier(criterion="entropy", max_depth=8)

# Train Decision Tree Classifer
clf = clf.fit(X_train,y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)

dot_data = StringIO()
export_graphviz(clf, out_file=dot_data,
                filled=True, rounded=True,
                special_characters=True,feature_names = feature_cols,class_names=['0','1','2'])
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('lungcancer.png')
Image(graph.create_png())
```



In [7]:
```
# Model Accuracy, how often is the classifier correct?
print("Accuracy: ",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.91333333334
```

Logistic regression

In [1]:
```
#import pandas
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
%matplotlib inline

import warnings
warnings.filterwarnings('ignore')
```

In [4]:
```
data = pd.read_csv('lungcancer.csv')
data['Level'].replace(['Low', 'Medium', 'High'], [1, 2, 3], inplace=True)
```

In [5]:
```
#Training and Testing Dataset
from sklearn import linear_model
reg = linear_model.LogisticRegression()
data2 = data.values
X = (data2[:,0:22])
y = (data2[:,23])

print(X)
print(y.shape)
```

```
[[33  1  2 ...  1  2  3]
 [17  1  3 ...  2  1  7]
 [35  1  4 ...  4  6  7]
 ...
 [24  1  3 ...  2  1  1]
 [35  1  2 ...  2  1  2]
 [38  2  5 ...  3  2  4]]
(499,)
```

In [6]:
```
#Minmax Normalisation
from sklearn.preprocessing import MinMaxScaler

min = MinMaxScaler()
X = min.fit_transform(X)
print(X)
```

```
[[0.3220339  0.         0.14285714 ... 0.         0.16666667 0.33333333]
 [0.05084746 0.         0.28571429 ... 0.125      0.         1.        ]
 [0.3559322  0.         0.42857143 ... 0.375      0.83333333 1.        ]
 ...
 [0.16949153 0.         0.28571429 ... 0.125      0.         0.        ]
 [0.3559322  0.         0.14285714 ... 0.125      0.         0.16666667]
 [0.40677966 1.         0.57142857 ... 0.25       0.16666667 0.5       ]]
```

In [7]:
```
#Regression Line fitting
from sklearn import linear_model
reg = linear_model.LinearRegression()
reg.fit(X, y)

print('Coefficients', reg.coef_)
print('Intercept', reg.intercept_)
```

```
Coefficients [ 0.02330518  0.03481316  0.11851015  0.41859646 -0.05443973 -0.44506581
  0.78465867 -0.17095841 -0.00546746  0.20730068 -0.16720894  0.43259462
 -0.23881197  0.84706821  0.802145   -0.2343941   0.12400873 -0.02467707
  0.53506793  0.44961668  0.13668705  0.21544842]
Intercept 0.3395820926507498
```

In [8]:
```
#Prediction
y_predict = reg.predict(X)
for i in range (0, len(X)):
    print(y[i], y_predict[i])

data['Pred'] = y_predict
data
```

```
1 1.3472123617044043
2 1.6409531526255043
3 2.884273758169396
3 2.937001978136002
3 2.774709603928428
3 2.884273758169396
1 1.3895305859993567
1 0.9870473840977168
2 2.0856256851514035
2 1.6248000168960965
3 3.1164140129277125
3 3.0534089509321936
2 2.2937278899859357
3 2.91888694718706
1 1.2238895247282855
2 2.0376235097249293
2 1.6409531526255043
3 2.91888694718706
```

In [22]:
```
#CASE-1:
clf = LogisticRegression(tol=1e-4, random_state=0, max_iter=1000)
clf.fit(X,y)
pred=clf.predict(X[:1, :])
print(pred)
print('Accuracy : ',clf.score(X, y))
```

```
[1]
Accuracy :  0.78666666667
```

In [23]:
```
#CASE -2:
clf =LogisticRegression(tol=1e-4, random_state=89, max_iter=1000)
clf.fit(X,y)
pred=clf.predict(X[:2, :])
print(pred)
print('Accuracy : ',clf.score(X, y))
```

```
[1 2]
Accuracy :  0.78666666667
```

In [24]:
```
#CASE -3:
clf =LogisticRegression(tol=1e-4, random_state=35, max_iter=1000)
clf.fit(X,y)
pred=clf.predict(X[:2, :])
print(pred)
print('Accuracy : ',clf.score(X, y))
```

```
[1 2]
Accuracy :  0.78666666667
```

In [25]:
```
#CASE -4:
clf =LogisticRegression(penalty='none',tol=1e-4, random_state=92, max_iter=1000)
clf.fit(X,y)
pred=clf.predict(X[:2, :])
print(pred)
print('Accuracy : ',clf.score(X, y))
```

```
[1 2]
Accuracy :  0.79333333334
```

In [26]:
```
#CASE -5:
clf =LogisticRegression(penalty='l2',tol=1e-4, random_state=92, max_iter=1000)
clf.fit(X,y)
pred=clf.predict(X[:2, :])
print(pred)
print('Accuracy : ',clf.score(X, y))
```

```
[1 2]
Accuracy :  0.75649846549
```

In [27]:
```
#CASE -6:
clf =LogisticRegression(solver='newton-cg',penalty='l2',tol=1e-4, random_state=92, max_iter=1000)
clf.fit(X,y)
pred=clf.predict(X[:2, :])
print(pred)
print('Accuracy : ',clf.score(X, y))
```

```
[1 2]
Accuracy :  0.78666666667
```

In [28]:
```
#CASE -6:
clf =LogisticRegression(solver='newton-cg',penalty='none',tol=1e-4, random_state=92, max_iter=1000)
clf.fit(X,y)
pred=clf.predict(X[:2, :])
print(pred)
print('Accuracy : ',clf.score(X, y))
```

```
[1 2]
Accuracy :  0.79333333334
```

In [29]:
```
#CASE -7:
clf =LogisticRegression(solver='liblinear',penalty='l1',tol=1e-4, random_state=92, max_iter=1000)
clf.fit(X,y)
pred=clf.predict(X[:2, :])
print(pred)
print('Accuracy : ',clf.score(X, y))
```

```
[1 2]
Accuracy :  0.75649846549
```

In [30]:
```
#CASE -9:
clf =LogisticRegression(solver='saga',penalty='l1',tol=1e-4, random_state=92, max_iter=1000)
clf.fit(X,y)
pred=clf.predict(X[:2, :])
print(pred)
print('Accuracy : ',clf.score(X, y))
```

```
[1 2]
Accuracy :  0.79333333334
```

In [31]:
```
#CASE -10:
clf =LogisticRegression(solver='saga',penalty='l2',tol=1e-4, random_state=92, max_iter=1000)
clf.fit(X,y)
pred=clf.predict(X[:2, :])
print(pred)
print('Accuracy : ',clf.score(X, y))
```

```
[1 2]
Accuracy :  0.78666666667
```

In [32]:
```
#CASE -11:
clf =LogisticRegression(solver='saga',penalty='none',tol=1e-4, random_state=92, max_iter=1000)
clf.fit(X,y)
pred=clf.predict(X[:2, :])
print(pred)
print('Accuracy : ',clf.score(X, y))
```

```
[1 2]
Accuracy :  0.79333333334
```

In [33]:
```
#CASE -12:
clf =LogisticRegression(solver='saga',penalty='l1',multi_class='auto',tol=1e-4, random_state=92, max_iter=1000)
clf.fit(X,y)
pred=clf.predict(X[:2, :])
print(pred)
print('Accuracy : ',clf.score(X, y))
```

```
[1 2]
Accuracy :  0.79333333334
```

In [34]:
```
#CASE -13:
clf =LogisticRegression(solver='saga',penalty='l2',multi_class='multinomial',tol=1e-4, random_state=92, max_iter=1000)
clf.fit(X,y)
pred=clf.predict(X[:2, :])
print(pred)
print('Accuracy : ',clf.score(X, y))
```

```
[1 2]
Accuracy :  0.78666666667
```

Random Forest

In [1]:
```
# importing required libraries
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn import model_selection
```

In [2]:
```
# read the train and test dataset
dat = pd.read_csv('lungcancer.csv')
```

In [3]:
```
# view the top 3 rows of the dataset
print(dat.head(9))
```

```
   Age  Gender  Air_Pollution  Alcohol_Use  Dust_Allergy  \
0   33       1              2            4             5
1   17       1              3            1             5
2   35       1              4            5             6
3   37       1              7            7             7
4   46       1              6            8             7
5   35       1              4            5             6
6   52       2              2            4             5
7   28       2              3            1             4
8   35       2              4            5             6

   OccuPational_Hazards  Genetic_Risk  Chronic_Lung_Disease  Balanced_Diet  \
0                     4             3                     2              2
1                     3             4                     2              2
2                     5             5                     4              6
3                     7             6                     7              7
4                     7             7                     6              7
5                     5             5                     4              6
6                     4             3                     2              2
7                     3             2                     3              4
8                     5             6                     5              5

   Obesity  ...  Fatigue  Weight_Loss  Shortness_Of_Breath  Wheezing  \
0        4  ...        3            4                    2         2
1        2  ...        1            3                    7         8
2        7  ...        8            7                    9         2
3        7  ...        4            2                    3         1
4        7  ...        3            2                    4         1
5        7  ...        8            7                    9         2
6        4  ...        3            4                    2         2
7        3  ...        3            2                    2         4
8        5  ...        1            4                    3         2

   Swallowing_Difficulty  Clubbing_Of_Finger_Nails  Frequent_Cold  Dry_Cough  \
0                      3                         1              2          3
1                      6                         2              1          7
2                      1                         4              6          7
3                      4                         5              6          7
4                      4                         2              4          2
5                      1                         4              6          7
6                      3                         1              2          3
7                      2                         2              3          4
8                      4                         6              2          4

   Snoring   Level
0        4     Low
1        2  Medium
2        2    High
3        5    High
4        3    High
5        2    High
6        4     Low
7        3     Low
8        1  Medium

[9 rows x 24 columns]
```

In [4]:
```
dat.describe().transpose()
```

Out[4]:

	count	mean	std	min	25%	50%	75%	max
Age	499.0	37.042084	12.167020	14.0	27.0	35.0	45.0	73.0
Gender	499.0	1.408818	0.492109	1.0	1.0	1.0	2.0	2.0
Air_Pollution	499.0	3.749499	1.993809	1.0	2.0	3.0	6.0	8.0
Alcohol_Use	499.0	4.428858	2.602217	1.0	2.0	4.0	7.0	8.0
Dust_Allergy	499.0	5.046092	2.013977	1.0	3.0	5.0	7.0	8.0
OccuPational_Hazards	499.0	4.747495	2.111445	1.0	3.0	5.0	7.0	8.0
Genetic_Risk	499.0	4.498998	2.117412	1.0	2.0	5.0	7.0	7.0
Chronic_Lung_Disease	499.0	4.322645	1.868710	1.0	3.0	4.0	6.0	7.0
Balanced_Diet	499.0	4.430862	2.123858	1.0	2.0	4.0	7.0	7.0
Obesity	499.0	4.348697	2.127147	1.0	3.0	4.0	7.0	7.0
Smoking	499.0	3.919840	2.465339	1.0	2.0	3.0	7.0	8.0
Passive_Smoker	499.0	4.150301	2.297077	1.0	2.0	4.0	7.0	8.0
Chest_Pain	499.0	4.376754	2.305559	1.0	2.0	4.0	7.0	9.0
Coughing_Of_Blood	499.0	4.741483	2.430418	1.0	3.0	4.0	7.0	9.0
Fatigue	499.0	3.733467	2.227769	1.0	2.0	3.0	5.0	9.0
Weight_Loss	499.0	3.757515	2.179523	1.0	2.0	3.0	6.0	8.0
Shortness_Of_Breath	499.0	4.144289	2.277713	1.0	2.0	4.0	6.0	9.0
Wheezing	499.0	3.693387	2.027961	1.0	2.0	4.0	5.0	8.0
Swallowing_Difficulty	499.0	3.709419	2.231963	1.0	2.0	4.0	5.0	8.0
Clubbing_Of_Finger_Nails	499.0	3.843687	2.361764	1.0	2.0	4.0	5.0	9.0
Frequent_Cold	499.0	3.432866	1.842439	1.0	2.0	3.0	4.0	7.0
Dry_Cough	499.0	3.821643	2.038350	1.0	2.0	4.0	6.0	7.0
Snoring	499.0	2.891784	1.506450	1.0	2.0	3.0	4.0	7.0

In [5]:
```
x1 = dat.drop('Level',axis=1).values
y1 = dat['Level'].values
```

In [6]:
```
train_x, test_x, train_y, test_y = model_selection.train_test_split(x1,y1,test_size=0.34,random_state=100)
print('\nShape of training data :',train_x.shape)
print('\nShape of testing data :',test_y.shape)
```

```
Shape of training data : (329, 23)

Shape of testing data : (170,)
```

In [7]:
```
model = RandomForestClassifier()
# fit the model with the training data
model.fit(train_x,train_y)
# predict the target on the train dataset
predict_train = model.predict(train_x)
# Accuray Score on train dataset
accuracy_train = accuracy_score(train_y,predict_train)
print('Accuracy on train data: ',accuracy_train)
# predict the target on the test dataset
predict_test = model.predict(test_x)
# Accuracy Score on test dataset
accuracy_test = accuracy_score(test_y,predict_test)
print('\nAccuracy on test data: ',accuracy_test)
```

```
Accuracy on train data: 1.0
Accuracy on test data: 0.92333333334
```

In [8]:
```
model = RandomForestClassifier(criterion = "entropy")
# fit the model with the training data
model.fit(train_x,train_y)
# predict the target on the train dataset
predict_train = model.predict(train_x)
# Accuray Score on train dataset
accuracy_train = accuracy_score(train_y,predict_train)
print('Accuracy on train data: ',accuracy_train)
# predict the target on the test dataset
predict_test = model.predict(test_x)
# Accuracy Score on test dataset
accuracy_test = accuracy_score(test_y,predict_test)
print('\nAccuracy on test data: ',accuracy_test)
```

```
Accuracy on train data: 1.0
Accuracy on test data: 0.76767676767
```

In [9]:
```
model = RandomForestClassifier(n_estimators = 50)
# fit the model with the training data
model.fit(train_x,train_y)
# predict the target on the train dataset
predict_train = model.predict(train_x)
# Accuray Score on train dataset
accuracy_train = accuracy_score(train_y,predict_train)
print('Accuracy on train data: ',accuracy_train)
# predict the target on the test dataset
predict_test = model.predict(test_x)
# Accuracy Score on test dataset
accuracy_test = accuracy_score(test_y,predict_test)
print('\nAccuracy on test data: ',accuracy_test)
```

```
Accuracy on train data: 1.0
Accuracy on test data: 0.9823333334
```

In [10]:
```
model.fit(train_x,train_y)
# fit the model with the training data
model.fit(train_x,train_y)
# predict the target on the train dataset
predict_train = model.predict(train_x)
# Accuray Score on train dataset
accuracy_train = accuracy_score(train_y,predict_train)
print('Accuracy on train data: ',accuracy_train)
# predict the target on the test dataset
predict_test = model.predict(test_x)
# Accuracy Score on test dataset
accuracy_test = accuracy_score(test_y,predict_test)
print('\nAccuracy on test data: ',accuracy_test)
```

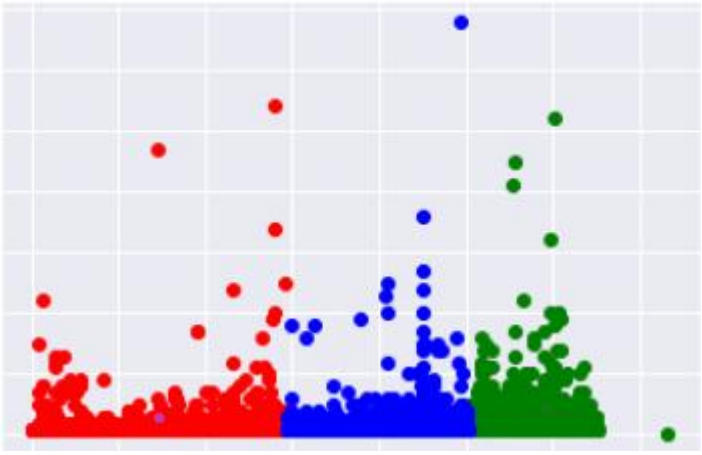
```
Accuracy on train data: 1.0
Accuracy on test data: 0.97959595959
```

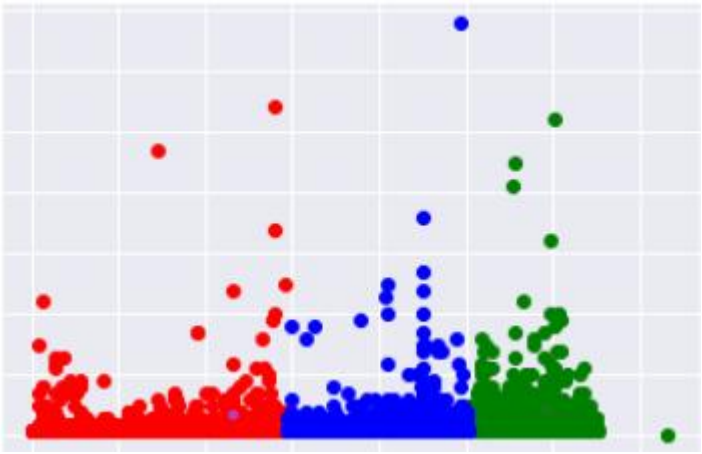
In [11]:
```
model = RandomForestClassifier(criterion = "entropy", n_estimators = 50, min_samples_split=20)
# fit the model with the training data
model.fit(train_x,train_y)
# predict the target on the train dataset
predict_train = model.predict(train_x)
# Accuray Score on train dataset
accuracy_train = accuracy_score(train_y,predict_train)
print('Accuracy on train data: ',accuracy_train)
# predict the target on the test dataset
predict_test = model.predict(test_x)
# Accuracy Score on test dataset
accuracy_test = accuracy_score(test_y,predict_test)
print('\nAccuracy on test data: ',accuracy_test)
```

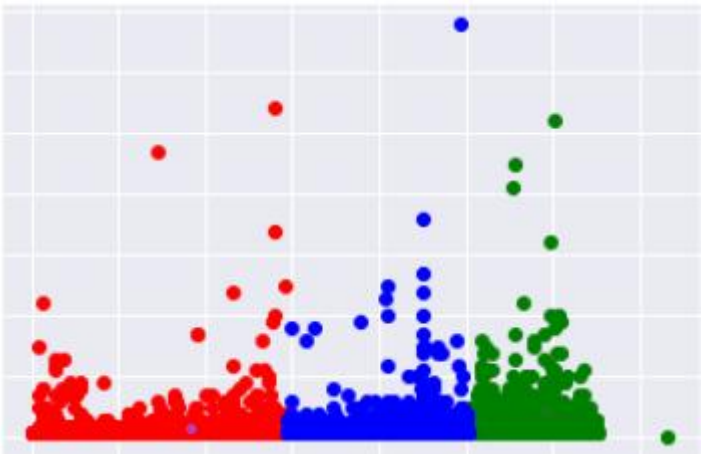
```
Accuracy on train data: 1.0
Accuracy on test data: 0.97866666667
```

AFTER PROPER CLUSTERING OF THE LUNG CANCER DATASET:

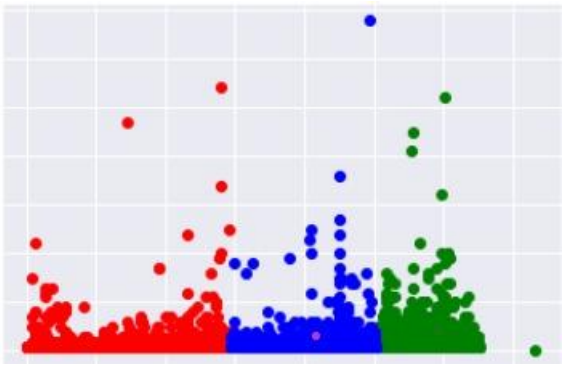
LOW LEVEL CANCER: (For values given, the predictions are shown lying in the cluster according to the prediction)

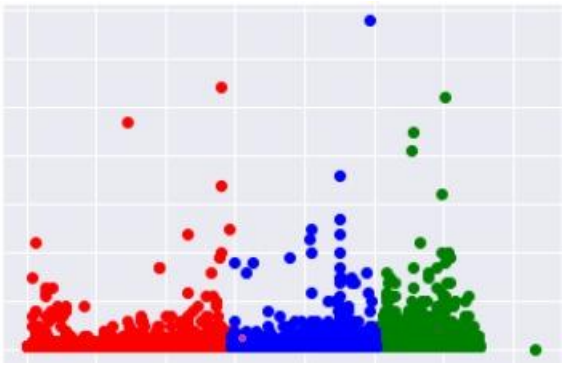


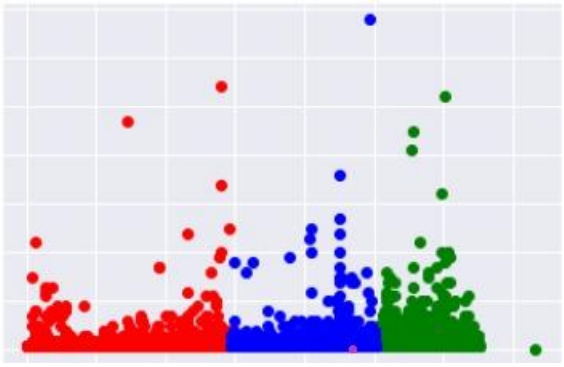




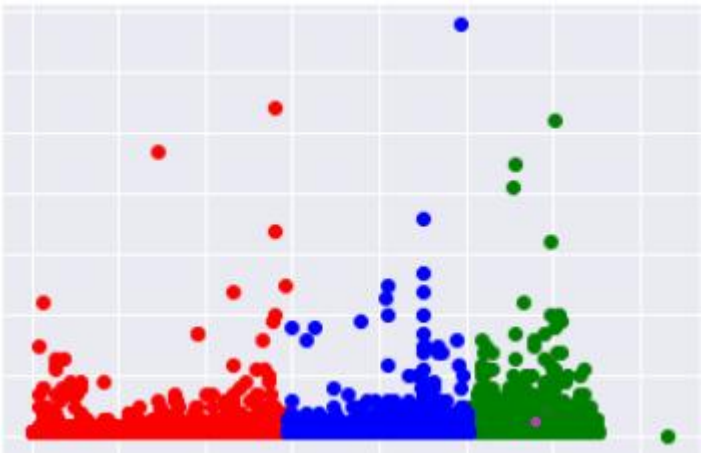
MEDIUM LEVEL CANCER: (For values given, the predictions are shown lying in the cluster according to the prediction)

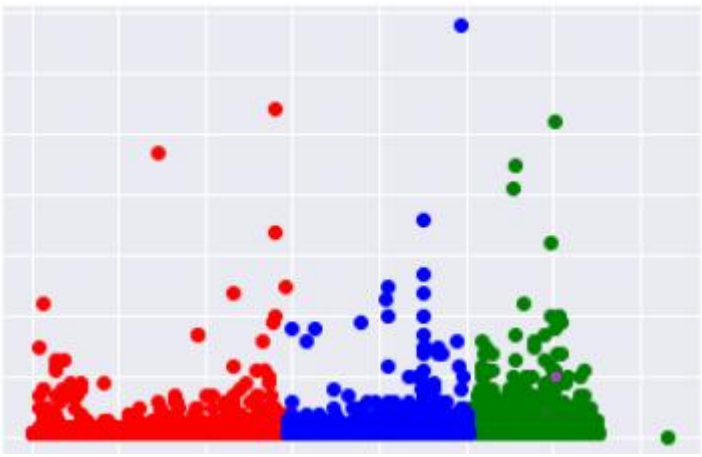


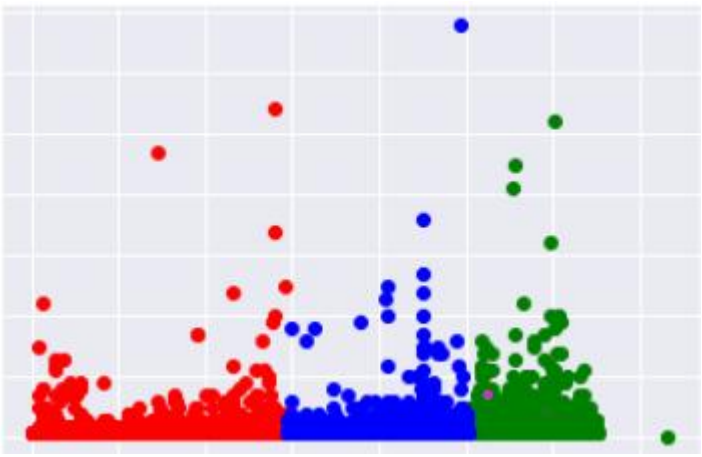




HIGH LEVEL CANCER: (For values given, the predictions are shown lying in the cluster according to the prediction)







Accuracy:

Decision tree: ***0.915***

Logistic regression: ***0.793***

Random forest: ***0.979***

BREAST CANCER PREDICTION

Decision Trees

In [1]:
```
# Load libraries
import pandas as pd
from sklearn.tree import DecisionTreeClassifier # Import Decision Tree Classifier
from sklearn.model_selection import train_test_split # Import train_test_split function
from sklearn import metrics #Import scikit-learn metrics module for accuracy calculation
```

In [31]:
```
col_names = [
                'Diagnosis',
                'Mean_Radius',
                'Mean_Texture',
                'Mean_Perimeter',
                'Mean_Area',
                'Mean_Smoothness',
                'Mean_Compactness',
                'Mean_Concavity',
                'Mean_Concave_Points',
                'Mean_Symmetry',
                'Mean_Fractal_Dimension',
                'Radius_Error',
                'Texture_Error',
                'Perimeter_Error',
                'Area_Error',
                'Smoothness_Error',
                'Compactness_Error',
                'Concavity_Error',
                'Concave_Points_Error',
                'Symmetry_Error',
                'Fractal_Dimension_Error',
                'Worst_Radius',
                'Worst_Texture',
                'Worst_Perimeter',
                'Worst_Area',
                'Worst_Smoothness',
                'Worst_Compactness',
                'Worst_Concavity',
                'Worst_Concave_Points',
                'Worst_Symmetry',
                'Worst_Fractal_Dimension'
]
# Load dataset
pima = pd.read_csv(r'/content/breastcancer.csv', header=None, names=col_names)
```

In [32]:
```
pima.head()
```

Out[32]:

	Diagnosis	Mean_Radius	Mean_Texture	Mean_Perimeter	Mean_Area	Mean_Smoothness	Mean_Compactness	Mean_Concavity\t	Mean_Concave_Points	Mear
0	Diagnosis	Mean_Radius	Mean_Texture	Mean_Perimeter	Mean_Area	Mean_Smoothness	Mean_Compactness	Mean_Concavity	Mean_Concave_Points	Mea
1	M	17.99	10.38	122.8	1001	0.1184	0.2776	0.3001	0.1471	
2	M	20.57	17.77	132.9	1326	0.08474	0.07864	0.0869	0.07017	
3	M	19.69	21.25	130	1203	0.1096	0.1599	0.1974	0.1279	
4	M	11.42	20.38	77.58	386.1	0.1425	0.2839	0.2414	0.1052	

5 rows × 31 columns

In [33]:
```
#split dataset in features and target variable
feature_cols = [
                    'Mean_Radius',
                    'Mean_Texture',
                    'Mean_Perimeter',
                    'Mean_Area',
                    'Mean_Smoothness',
                    'Mean_Compactness',
                    'Mean_Concavity',
                    'Mean_Concave_Points',
                    'Mean_Symmetry',
                    'Mean_Fractal_Dimension',
                    'Radius_Error',
                    'Texture_Error',
                    'Perimeter_Error',
                    'Area_Error',
                    'Smoothness_Error',
                    'Compactness_Error',
                    'Concavity_Error',
                    'Concave_Points_Error',
                    'Symmetry_Error',
                    'Fractal_Dimension_Error',
                    'Worst_Radius',
                    'Worst_Texture',
                    'Worst_Perimeter',
                    'Worst_Area',
                    'Worst_Smoothness',
```

```
                    'Worst_Compactness',
                    'Worst_Concavity',
                    'Worst_Concave_Points',
                    'Worst_Symmetry',
                    'Worst_Fractal_Dimension'
]
X = pima[feature_cols] # Features
y = pima.Diagnosis # Target variable
```

In [34]:
```
# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X[1:], y[1:], test_size=0.3, random_state=1) # 70% training and 30% test
```

DECISION TREE

In [35]:
```
# Create Decision Tree classifer object
clf = DecisionTreeClassifier()
```

In [36]:
```
# Train Decision Tree Classifer
clf = clf.fit(X_train,y_train)
```

In [37]:
```
#Predict the response for test dataset
y_pred = clf.predict(X_test)
```

In [38]:
```
# Model Accuracy, how often is the classifier correct?
print("Accuracy: ",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy:  0.9298245614035088
```

In [39]:
```
pip install graphviz
```

```
Requirement already satisfied: graphviz in /usr/local/lib/python3.7/dist-packages (0.10.1)
```

In [40]:
```
pip install pydotplus
```

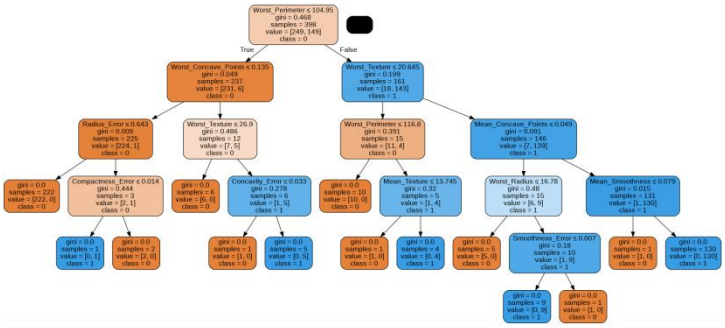
```
Requirement already satisfied: pydotplus in /usr/local/lib/python3.7/dist-packages (2.0.2)
Requirement already satisfied: pyparsing>=2.0.1 in /usr/local/lib/python3.7/dist-packages (from pydotplus) (3.0.8)
```

In [41]:
```
pip install --upgrade scikit-learn==0.20.3
```

```
Requirement already satisfied: scikit-learn==0.20.3 in /usr/local/lib/python3.7/dist-packages (0.20.3)
Requirement already satisfied: scipy>=0.13.3 in /usr/local/lib/python3.7/dist-packages (from scikit-learn==0.20.3) (1.4.1)
Requirement already satisfied: numpy>=1.8.2 in /usr/local/lib/python3.7/dist-packages (from scikit-learn==0.20.3) (1.21.6)
```

In [42]:
```
from sklearn.tree import export_graphviz
from sklearn.externals.six import StringIO
from IPython.display import Image
import pydotplus
```

In [43]:
```
dot_data = StringIO()
export_graphviz(clf, out_file=dot_data,
                filled=True, rounded=True,
                special_characters=True,feature_names = feature_cols,class_names=['0','1'])
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('breastcancer.png')
Image(graph.create_png())
```



In [44]:
```
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

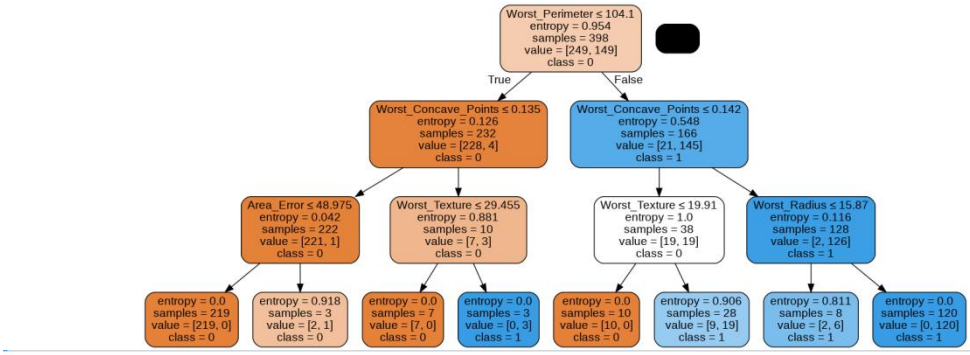
```
Accuracy: 0.9298245614035088
```

In [46]:
```
# Create Decision Tree classifer object
clf = DecisionTreeClassifier(criterion="entropy", max_depth=3)

# Train Decision Tree Classifer
clf = clf.fit(X_train,y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)

dot_data = StringIO()
export_graphviz(clf, out_file=dot_data,
                filled=True, rounded=True,
                special_characters=True,feature_names = feature_cols,class_names=['0','1'])
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('breastcancer.png')
Image(graph.create_png())
```



In [47]:
```
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

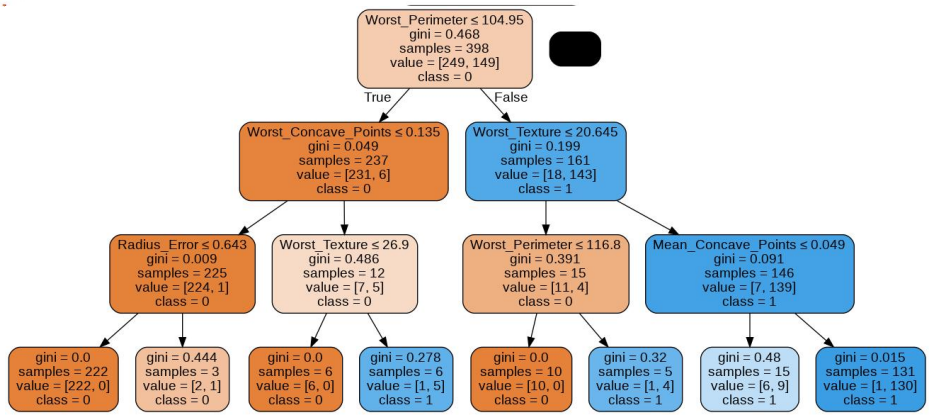
```
Accuracy: 0.8830409356725146
```

In [48]:
```
# Create Decision Tree classifer object
clf = DecisionTreeClassifier(criterion="gini", max_depth=3)

# Train Decision Tree Classifer
clf = clf.fit(X_train,y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)

dot_data = StringIO()
export_graphviz(clf, out_file=dot_data,
                filled=True, rounded=True,
                special_characters=True,feature_names = feature_cols,class_names=['0','1'])
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('breastcancer.png')
Image(graph.create_png())
```



In [49]:
```
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

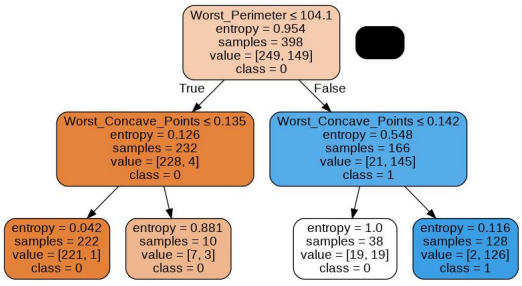
```
Accuracy: 0.9122807017543859
```

In [50]:
```
# Create Decision Tree classifer object
clf = DecisionTreeClassifier(criterion="entropy", max_depth=2)

# Train Decision Tree Classifer
clf = clf.fit(X_train,y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)

dot_data = StringIO()
export_graphviz(clf, out_file=dot_data,
                filled=True, rounded=True,
                special_characters=True,feature_names = feature_cols,class_names=['0','1'])
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('breastcancer.png')
Image(graph.create_png())
```



In [51]:
```
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

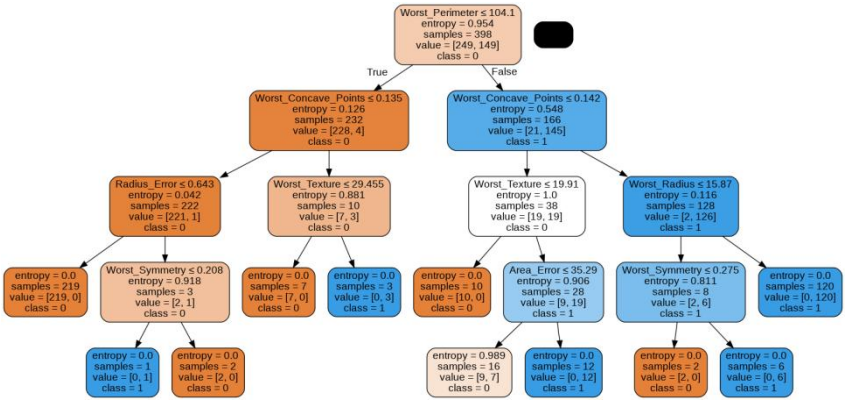
```
Accuracy: 0.8713450292397661
```

In [52]:
```
# Create Decision Tree classifer object
clf = DecisionTreeClassifier(criterion="entropy", max_depth=4)

# Train Decision Tree Classifer
clf = clf.fit(X_train,y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)

dot_data = StringIO()
export_graphviz(clf, out_file=dot_data,
                filled=True, rounded=True,
                special_characters=True,feature_names = feature_cols,class_names=['0','1'])
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('breastcancer.png')
Image(graph.create_png())
```



In [53]:
```
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

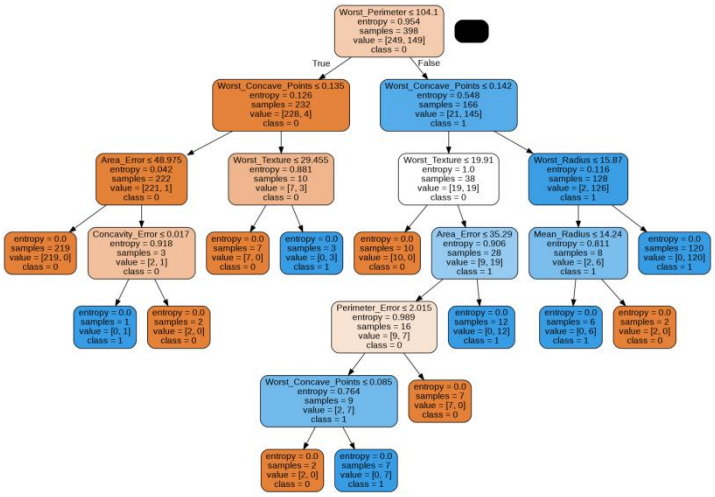
```
Accuracy: 0.9298245614035088
```

In [54]:
```
# Create Decision Tree classifer object
clf = DecisionTreeClassifier(criterion="entropy", max_depth=8)

# Train Decision Tree Classifer
clf = clf.fit(X_train,y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)

dot_data = StringIO()
export_graphviz(clf, out_file=dot_data,
                filled=True, rounded=True,
                special_characters=True,feature_names = feature_cols,class_names=['0','1'])
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('breastcancer.png')
Image(graph.create_png())
```



In [55]:
```
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.9122807017543859
```

Support Vector Machine

In [99]:
```
# Importing the libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [100]:
```
# Importing & Reading the dataset
breastcancer = pd.read_csv("breastcancer.csv")
```

In [101]:
```
df = pd.read_csv("breastcancer.csv")
df.head(10)
```

Out[101]:

	Diagnosis	Mean_Radius	Mean_Texture	Mean_Perimeter	Mean_Area	Mean_Smoothness	Mean_Compactness	Mean_Concavity	Mean_Concave_Points	Mea
0	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	
1	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	
2	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	
3	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	
4	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	
5	M	12.45	15.70	82.57	477.1	0.12780	0.17000	0.15780	0.08089	
6	M	18.25	19.98	119.60	1040.0	0.09463	0.10900	0.11270	0.07400	
7	M	13.71	20.83	90.20	577.9	0.11890	0.16450	0.09366	0.05985	
8	M	13.00	21.82	87.50	519.8	0.12730	0.19320	0.18590	0.09353	
9	M	12.46	24.04	83.97	475.9	0.11860	0.23960	0.22730	0.08543	

10 rows × 31 columns

In [102]:
```
# Our independent variables X
X = breastcancer.iloc[:, 1:31].values
X=X
print(X)
# our dependent variable y
y = breastcancer.iloc[:, 0:1].values
```

```
[[1.799e+01 1.038e+01 1.228e+02 ... 2.654e-01 4.601e-01 1.189e-01]
 [2.057e+01 1.777e+01 1.329e+02 ... 1.860e-01 2.750e-01 8.902e-02]
 [1.969e+01 2.125e+01 1.300e+02 ... 2.430e-01 3.613e-01 8.758e-02]
 ...
 [1.660e+01 2.808e+01 1.083e+02 ... 1.418e-01 2.218e-01 7.820e-02]
 [2.060e+01 2.933e+01 1.401e+02 ... 2.650e-01 4.087e-01 1.240e-01]
 [7.760e+00 2.454e+01 4.792e+01 ... 0.000e+00 2.871e-01 7.039e-02]]
```

In [103]:
```
from sklearn.neural_network import MLPRegressor
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import r2_score
import pandas as pd
```

In [104]:
```
import numpy as np
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y,random_state=1)
```

In [105]:
```
clf = make_pipeline(SVC())
clf.fit(X, y)
y_pred=clf.predict(X_test)

print(clf.score(X_test,y_test))
```

```
0.9536082474226805
```

In [106]:
```
clf = make_pipeline(StandardScaler(),SVC())
clf.fit(X, y)
y_pred=clf.predict(X_test)
print(clf.score(X_test,y_test))
```

```
0.9484536082474226
```

In [107]:
```
clf = make_pipeline(SVC(C=10.0))
clf.fit(X, y)
y_pred=clf.predict(X_test)
print(clf.score(X_test,y_test))
```

```
0.9484536082474226
```

In [108]:
```
clf = make_pipeline(StandardScaler(),SVC(C=10.0))
clf.fit(X, y)
y_pred=clf.predict(X_test)
print(clf.score(X_test,y_test))
```

```
0.9536082474226805
```

In [109]:
```
clf = make_pipeline(SVC(kernel="linear"))
clf.fit(X, y)
y_pred=clf.predict(X_test)
print(clf.score(X_test,y_test))
```

```
0.9484536082474226
```

In [110]:
```
clf = make_pipeline(SVC(kernel="poly"))
clf.fit(X, y)
y_pred=clf.predict(X_test)
print(clf.score(X_test,y_test))
```

```
0.9536082474226805
```

In [111]:
```
clf = make_pipeline(SVC(kernel="rbf"))
clf.fit(X, y)
y_pred=clf.predict(X_test)
print(clf.score(X_test,y_test))
```

```
0.9484536082474226
```

In [112]:
```
clf = make_pipeline(SVC(kernel="sigmoid"))
clf.fit(X, y)
y_pred=clf.predict(X_test)
print(clf.score(X_test,y_test))
```

```
0.9536082474226805
```

In [113]:
```
clf = make_pipeline(StandardScaler(),SVC(kernel="poly",degree=1))
clf.fit(X, y)
y_pred=clf.predict(X_test)
print(clf.score(X_test,y_test))
```

```
0.9484536082474226
```

In [114]:
```
clf = make_pipeline(StandardScaler(),SVC(kernel="poly",degree=2))
clf.fit(X, y)
y_pred=clf.predict(X_test)
print(clf.score(X_test,y_test))
```

```
0.9484536082474226
```

In [115]:
```
clf = make_pipeline(StandardScaler(),SVC(kernel="poly",degree=3))
clf.fit(X, y)
y_pred=clf.predict(X_test)
print(clf.score(X_test,y_test))
```

```
0.9484536082474226
```

In [116]:
```
clf = make_pipeline(StandardScaler(),SVC(kernel="poly",degree=5))
clf.fit(X, y)
y_pred=clf.predict(X_test)
print(clf.score(X_test,y_test))
```

```
0.9484536082474226
```

In [117]:
```
clf = make_pipeline(SVC(kernel="rbf",gamma='scale'))
clf.fit(X, y)
y_pred=clf.predict(X_test)
print(clf.score(X_test,y_test))
```

```
0.9536082474226805
```

In [118]:
```
clf = make_pipeline(SVC(kernel="rbf",gamma='auto'))
clf.fit(X, y)
y_pred=clf.predict(X_test)
print(clf.score(X_test,y_test))
```

```
0.9484536082474226
```

In [119]:
```
clf = make_pipeline(SVC(C=10.0,kernel="rbf",gamma='auto'))
clf.fit(X, y)
y_pred=clf.predict(X_test)
print(clf.score(X_test,y_test))
```

```
0.9484536082474226
```

In [120]:
```
def accuracy(confusion_matrix):
  diagonal_sum = confusion_matrix.trace()
  sum_of_all_elements = confusion_matrix.sum()
  return diagonal_sum / sum_of_all_elements
```

In [121]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y,random_state=1)
clf = make_pipeline(SVC())
clf.fit(X, y)
y_pred=clf.predict(X_test)
print(y_pred)
```

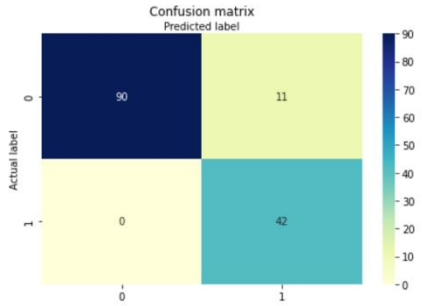
```
['B' 'B' 'M' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'M' 'B' 'M' 'B' 'B' 'M' 'B' 'B'
 'M' 'B' 'M' 'M' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'M' 'B' 'M' 'B' 'B' 'B'
 'M' 'M' 'M' 'B' 'B' 'M' 'M' 'M' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'M' 'B'
 'M' 'M' 'M' 'B' 'B' 'M' 'M' 'M' 'B' 'M' 'B' 'B' 'M' 'M' 'B' 'B' 'B' 'M'
 'M' 'B' 'B' 'B' 'B' 'M' 'M' 'B' 'B' 'B' 'M' 'M' 'B' 'B' 'B' 'B' 'B' 'B'
 'B' 'B' 'M' 'B' 'M' 'B' 'M' 'B' 'B' 'B' 'B' 'M' 'B' 'B' 'B' 'M' 'B' 'B'
 'B' 'B' 'B' 'B' 'M' 'B' 'B' 'M' 'M' 'B' 'B' 'B' 'B' 'B' 'B' 'M' 'B' 'B'
 'B' 'B' 'B' 'B' 'M' 'B' 'B' 'B' 'B' 'B' 'M' 'B' 'B' 'B' 'B' 'B' 'B' 'B']
```

In [122]:
```
#Importing Confusion Matrix
from sklearn.metrics import confusion_matrix
#Comparing the predictions against the actual observations in y_val
cm = confusion_matrix(y_pred, y_test)
print(cm)
#Printing the accuracy
#, accuracy(cm)
print("Accuracy:",accuracy(cm))
```

```
[[90 11]
 [ 0 42]]
Accuracy: 0.9230769230769231
```

In [123]:
```
# Commented out IPython magic to ensure Python compatibility.
# import required modules
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
# %matplotlib inline
class_names=[0,1] # name of classes
fig, ax = plt.subplots()
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)
# create heatmap
sns.heatmap(pd.DataFrame(cm), annot=True, cmap="YlGnBu" ,fmt='g')
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
```

Out[123]: Text(0.5, 257.44, 'Predicted label')



In [124]:
```
def accuracy(confusion_matrix):
  diagonal_sum = confusion_matrix.trace()
  sum_of_all_elements = confusion_matrix.sum()
  return diagonal_sum / sum_of_all_elements
```

In [125]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y,
stratify=y,random_state=1)
clf = make_pipeline(SVC(C=10.0,kernel="rbf",gamma='auto'))
clf.fit(X, y)
y_pred=clf.predict(X_test)
print(y_pred)
```

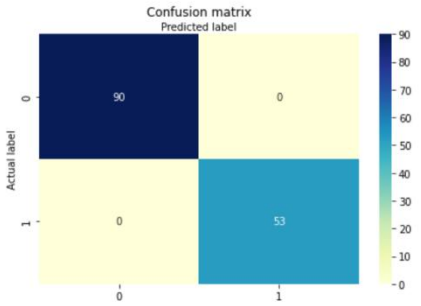
```
['B' 'B' 'M' 'B' 'B' 'M' 'B' 'B' 'B' 'M' 'M' 'B' 'M' 'B' 'B' 'M' 'B' 'B'
 'M' 'B' 'M' 'M' 'B' 'B' 'B' 'M' 'B' 'B' 'B' 'B' 'M' 'M' 'M' 'B' 'M' 'M'
 'M' 'M' 'M' 'B' 'B' 'M' 'M' 'M' 'B' 'B' 'B' 'B' 'B' 'B' 'M' 'B' 'B' 'B'
 'M' 'M' 'M' 'B' 'B' 'M' 'M' 'M' 'B' 'M' 'B' 'B' 'M' 'M' 'B' 'B' 'B' 'M'
 'M' 'B' 'B' 'B' 'M' 'M' 'B' 'B' 'B' 'M' 'M' 'M' 'M' 'B' 'B' 'B' 'B' 'B'
 'B' 'B' 'M' 'B' 'M' 'M' 'M' 'B' 'B' 'B' 'M' 'B' 'B' 'M' 'B' 'M' 'B' 'B'
 'B' 'B' 'B' 'M' 'M' 'B' 'B' 'M' 'M' 'B' 'B' 'M' 'B' 'B' 'B' 'M' 'B' 'B'
 'B' 'B' 'B' 'B' 'M' 'B' 'B' 'B' 'B' 'B' 'M' 'B' 'B' 'B' 'B' 'B' 'B' 'B']
```

In [126]:
```
#Importing Confusion Matrix
from sklearn.metrics import confusion_matrix
#Comparing the predictions against the actual observations in y_val
cm = confusion_matrix(y_pred, y_test)
print(cm)
#Printing the accuracy
print("Accuracy: ",accuracy(cm))
```

```
[[90  0]
 [ 0 53]]
Accuracy: 0.924576358421
```

In [127]:
```
# Commented out IPython magic to ensure Python compatibility.
# import required modules
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
# %matplotlib inline
class_names=[0,1] # name of classes
fig, ax = plt.subplots()
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)
# create heatmap
sns.heatmap(pd.DataFrame(cm), annot=True, cmap="YlGnBu" ,fmt='g')
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
```

Out[127]: Text(0.5, 257.44, 'Predicted label')



Random Forest

In [ ]:
```
# importing required libraries
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn import model_selection
```

In [ ]:
```
# read the train and test dataset
dat = pd.read_csv('breastcancer.csv')
```

In [ ]:
```
# view the top 3 rows of the dataset
print(dat.head(9))
```

```
  Diagnosis  Mean_Radius  Mean_Texture  Mean_Perimeter  Mean_Area  \
0         M        17.99         10.38          122.80     1001.0
1         M        20.57         17.77          132.90     1326.0
2         M        19.69         21.25          130.00     1203.0
3         M        11.42         20.38           77.58      386.1
4         M        20.29         14.34          135.10     1297.0
5         M        12.45         15.70           82.57      477.1
6         M        18.25         19.98          119.60     1040.0
7         M        13.71         20.83           90.20      577.9
8         M        13.00         21.82           87.50      519.8

   Mean_Smoothness  Mean_Compactness  Mean_Concavity  Mean_Concave_Points  \
0          0.11840           0.27760         0.30010              0.14710
1          0.08474           0.07864         0.08690              0.07017
2          0.10960           0.15990         0.19740              0.12790
3          0.14250           0.28390         0.24140              0.10520
4          0.10030           0.13280         0.19800              0.10430
5          0.12780           0.17000         0.15780              0.08089
6          0.09463           0.10900         0.11270              0.07400
7          0.11890           0.16450         0.09366              0.05985
8          0.12730           0.19320         0.18590              0.09353

   Mean_Symmetry  ...  Worst_Radius  Worst_Texture  Worst_Perimeter  \
0         0.2419  ...         25.38          17.33           184.60
1         0.1812  ...         24.99          23.41           158.80
2         0.2069  ...         23.57          25.53           152.50
3         0.2597  ...         14.91          26.50            98.87
4         0.1809  ...         22.54          16.67           152.20
5         0.2087  ...         15.47          23.75           103.40
6         0.1794  ...         22.88          27.66           153.20
7         0.2196  ...         17.06          28.14           110.60
8         0.2350  ...         15.49          30.73           106.20

   Worst_Area  Worst_Smoothness  Worst_Compactness  Worst_Concavity  \
0      2019.0            0.1622             0.6656           0.7119
1      1956.0            0.1238             0.1866           0.2416
2      1709.0            0.1444             0.4245           0.4504
3       567.7            0.2098             0.8663           0.6869
4      1575.0            0.1374             0.2050           0.4000
5       741.6            0.1791             0.5249           0.5355
6      1606.0            0.1442             0.2576           0.3784
7       897.0            0.1654             0.3682           0.2678
8       739.3            0.1703             0.5401           0.5390

   Worst_Concave_Points  Worst_Symmetry  Worst_Fractal_Dimension
0                0.2654          0.4601                  0.11890
1                0.1860          0.2750                  0.08902
2                0.2430          0.3613                  0.08758
3                0.2575          0.6638                  0.17300
4                0.1625          0.2364                  0.07678
5                0.1741          0.3985                  0.12440
6                0.1932          0.3063                  0.08368
7                0.1556          0.3196                  0.11510
8                0.2060          0.4378                  0.10720

[9 rows x 31 columns]
```

In [ ]:
```
dat.describe().transpose()
```

Out[30]:

	count	mean	std	min	25%	50%	75%	max
Mean_Radius	569.0	14.127292	3.524049	6.981000	11.700000	13.370000	15.780000	28.11000
Mean_Texture	569.0	19.289649	4.301036	9.710000	16.170000	18.840000	21.800000	39.28000
Mean_Perimeter	569.0	91.969033	24.298981	43.790000	75.170000	86.240000	104.100000	188.50000
Mean_Area	569.0	654.889104	351.914129	143.500000	420.300000	551.100000	782.700000	2501.00000
Mean_Smoothness	569.0	0.096360	0.014064	0.052630	0.086370	0.095870	0.105300	0.16340
Mean_Compactness	569.0	0.104341	0.052813	0.019380	0.064920	0.092630	0.130400	0.34540
Mean_Concavity	569.0	0.088799	0.079720	0.000000	0.029560	0.061540	0.130700	0.42680
Mean_Concave_Points	569.0	0.048919	0.038803	0.000000	0.020310	0.033500	0.074000	0.20120
Mean_Symmetry	569.0	0.181162	0.027414	0.106000	0.161900	0.179200	0.195700	0.30400
Mean_Fractal_Dimension	569.0	0.062798	0.007060	0.049960	0.057700	0.061540	0.066120	0.09744
Radius_Error	569.0	0.405172	0.277313	0.111500	0.232400	0.324200	0.478900	2.87300
Texture_Error	569.0	1.216853	0.551648	0.360200	0.833900	1.108000	1.474000	4.88500
Perimeter_Error	569.0	2.866059	2.021855	0.757000	1.606000	2.287000	3.357000	21.98000
Area_Error	569.0	40.337079	45.491006	6.802000	17.850000	24.530000	45.190000	542.20000
Smoothness_Error	569.0	0.007041	0.003003	0.001713	0.005169	0.006380	0.008146	0.03113
Compactness_Error	569.0	0.025478	0.017908	0.002252	0.013080	0.020450	0.032450	0.13540
Concavity_Error	569.0	0.031894	0.030186	0.000000	0.015090	0.025890	0.042050	0.39600
Concave_Points_Error	569.0	0.011796	0.006170	0.000000	0.007638	0.010930	0.014710	0.05279
Symmetry_Error	569.0	0.020542	0.008266	0.007882	0.015160	0.018730	0.023480	0.07895
Fractal_Dimension_Error	569.0	0.003795	0.002646	0.000895	0.002248	0.003187	0.004558	0.02984
Worst_Radius	569.0	16.269190	4.833242	7.930000	13.010000	14.970000	18.790000	36.04000
Worst_Texture	569.0	25.677223	6.146258	12.020000	21.080000	25.410000	29.720000	49.54000
Worst_Perimeter	569.0	107.261213	33.602542	50.410000	84.110000	97.660000	125.400000	251.20000
Worst_Area	569.0	880.583128	569.356993	185.200000	515.300000	686.500000	1084.000000	4254.00000
Worst_Smoothness	569.0	0.132369	0.022832	0.071170	0.116600	0.131300	0.146000	0.22260
Worst_Compactness	569.0	0.254265	0.157336	0.027290	0.147200	0.211900	0.339100	1.05800
Worst_Concavity	569.0	0.272188	0.208624	0.000000	0.114500	0.226700	0.382900	1.25200
Worst_Concave_Points	569.0	0.114606	0.065732	0.000000	0.064930	0.099930	0.161400	0.29100
Worst_Symmetry	569.0	0.290076	0.061867	0.156500	0.250400	0.282200	0.317900	0.66380
Worst_Fractal_Dimension	569.0	0.083946	0.018061	0.055040	0.071460	0.080040	0.092080	0.20750

In [ ]:
```
x1 = dat.drop('Diagnosis',axis=1).values
y1 = dat['Diagnosis'].values
```

In [ ]:
```
train_x, test_x, train_y, test_y = model_selection.train_test_split(x1,y1,test_size=0.34,random_state=100)
print('\nShape of training data :',train_x.shape)
print('\nShape of testing data :',test_y.shape)
```

```
Shape of training data : (375, 30)

Shape of testing data : (194,)
```

In [ ]:
```
model = RandomForestClassifier()
# fit the model with the training data
model.fit(train_x,train_y)
# number of trees used
print('Number of trees used: ',model.n_estimators)
# predict the target on the train dataset
predict_train = model.predict(train_x)
# Accuray Score on train dataset
accuracy_train = accuracy_score(train_y,predict_train)
print('Accuracy on train dataset: ',accuracy_train)
# predict the target on the test dataset
predict_test = model.predict(test_x)
# Accuracy Score on test dataset
accuracy_test = accuracy_score(test_y,predict_test)
print('\nAccuracy on test dataset: ',accuracy_test)
```

```
Number of trees used:  100
Accuracy on train dataset:  1.0

Accuracy on test dataset:  0.9536082474226805
```

In [ ]:
```
model = RandomForestClassifier(criterion = "entropy")
# fit the model with the training data
model.fit(train_x,train_y)
# number of trees used
print('Number of trees used: ',model.n_estimators)
# predict the target on the train dataset
predict_train = model.predict(train_x)
# Accuray Score on train dataset
accuracy_train = accuracy_score(train_y,predict_train)
print('Accuracy on train dataset: ',accuracy_train)
# predict the target on the test dataset
predict_test = model.predict(test_x)
# Accuracy Score on test dataset
accuracy_test = accuracy_score(test_y,predict_test)
print('\nAccuracy on test dataset: ',accuracy_test)
```

```
Number of trees used:  100
Accuracy on train dataset:  1.0

Accuracy on test dataset:  0.9484536082474226
```

In [ ]:
```
model = RandomForestClassifier(n_estimators = 50)
# fit the model with the training data
model.fit(train_x,train_y)
# number of trees used
print('Number of trees used: ',model.n_estimators)
# predict the target on the train dataset
predict_train = model.predict(train_x)
# Accuray Score on train dataset
accuracy_train = accuracy_score(train_y,predict_train)
print('Accuracy on train dataset: ',accuracy_train)
# predict the target on the test dataset
predict_test = model.predict(test_x)
# Accuracy Score on test dataset
accuracy_test = accuracy_score(test_y,predict_test)
print('\nAccuracy on test dataset: ',accuracy_test)
```

```
Number of trees used:  50
Accuracy on train dataset:  1.0

Accuracy on test dataset:  0.9536082474226805
```

In [ ]:
```
model = RandomForestClassifier(criterion = "entropy", n_estimators = 50)
# fit the model with the training data
model.fit(train_x,train_y)
# number of trees used
print('Number of trees used: ',model.n_estimators)
# predict the target on the train dataset
predict_train = model.predict(train_x)
# Accuray Score on train dataset
accuracy_train = accuracy_score(train_y,predict_train)
print('Accuracy on train dataset: ',accuracy_train)
# predict the target on the test dataset
predict_test = model.predict(test_x)
# Accuracy Score on test dataset
accuracy_test = accuracy_score(test_y,predict_test)
print('\nAccuracy on test dataset: ',accuracy_test)
```

```
Number of trees used:  50
Accuracy on train dataset:  1.0

Accuracy on test dataset:  0.9536082474226805
```

In [44]:
```
model = RandomForestClassifier(criterion = "entropy", n_estimators = 50, min_samples_split=20)
# fit the model with the training data
model.fit(train_x,train_y)
# number of trees used
print('Number of trees used: ',model.n_estimators)
# predict the target on the train dataset
predict_train = model.predict(train_x)
# Accuray Score on train dataset
accuracy_train = accuracy_score(train_y,predict_train)
print('Accuracy on train dataset: ',accuracy_train)
# predict the target on the test dataset
predict_test = model.predict(test_x)
# Accuracy Score on test dataset
accuracy_test = accuracy_score(test_y,predict_test)
print('\nAccuracy on test dataset: ',accuracy_test)
```

```
Number of trees used:  50
Accuracy on train dataset: 1.0

Accuracy on test dataset: 0.98313131313134
```

Accuracy:

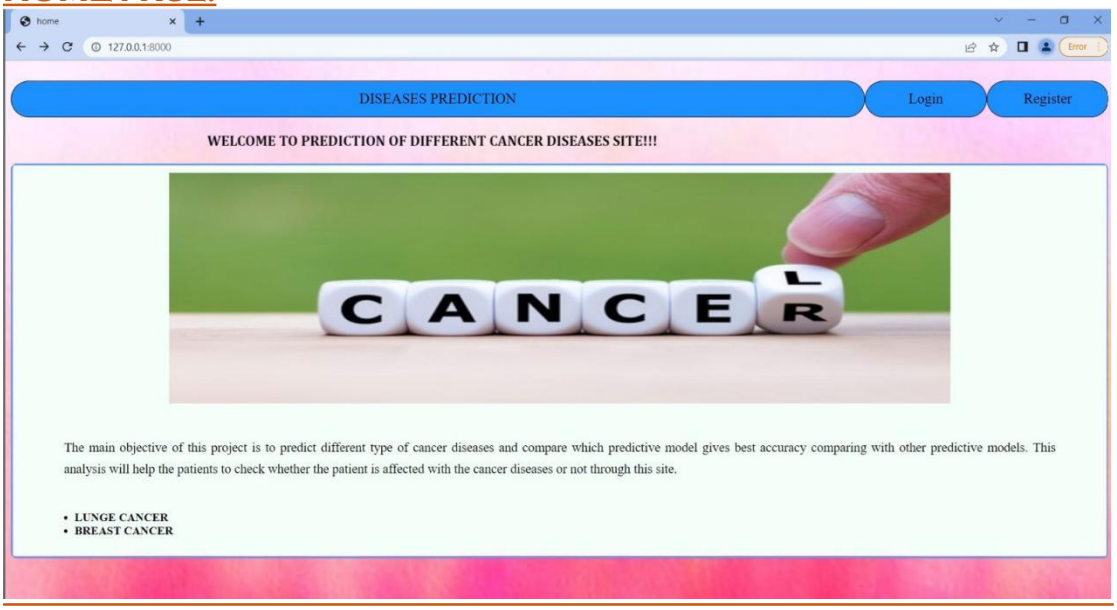
Decision tree: ***0.93***

SVM: ***0.954***

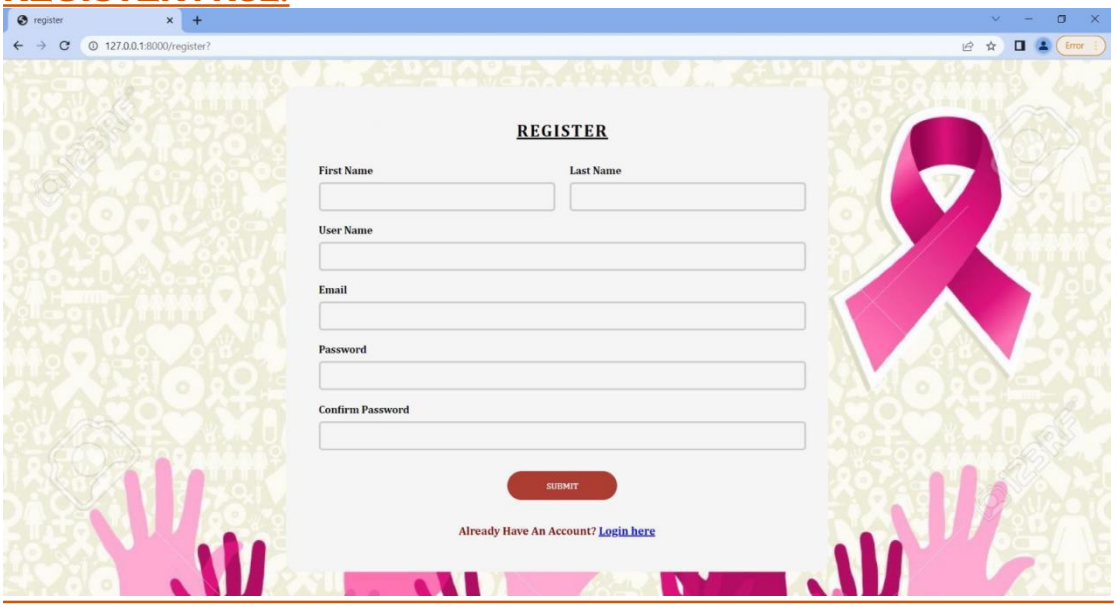
Random forest: ***0.98***

WORKING ON OUR PROJECT

HOME-PAGE:

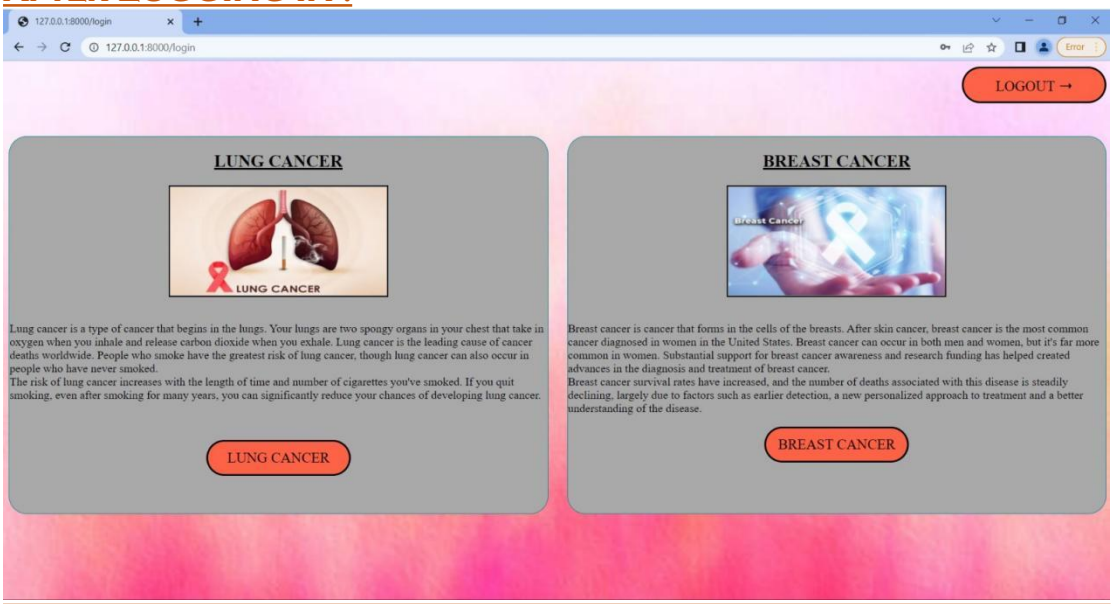


REGISTER-PAGE:



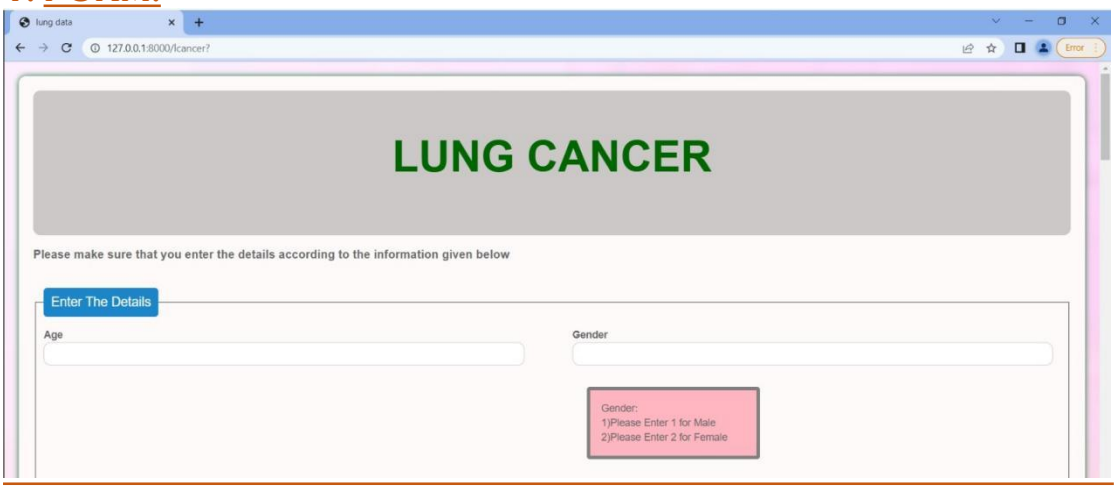
LOGIN-PAGE:

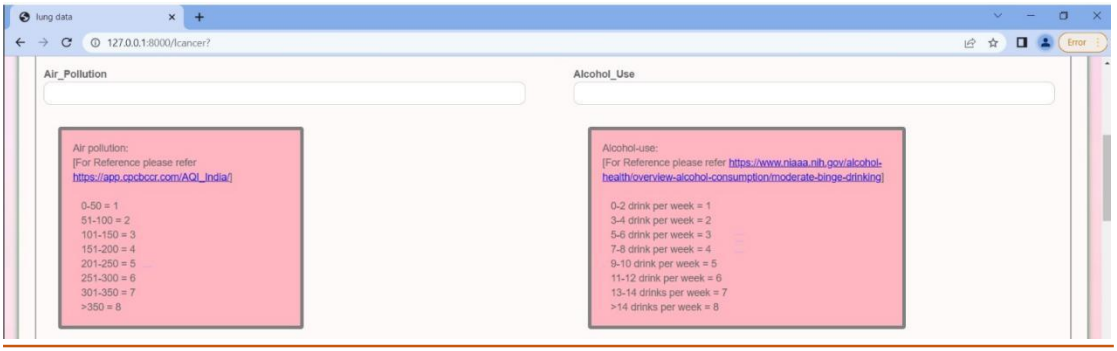
AFTER LOGGING IN :

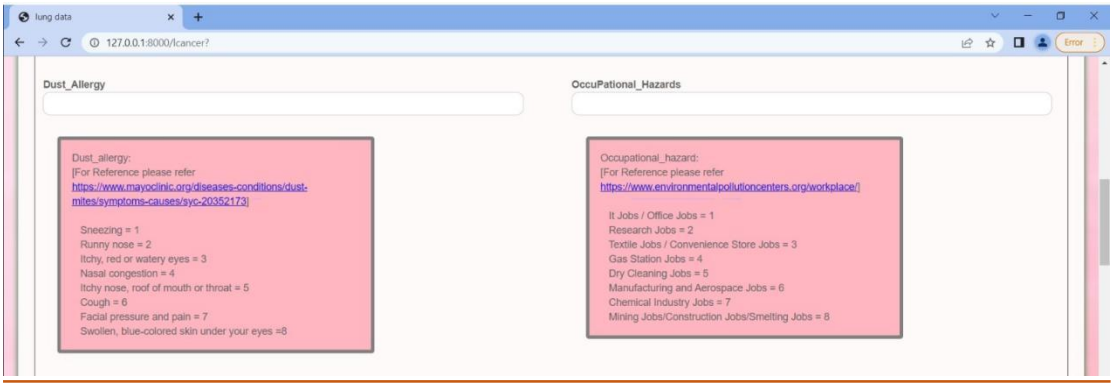


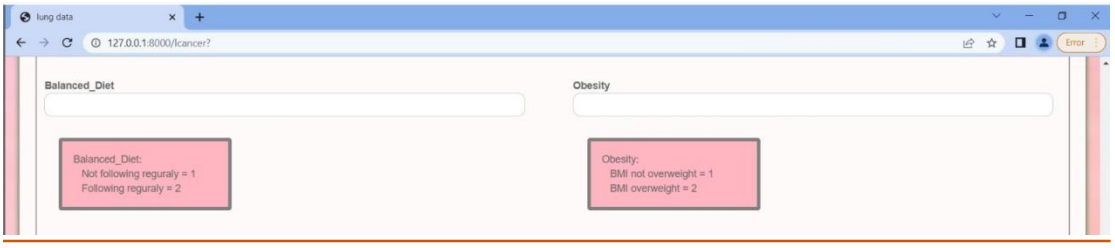
LUNG_CANCER

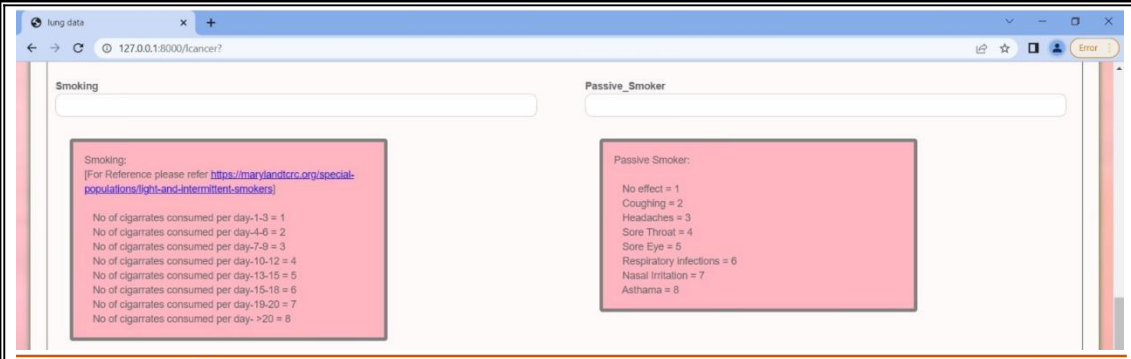
1. FORM:

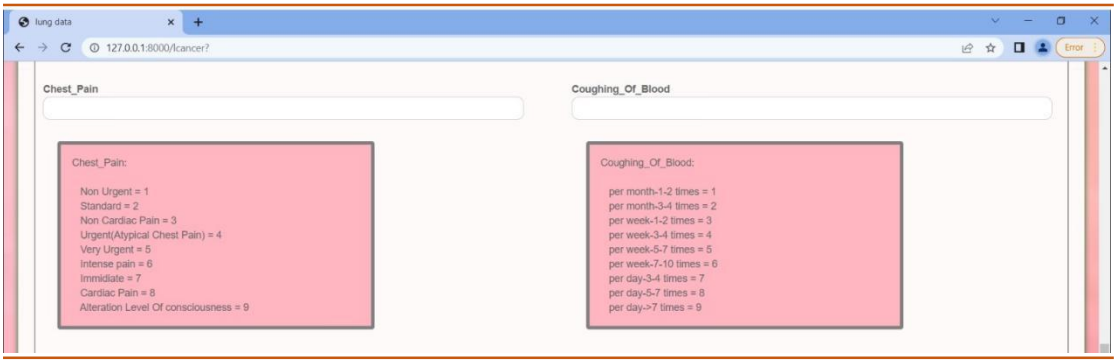




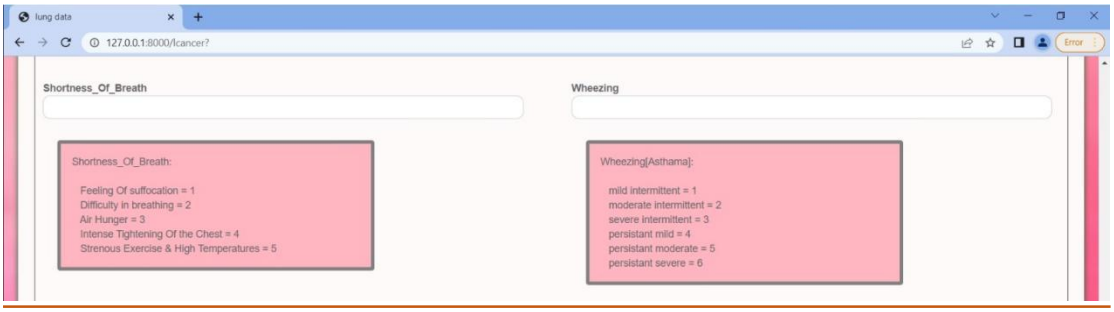


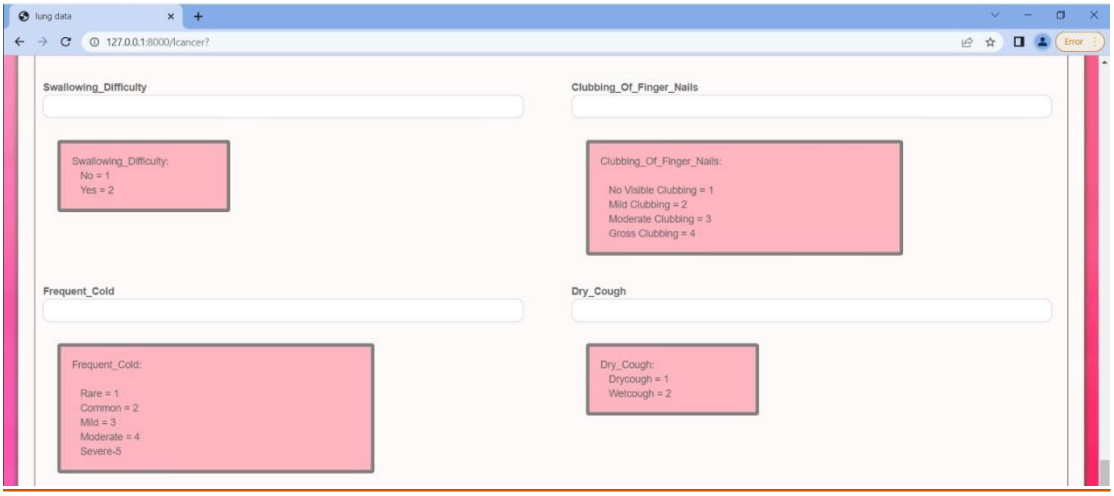


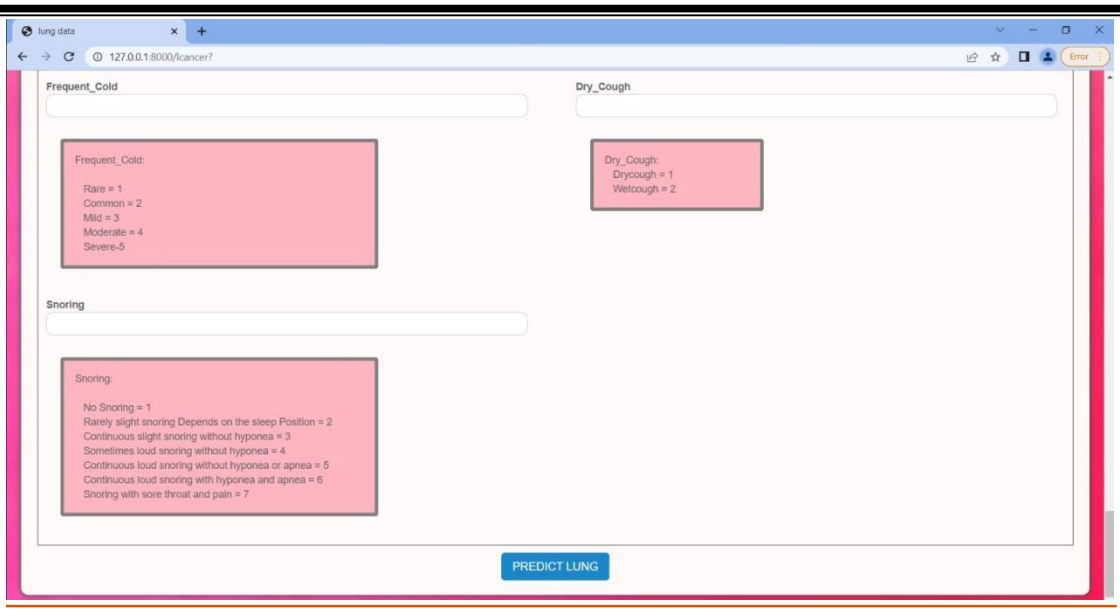

lung data
127.0.0.1:8000/lcancer?
Error
Smoking
Smoking:
[For Reference please refer https://marylandtccc.org/special-populations/light-and-intermittent-smokers]
No of cigarrates consumed per day-1-3 = 1
No of cigarrates consumed per day-4-6 = 2
No of cigarrates consumed per day-7-9 = 3
No of cigarrates consumed per day-10-12 = 4
No of cigarrates consumed per day-13-15 = 5
No of cigarrates consumed per day-15-18 = 6
No of cigarrates consumed per day-19-20 = 7
No of cigarrates consumed per day- >20 = 8
Passive_Smoker
Passive Smoker:
No effect = 1
Coughing = 2
Headaches = 3
Sore Throat = 4
Sore Eye = 5
Respiratory infections = 6
Nasal Irritation = 7
Asthama = 8


lung data
127.0.0.1:8000/lcancer?
Error
Chest_Pain
Chest_Pain:
Non Urgent = 1
Standard = 2
Non Cardiac Pain = 3
Urgent(Atypical Chest Pain) = 4
Very Urgent = 5
Intense pain = 6
Immidiate = 7
Cardiac Pain = 8
Alteration Level Of consciousness = 9
Coughing_Of_Blood
Coughing_Of_Blood:
per month-1-2 times = 1
per month-3-4 times = 2
per week-1-2 times = 3
per week-3-4 times = 4
per week-5-7 times = 5
per week-7-10 times = 6
per day-3-4 times = 7
per day-5-7 times = 8
per day->7 times = 9

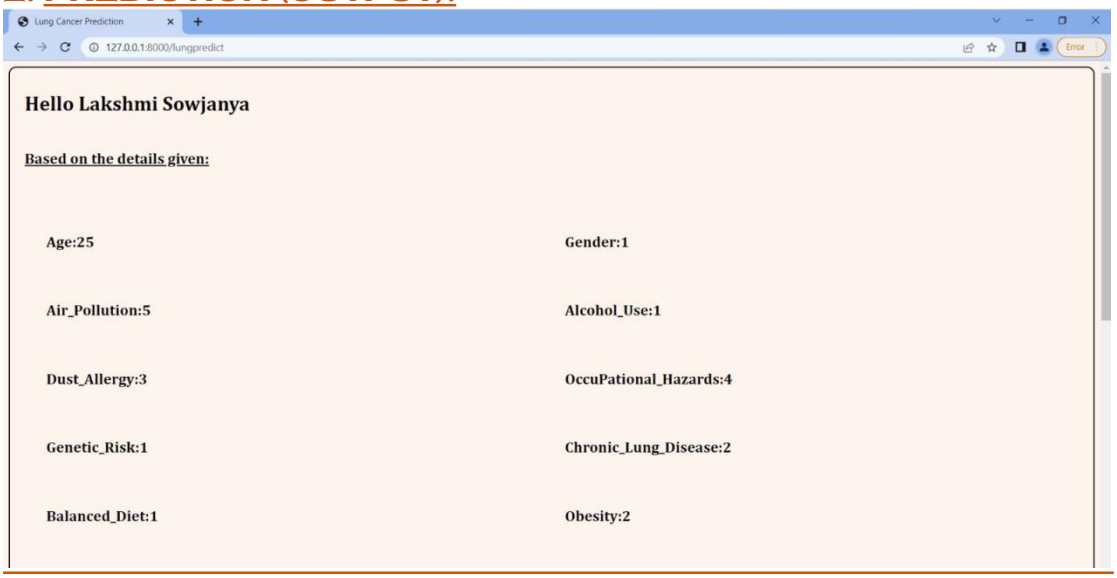

lung data
127.0.0.1:8000/lcancer?
Error
Fatigue
Fatigue:
Fully Alert = 1
Less Alert(Okay) = 2
Dehydrated+Drowsy = 3
Tired = 4
Streesed+Struggling = 5
Normal Emergency = 6
Extremely Urgent = 7
Weight_Loss
Weight_Loss:
1-2 kg per month = 1
3-4 kg per month = 2
5-6 kg per month = 3
7-8 kg per month = 4
9-10 kg per month = 5
10-12 kg per month = 6
13-15 kg per month = 7
>15 kg per month = 8

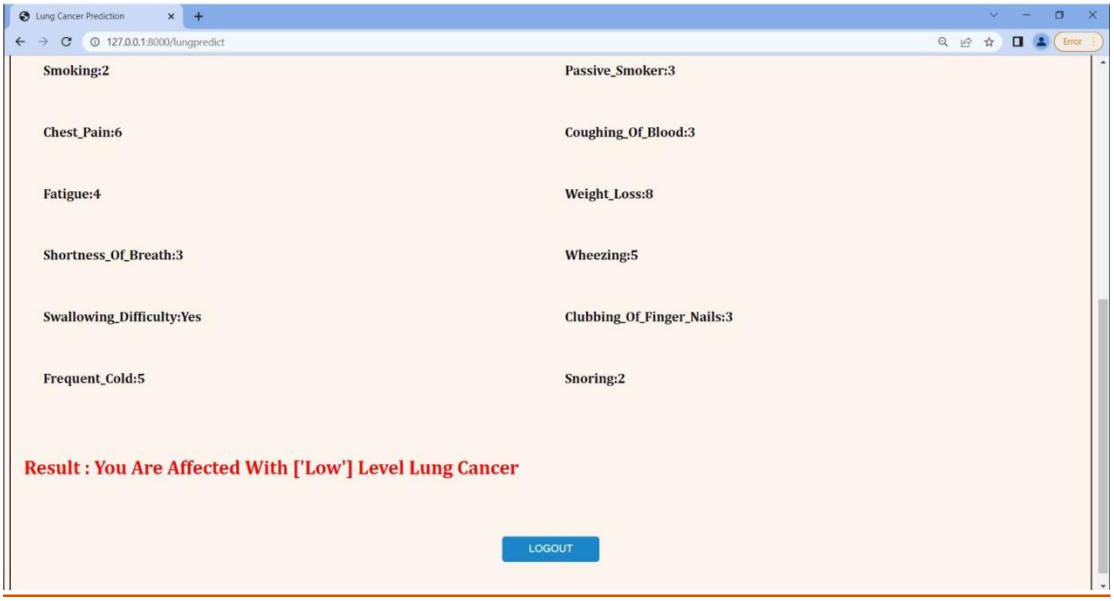

lung data
127.0.0.1:8000/lcancer?
Error
Shortness_Of_Breath
Shortness_Of_Breath:
Feeling Of suffocation = 1
Difficulty in breathing = 2
Air Hunger = 3
Intense Tightening Of the Chest = 4
Strenous Exercise & High Temperatures = 5
Wheezing
Wheezing[Asthama]:
mild intermittent = 1
moderate intermittent = 2
severe intermittent = 3
persistant mild = 4
persistant moderate = 5
persistant severe = 6


lung data
127.0.0.1:8000/lcancer?
Error
Swallowing_Difficulty
Swallowing_Difficulty:
No = 1
Yes = 2
Clubbing_Of_Finger_Nails
Clubbing_Of_Finger_Nails:
No Visible Clubbing = 1
Mild Clubbing = 2
Moderate Clubbing = 3
Gross Clubbing = 4
Frequent_Cold
Frequent_Cold:
Rare = 1
Common = 2
Mild = 3
Moderate = 4
Severe-5
Dry_Cough
Dry_Cough:
Drycough = 1
Wetcough = 2



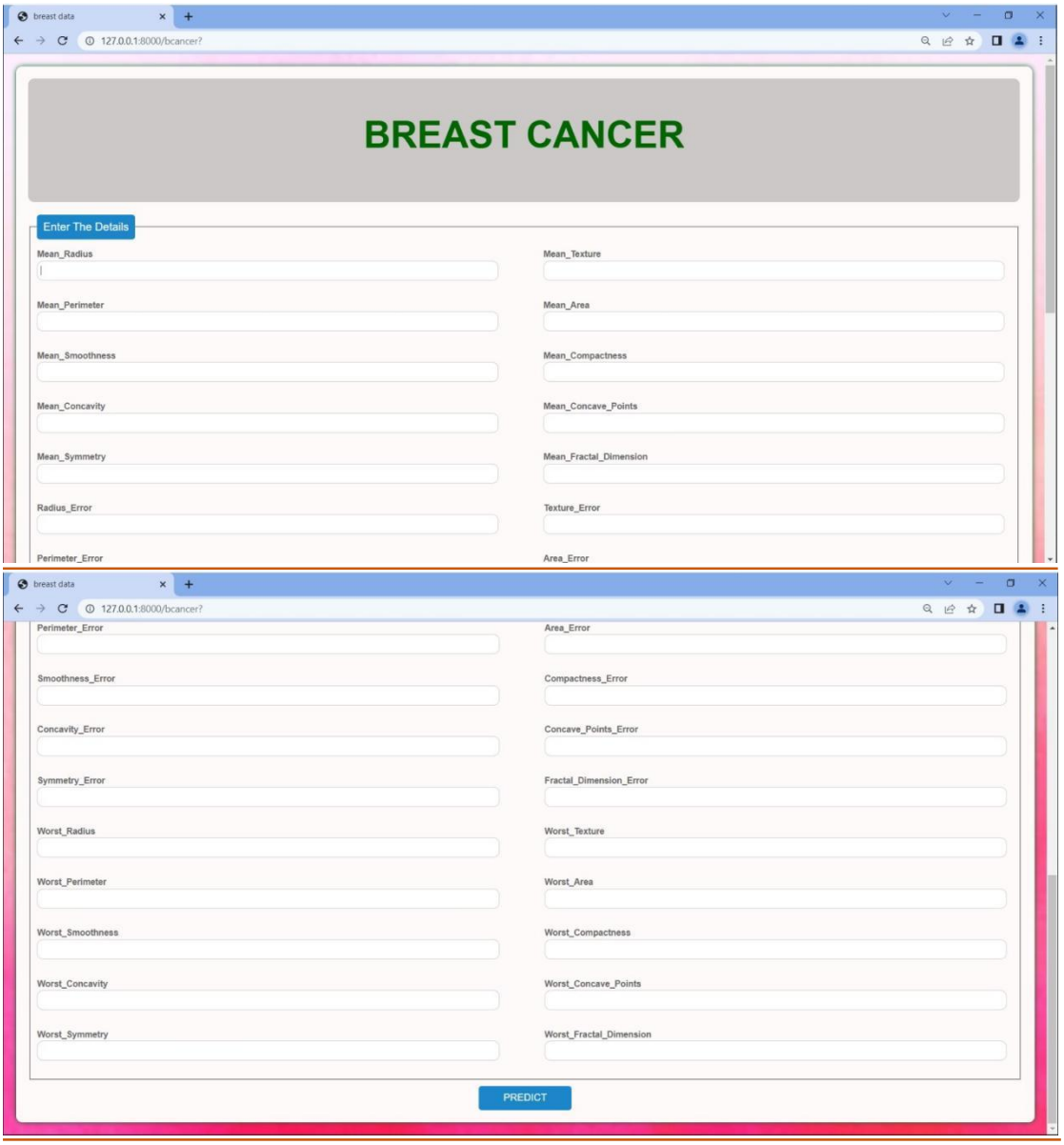
2. PREDICTION (OUTPUT):





BREAST CANCER

1. FORM:



2. PREDICTION (OUTPUT):

Hello samantha

Based on the details given:

Mean_Radius:25	Mean_Texture:3
Mean_Perimeter:625	Mean_Area:15
Mean_Smoothness:25	Mean_Compactness:6
Mean_Concavity:1	Mean_Concave_Points:23
Mean_Symmetry:25	Mean_Fractal_Dimension:6
Radius_Error:2	Texture_Error:9
Perimeter_Error:56	Area_Error:15
Smoothness_Error:45	Compactness_Error:4
Concavity_Error:62	Concave_Points_Error:2
Symmetry_Error:45	Fractal_Dimension_Error:6
Worst_Radius:54	Worst_Texture:6
Worst_Perimeter:45	Worst_Area:62
Worst_Smoothness:45	Worst_Compactness:62
Worst_Concavity:65	Worst_Concave_Points:65
Worst_Symmetry:15	Worst_Fractal_Dimension:15

Result : You Are ['Affected']

LOGOUT

CONCLUSION

By the end of the project, we want to conclude that with dataset (can be symptoms / lab-records) and appropriate Machine Learning algorithm we can predict the disease. We chose Lung cancer and breast cancer in this project and created a website where we give our symptoms/lab-records as input and get the severity of disease as output. As Random Forest worked well and gave the best accuracy for both the datasets (lungcancer.csv and breastcancer.csv) we chose that to predict the outputs. We can also extend the scope of our projects by adding other diseases too like Diabetes, Malaria, Jaundice, Dengue, and Tuberculosis.

The outcome of this project helps patients to predict the severity of their disease with no cost and time. And it also helps doctors deal patients will better equipment and data. This project is beneficial to both Patients and doctors.In this project, the lung cancer prediction is symptom based and breast cancer prediction is dependent on values from the Lab-Reports.

We developed prediction application which predicts both the lung and breast cancer with accuracy of nearly 0.98 for both cases and in future we can extend the project to other disease predictions.

Lung cancer:

ML ALGORITHM	ACCURACY
DECISION TREE	0.915
LOGISTIC REGRESSION	0.795
RANDOM FOREST	0.979

Breast cancer:

ALGORITHMS USED	ACCURACIES
DECISION TREE	**0.93**
SVM	**0.954**
RANDOM FOREST	**0.98**

REFERENCES

- Datasets:
  - ✓ breast cancer datasets: https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data
  - ✓ lung cancer datasets: https://data.world/cancerdatahp/lung-cancer-data

- Reference papers:
  1) https://ieeexplore.ieee.org/abstract/document/8869001
  2) https://link.springer.com/chapter/10.1007/978-981-16-0443-0_26
  3) https://www.sciencedirect.com/science/article/pii/S2214785321027206
  4) https://ieeexplore.ieee.org/document/8739696
  5) https://link.springer.com/article/10.1007/s42979-020-00305-w
  6) https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.295.11&rep=rep1&type=pdf
  7) https://link.springer.com/chapter/10.1007/978-981-15-6648-6_11
  8) https://aacrjournals.org/cancerpreventionresearch/article/1/4/250/46410/An-Expanded Risk Prediction-Model-for-Lung
  9) https://www.sciencedirect.com/science/article/pii/S1877050921014629
  10) https://www.sciencedirect.com/science/article/pii/S1877050918309323

- Reference for web-development part of the project: https://www.youtube.com/watch?v=oyrda7utuha

GOOGLE DRIVE LINK FOR PROJECT CODES :
https://drive.google.com/drive/folders/1Q_RJHtPnIG9z3VhYVFvp071grI79gwdw?usp=sharing

THE END