RECURRENT GASTRIC CANCER PREDICTION USING RANDOMIZED SEARCH CV OPTIMIZER

A PROJECT REPORT

Submitted by

VEDHAPRIYAA .S

19CSR217

VISHAL RUPAK. V.R

19CSR223

VISHNU.M.K

19CSR225

in partial fulfillment of the

requirements for the award of the

degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

KONGU ENGINEERING COLLEGE

(Autonomous)

PERUNDURAI ERODE-638060

AUGUST 2022

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

KONGU ENGINEERING COLLEGE

(Autonomous)

PERUNDURAI, ERODE -638060

AUGUST 2022

BONAFIDE CERTIFICATE

This is to certify that the Project Report **entitled RECURRENT GASTRIC CANCER PREDICTION USING RANDOMIZED SEARCH CV OPTIMIZER** is the bonafide record of project work done by **VEDHAPRIYAA.S (Register No:19CSR217), VISHAL RUPAK V.R (Register No:19CSR223), VISHNU.M.K (Register No:19CSR225)** in partial fulfillment of the requirements for the award of the Degree of Bachelor of Engineering in **Computer Science and Engineering** of Anna University, Chennai during the year 2021-2022.

SUPERVISOR **HEAD OF THE DEPARTMENT**

(Signature with seal)

Date:

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

KONGU ENGINEERING COLLEGE

(Autonomous)

PERUNDURAI, ERODE -638060

AUGUST 2022

DECLARATION

We affirm that Project Report titled **RECURRENT GASTRIC CANCER PREDICTION USING RANDOMIZED SEARCH CV OPTIMIZER** being submitted in partial fulfillment of the requirements for the award of Degree of Bachelor of Engineering is the original work carried out by us. It has not formed part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

Date: **S.VEDHAPRIYAA**

(Reg. No.:19CSR217)

V.R.VISHAL RUPAK

(Reg. No.:19CSR223)

M.K.VISHNU

(Reg. No.:19CSR225)

I certify that the declaration made by the above candidate is true to the best of my knowledge.

Name & Signature of the Supervisor with seal

Date:

ABSTRACT

Gastric cancer has the second-highest mortality rate and the fourth-highest incidence of malignant tumors worldwide. Early stomach cancer has a fair prognosis, but the symptoms are unusual and the signs are not immediately apparent. The goal of the project was to use patient clinicopathological data and machine learning techniques to increase the precision of a prediction model for the emergence of gastric cancer. The current study proposed the first four parameters influencing postoperative recurrence of gastric cancer and shown that machine learning techniques can be used to predict the recurrence of patients with gastric cancer after an operation by evaluating the performance of Logistic regression, Decision Tree, Gradient Boosted Decision Tree, Random Forest, and Gradient Boosting Machine. In this case, Random Forest is enhanced using the Random Search Cross Validation optimizer to increase the model's performance in predicting cancer. Random Search Cross Validation performs a random search over parameters, sampling each setting from a range of potential parameter values. The dataset contains 2012 patient records as well as 51 characteristics.

ACKNOWLEDGEMENT

Owing deeply to the supreme, we extend our thanks to the Almighty, who has blessed us to come out successfully with our project. We take pleasure in expressing our deep sense of gratitude to our beloved parents.

We extend our hearty gratitude to our honorable Correspondent **Thiru.P.Sachithanandan,** and other trust members for having provided us with all necessary infrastructures to undertake this project. We extend our thanks to our Principal, **Dr.V.Balusamy B.E(Hons).,M.Tech.,PhD,** for his consistent encouragement throughout our college days.

We express our deep sense of gratitude to **Dr. N.Shanthi M.E., Ph.D.,** Head of the Department, for her constant support. We also express our sincere thanks to our project coordinator **Ms.C.Sharmila M.E.,** Assistant Professor, for her constant encouragement for the development of our project. We take immense pleasure to express our heartfelt thanks to our supervisor**, Ms.M.Sangeetha M.E.,** for her valuable guidance for our project.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ML	-	Machine Learning
BMI	-	Body Mass Index
AUC	-	Area under the ROC Curve
RF	-	Random Forest
GBM	-	Gradient Boosting Machine
LGBM	-	Light GBM
LR	-	Logistic Regression
GBDT	-	Gradient Boosted Decision Tree
GBC	-	Gradient Boosting Classifier
CV	-	Cross Validation
DT	-	Decision Tree
RSCV	-	Random Search CV
GPU	-	Graphical Processing Unit
CPU	-	Central Processing Unit
RAM	-	Random Access Memory

CHAPTER 1

INTRODUCTION

1.1 MACHINE LEARNING

Computational methods are employed in the study of machine learning to convert empirical data into practical models. Machine learning emerged from the fields of classical statistics and artificial intelligence. The term "machine learning" refers to a system's ability to acquire and incorporate knowledge through extensive observation, as well as to develop and extend itself by acquiring new information rather than having it preprogrammed into it. Intelligent tutors use machine learning techniques to learn new teaching methods, assess students' skills, and learn more about them. They improve their teaching by repeatedly observing how students behave and making generalizations about the subject or student.

1.2 MACHINE LEARNING TECHNIQUES

To solve various problems, deep machine techniques are used. Some of them include,

- Learning Without Supervision
- Learning Under Supervision

1.2.1 Supervised Learning

Machines are educated using appropriately "labeled" training data, and then utilizing that data to anticipate the output, is known as supervised learning. The labeled data refers to input data that has already been assigned the appropriate output. In supervised learning, the training data that is given to the computers serves as the supervisor, instructing them on how to correctly predict the output. It employs the same idea that a pupil would learn under a teacher's guidance. The method of supervised learning involves giving the machine learning model the right input data as well as the output data. Finding a mapping function to link the input variable (x) with the output variable is the goal of a supervised learning algorithm (y). With the help of supervised learning, the model can predict the output on the basis of prior experiences.

1.2.2 Unsupervised Learning

Unsupervised learning, also known as unsupervised machine learning, analyses and clusters unlabeled datasets using machine learning techniques. Without the need for human intervention, these algorithms uncover hidden patterns or data groupings. Because of its capacity to detect similarities and contrasts in data, it is a perfect solution for exploratory data analysis, cross-selling techniques, consumer segmentation, and picture identification.

1.3 MACHINE LEARNING ALGORITHMS AND FLOW DIAGRAMS

The various Machine learning architectures are

- Random forest
- Logistic Regression
- Decision tree
- Gradient Boosting Classifier
- LightGBM

1.3.1 Random Forest

Popular machine learning algorithm Random Forest is a part of the supervised learning methodology. It can be applied to ML issues involving both classification and regression. It is built on the idea of ensemble learning, which is a method of integrating various classifiers to address difficult issues and enhance model performance. Random Forest, as the name implies, is a classifier that uses a number of decision trees on different subsets of the provided dataset and averages them to increase the dataset's predictive accuracy. Instead of depending on a single decision tree, the random forest uses forecasts from each tree and predicts the result based on the votes of the majority of predictions. The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

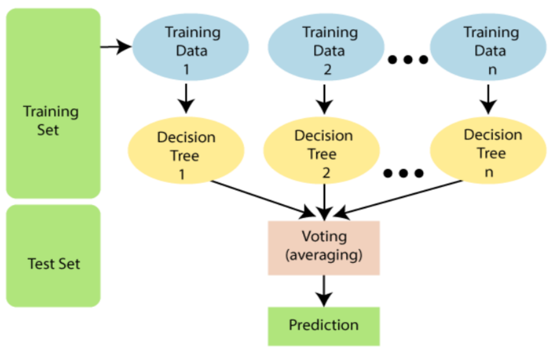


Figure 1.1.Random forest

1.3.2 Logistic regression

One of the most often used Machine Learning algorithms, within the category of Supervised Learning, is logistic regression. Using a predetermined set of independent factors, it is used to predict the categorical dependent variable. In a categorical dependent variable, the output is predicted via logistic regression. As a result, the result must be a discrete or categorical value. Rather than providing the exact values of 0 and 1, it provides the probabilistic values that fall between 0 and 1. It can be either Yes or No, 0 or 1, true or false, etc.

With the exception of how they are applied, logistic regression and linear regression are very similar. While logistic regression is used to solve classification difficulties, linear regression is used to solve regression problems. In logistic regression, fit a "S" shaped logistic function, which predicts two maximum values, rather than a regression line (0 or 1).

The logistic function's curve shows the possibility of several things, including whether or not the cells are malignant, whether or not a mouse is obese depending on its weight, etc. Because it can classify new data using both continuous and discrete datasets, logistic regression is a key machine learning approach.

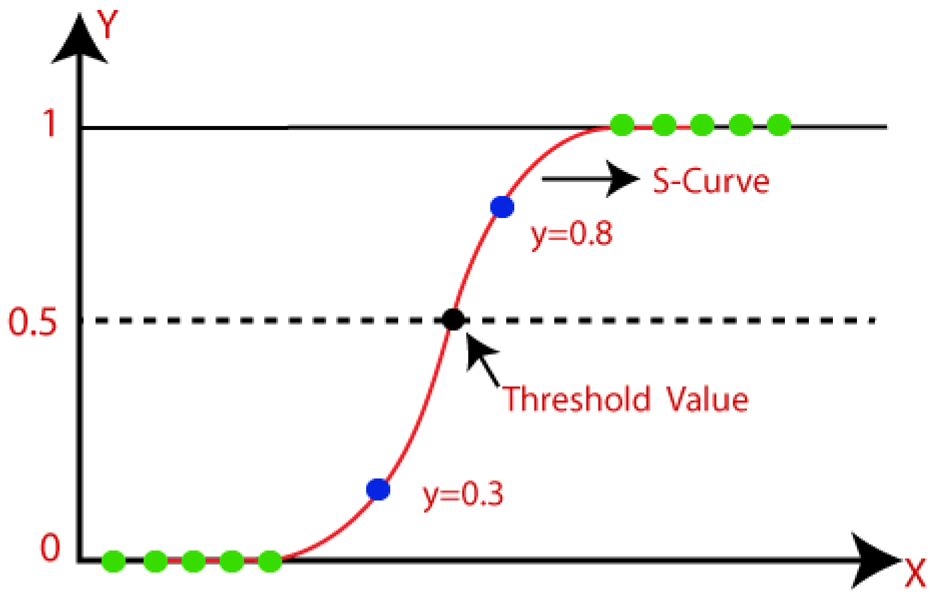


Figure 1.2.Logistic regression

1.3.3 Decision tree

A supervised learning method called a decision tree can be used to solve classification and regression problems, but it is typically favored for doing so. It is a tree-structured classifier, where internal nodes stand in for a dataset's features, branches for the decision-making process, and each leaf node for the classification result.

The Decision Node and Leaf Node are the two nodes of a decision tree. While Leaf nodes are the results of decisions and do not have any more branches, Decision nodes are used to create decisions and have numerous branches. The given dataset's features are used to execute the test or make the decisions. It is a graphical depiction for obtaining all feasible answers to a choice or problem based on predetermined conditions.

It is known as a decision tree because, like a tree, it begins with the root node and grows on subsequent branches to form a structure resembling a tree. The CART algorithm, which stands for Classification and Regression Tree algorithm, is used to construct a tree.

A decision tree only poses a question and divides the tree into subtrees according to the response (Yes/No).

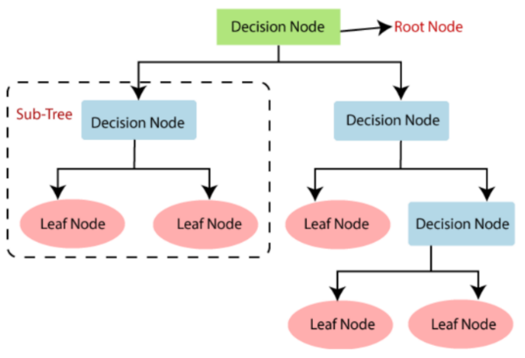


Figure 1.3.Decision tree

1.3.4 Gradient Boosting Classifier

Gradient boosting classifiers are a type of machine learning technique that combines several weak learning models to generate a powerful predictive model. When performing gradient boosting, decision trees are commonly employed. Gradient boosting models are gaining popularity due to their ability in classifying complicated datasets, and they have recently won a number of Kaggle data science challenges. Scikit-Learn, a Python machine learning toolkit, offers several gradient boosting classifier implementations, including XGBoost. In this post, we'll look at the theory underlying gradient boosting models/classifiers, as well as two distinct approaches to classification with gradient boosting classifiers in Scikit-Learn.

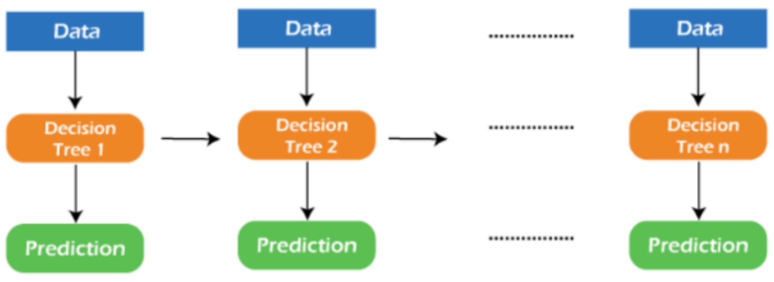


Figure 1.4.Gradient Boosting Classifier 4

1.3.5 Light GBM

The Light Gradient Boosting Machine is a more sophisticated version of the

Gradient Boosting Machine due to its effectiveness and quick speed. It can easily handle enormous volumes of data, unlike GBM and XGBM. It is inappropriate for data points with a small number, on the other hand. The nodes of the tree develop more quickly leaf-wise than level-wise under light GBM. In addition, the primary node in light GBM is divided into two secondary nodes before choosing which of the secondary nodes to split. Which of two nodes has the greatest loss determines this secondary node split.

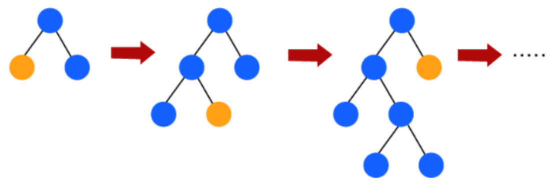


Figure 1.5.Light GBM

1.4 DATA PREPROCESSING

Data preprocessing, a subset of data preparation, refers to any type of processing performed on raw data in order to prepare it for further processing. It has traditionally been a critical first stage in the data mining process. Data preparation approaches have lately been modified for training machine learning and AI models as well as conducting inferences against them.

Data preprocessing converts data into a format that can be processed more readily and effectively in data mining, machine learning, and other data science operations. To achieve reliable findings, the techniques are typically used at the earliest phases of the machine learning and AI development pipeline.

Preprocessing data can be done using a variety of tools and techniques, such as the following:

- Sampling, which chooses a representative subset from a large population of data; transformation, which modifies raw data to create a single input.
- Denoising, which eliminates noise from data; imputation, which creates statistically relevant data for missing values.

- Normalization, which arranges data for easier access.
- Feature extraction, which isolates a relevant feature subset that is significant in a particular context.

1.5 RANDOM SEARCH CROSS VALIDATION

Machine learning models contain hyperparameters that must be set in order for the model to be customized to your dataset. The general effects of hyperparameters on a model are frequently understood, but determining how to optimally set a hyperparameter and combinations of interacting hyperparameters for a given dataset can be difficult. A better technique would be to objectively search different values for model hyperparameters and select a subset that results in a model that performs best on a particular dataset. It is known as hyperparameter optimization or hyperparameter tuning, and is supported by the scikit-learn Python machine learning toolkit.

CHAPTER 2

LITERATURE REVIEW

Sung Hoon Noh [1] included 1035 patients, with 520 receiving adjuvant capecitabine and oxaliplatin and 515 receiving observation. In the intention-to-treat population, the median follow-up for this research was 624 months (IQR 54–70). In the adjuvant capecitabine plus oxaliplatin group, 139 (27%) patients had disease-free survival events compared to 203 (39%) patients in the observation group (stratified hazard ratio [HR] 058; 95% CI 047-072; p00001). The estimated 5-year disease-free survival in the adjuvant capecitabine + oxaliplatin group was 68% (95% CI 63-73) versus 53% (47-58) in the observation alone group. By the clinical cutoff date, 103 patients (20%) in the adjuvant capecitabine and oxaliplatin group had died, compared to 141 patients (27%) in the observation group (stratified HR 066, 95% CI 051-085; p=0015). The estimated 5-year overall survival in the adjuvant capecitabine and oxaliplatin group was 78% (95% CI 74-82) versus 69% (64-73) in the observation group. After the primary analysis, no adverse event data were collected.

Tahmassebi, Amirhessam[3], Machine learning with mpMRI demonstrated stable performance, as evidenced by mean classification accuracies for RCB class prediction (AUC, 0.86) and DSS prediction (AUC, 0.92) based on XGBoost, and RFS prediction (AUC, 0.83) with logistic regression. When compared to other classifiers, the XGBoost classifier had the most steady performance with high accuracy. Changes in lesion size, complete pattern of shrinkage, and mean transit time on DCE-MRI; minimum ADC on DWI; and peritumoral edema on T2-weighted imaging were the most important features for predicting RCB class. Volume distribution, mean plasma flow, and mean transit time; DCE-MRI lesion size; and minimum, maximum, and mean ADC with DWI were the most important features for predicting RFS. The most relevant features for prediction of DSS were as follows: lesion size, volume distribution, and mean plasma flow on DCE-MRI, and maximum ADC with DWI.

Migita K [10], The mean BMI at the time of surgery was 22.5 kg/m2 (standard deviation, 3.3 kg/m2). According to the BMI subgroup, 73 (11.4%) patients were underweight, 431 (67.6%) were normal weight, and 134 (21%) were overweight. The 5-year overall survival (OS) rate for underweight patients was 66.6%, 81.3% for normal weight patients, and 79.9% for overweight patients (P = 0.001). The OS rate in underweight patients was significantly lower than in normal weight and overweight patients with stage I disease, and it was also lower than in normal weight individuals with stage II and III disease. Being underweight was revealed to be an independent predictor of OS in the multivariate analysis, but not in patients with stage II and III illness.

J. Kulig [11], Four hundred and ninety-two (25%) of the patients in 1992 were overweight. Greater BMI was linked with higher rates of postoperative cardiac complications (16% vs 12%, P = 0.001) and intra-abdominal abscess (6.9% vs 2.9%, P = 0.001). Other problems and fatality rates, however, remained unaffected. Overweight patients had a substantially longer median disease-specific survival (36.7 months, 95% CI 29.0-44.4) than those with BMI 25 kg/m2 (25.7 months, 95% CI 23.2-28.1; P = 0.003). The reduced frequency of patients with T3 and T4 tumors, metastatic lymph nodes, distant metastases, and non-curative resections contributed to these discrepancies. Age, degree of infiltration, lymph node metastases, distant metastases, and residual tumor type were found as independent predictive variables in a Cox proportional hazards model.

Lo, SS [4], Ones over the age of 75 had a worse overall survival rate than younger patients. However, there was no statistically significant difference in disease-related survival between the two groups. Recurrence was reported in 21 patients, with lymph node status being the most relevant predictor. Lymph node metastases occurred in 54 patients (11.3%), with 12 (4.4%) from mucosal tumors and 42 (20.3%) from submucosal tumors. Our patients had no lymph node metastases when the size of the mucosal tumor was less than 1 cm.

Tokunaga, M [8], Patients in the H-BMI group had more advanced illness. The 5-year survival rate in the H-BMI group was considerably higher than in the N-BMI group (81.5% vs 74.1%, respectively; P.001). Postoperative mortality was 1% in both groups (P =.482), however postoperative morbidity was 22% in the H-BMI group and 19% in the

N-BMI group (P =.007). Overweight, age, gender, surgical procedure, histology, operation year, pT, and pN were identified as independent predictive variables in multivariate analysis. Overweight was found to be an independent predictive factor in patients with pT1 and pN0 in subset analyses of pT and pN stages.

Kruhlikava, I [9], The median age at operation was 65 years (range: 27-84 years), with 207 (72.6%) men. Patients with the lowest BMI and those who were obese appeared to have a higher frequency of mild problems. Anastomotic leakage occurred in less than 10% of patients and was distributed evenly among the groups, as were the other significant problems. After controlling for potential confounders, there were no differences in 1-, 2-, or 5-year survival rates between the four body mass index groups. The five-year survival rates were 31.8%, 28.7%, 27.9%, and 26.1% for the four body mass index groups, respectively.

Eom, B. W [15], Age, tumor size, lymphovascular invasion, depth of invasion, and metastatic lymph nodes were found to be important predictive variables for overall survival in multivariate analyses. The concordance index in the external validation was 0.831 (95% confidence interval, 0.784-0.878), and the Hosmer-Lemeshow chi-square statistic was 3.92 (P = 0.917). Based on a multicenter database, a nomogram was created and validated to predict 5-year overall survival after curative resection of gastric cancer. This nomogram is useful for counseling patients and organizing follow-up appointments in general hospitals.

Alarcon-Ruiz, C. A [16], The poll was completed by 13,360 people. In the key variables, the response rate ranged from 99.8 to 100%. Patient satisfaction was inversely linked with waiting time (aOR: 0.98, 95% CI: 0.97-0.99), though the aOR was lower among those who reported a waiting time of 90 minutes (aOR: 0.92, 95% CI: 0.89-0.96). Patient satisfaction was directly associated with consultation time (aOR: 1.59, 95% CI: 1.26-2.01), while the aOR was higher among those who reported a consultation length of 15 minutes (aOR: 2.31, 95% CI: 1.66-3.21).

Wu, B [2], Recurrence was seen in 30 of 245 patients (12.24%) who had finished their follow-up. The median time between surgery and recurrence diagnosis was 28 months (range 3-188). Hematogenous recurrence was twice as common (3.27%) as peritoneal recurrence (1.63%). There were 9 cases of gastric stump cancer (3.67%), 7 cases of anastomotic recurrence (2.86%), and 4 cases of lymphatic recurrence (1.63%). Older patients (60 years old) had a higher rate of recurrence than younger ones (P 0.05). Lymph node metastasis and depth of invasion were found as risk variables for recurrence in multivariate analysis. Within three years of surgery, all patients with a positive family history of cancer had a recurrence.

Bickenbach, K.A [6], 1125 (60.7%) of the overall population of 1,853 patients were overweight. Patients who were overweight had more proximal tumors and a lower T stage. From 2000 to 2007, accurate complication statistics were available for a subgroup of patients. A BMI of 25 was linked to more postoperative problems (47.9 vs. 35.8%, p 0.001). This was attributed mostly to an increase in wound infections (8.9 vs. 4.7%, p = 0.02) and anastomotic leaks (11.8 vs. 5.4%, p = 0.002). Higher BMI, total gastrectomy, and use of neoadjuvant chemotherapy were all related with greater wound infection and anastomotic leak, according to multivariate logistic regression analysis. Overweight individuals were less likely (73.3 vs. 79.2%, p = 0.047) to have acceptable lymph node staging. There was no difference in overall survival or disease-specific survival between the two groups.

CHAPTER 3

REQUIREMENT SPECIFICATIONS

3.1 HARDWARE REQUIREMENTS

Processor	:	Intel Core i5
RAM	:	4 GB
Hard Disk	:	1 TB
Keyboard	:	Standard 104 keys

3.2 SOFTWARE REQUIREMENTS

Operating System	:	Windows 10
Tool	:	Google Colab

3.3 SOFTWARE DESCRIPTION

3.3.1 Google Colab

Google Colab is a research tool for machine learning education and research. The most recent versions of Chrome, Firefox, and Safari were used for the most thorough testing of Collaboratory in most popular browsers.

Google Colab is a free cloud service that now includes free GPU support. It also supports the Total Processing Unit. Drive files are directly imported, mounted, and uploaded.

Colab is ideal for everything from improving your Python coding skills to working with deep learning libraries such as PyTorch, Keras, TensorFlow, and OpenCV. You can upload your personal Jupyter Notebooks, notebooks directly from GitHub, notebooks directly from Kaggle files, upload Kaggle files, download your notebooks, mount your Google Drive and use whatever is stored there, import most of your favorite directories, create notebooks in Colab, store notebooks, share notebooks, mount your Google Drive, and pretty much anything else you might want to be able to do.

Jupyter is the open source project on which Colaboratory is built. Colaboratory allows you to use and share Jupyter notebooks with others without having to download, install, or run anything on your own computer other than a browser. The most important feature that sets Colab apart from other free cloud services is that it offers GPU and is completely free. It is extremely simple to modify the default hardware (CPU to GPU or vice versa).

The inputs and outputs of an interactive session are contained in Colab documents, as well as additional text that accompanies but does not execute the code. Colab notebook files can thus serve as a complete computational record of a session, containing executable code as well as explanatory text, mathematics, and rich representations of resulting objects.

3.3.2 Applications of Google Colab

Google Colab is widely used in a variety of industries, including business and financial analytics, marketing and trading, and the stock market.

- Education and research
- Analysis of financial trends
- Genetic modification
- Exploration of space
- Machine Learning from Text Mining
- Predictive analytics is being used to meet future demand for commodities such as gold, oil, and petroleum products

3.4 Packages

The following python packages are used.

3.4.1 Sklearn

Sklearn - Scikit-learn (previously scikits.learn and also known as sklearn) is a free software machine learning package for Python. It includes support-vector machines, random forests, gradient boosting, k-means, and DBSCAN as classification, regression, and clustering techniques, and is designed to work with the Python numerical and scientific libraries NumPy and SciPy.

3.4.2 Numpy

NumPy is a Python library that adds support for huge, multi-dimensional arrays and matrices, as well as a vast number of high-level mathematical functions to operate on these arrays. Numeric, NumPy's forefather, was invented by Jim Hugunin with help from several other people. Travis Oliphant built NumPy in 2005 by heavily modifying Numeric and combining features from the competitor Numarray. NumPy is open-source software with numerous contributors.

3.4.3 Pandas

Pandas is a data manipulation and analysis software package created for the Python computer language. It provides data structures and functions for manipulating numerical tables and time series in particular. It is free software distributed under the BSD three-clause license. The name is derived from the word "panel data," which is an econometrics term for data sets that comprise observations for the same individuals over several time periods. Its name is an allusion to the term "Python data analysis."

CHAPTER 4

PROPOSED WORK

Data preprocessing, a component of data preparation, describes any type of processing performed on raw data to prepare it for another data processing procedure.

The dataset had 51 attributes. Attribute selection was performed to select the factors affecting recurrence and metastasis to reduce the mortality of gastric cancer. And the dataset had missing values which were filled during Data Cleansing.

4.1 DATA PREPROCESSING

4.1.1 Data Reduction

Data reduction is the process of reducing the amount of capacity required to store data. Data reduction can increase storage efficiency and reduce costs. Storage vendors will often describe storage capacity in terms of raw capacity and effective capacity, which refers to data after the reduction. From the 51 attributes, 12 attributes were selected which are Sex, age(year), weight(kg), height(cm), BMI(Kg/m^2), Operation time(minutes), Location, Tumor size(cm), chemotherapy, dissection, death, circumference. The target variable Recurrence is included.

4.1.2 Data Cleansing

The process of finding and repairing corrupt or inaccurate records from a record set, table, or database is known as data cleansing. It entails identifying incomplete, erroneous, inaccurate, or irrelevant sections of the data and then replacing, changing, or removing the filthy or coarse data.

To fill missing values, KNN Imputer was utilized, which provides Imputation for completing missing values using k-Nearest Neighbors.

Missing values in each sample are imputed using the mean value from the training set's n neighbors nearest neighbors. Two samples are close if the features that neither lacks are similar.

4.1.3 Training and Test dataset

The dataset was split into a training set and a test set in ratio of 4:1 using sci-kit's train_test_split method. It is used to split arrays or matrices into random train and test subsets. It is a Quick utility that wraps input validation and next(ShuffleSplit().split(X, y)) and application to input data into a single call for splitting (and optionally subsampling) data in a one liner.

4.2 TRAINING PROCESS

Random forest model and Gradient Boosted decision Tree were trained using RandomForestClassifier, GradientBoostingClassifier from sklearn.ensemble respectively. Linear Regression model was trained using LogisticRegression from sklearn.linear_model. The Decision Tree model was trained using DecisionTreeClassifier from sklearn.tree. The Light Gradient Boosting Machine was trained using the LGBMClassifier from lightgbm. All models were trained using the same training dataset obtained from the train_test_split of sklearn.

4.3 OPTIMIZATION

The Random Forest model was optimized using the Random Search Cross Validation optimizer. The scikit-learn Python open-source machine learning library provides techniques to tune model hyperparameters.

The hyperparameters that can be configured are:

- n_estimators = number of trees in the forest
- max_features = max number of features considered for splitting a node
- max_depth = max number of levels in each decision tree

- min_samples_split = min number of data points placed in a node before the node is split
- min_samples_leaf = min number of data points allowed in a leaf node
- bootstrap = method for sampling data points (with or without replacement)

The overall optimization process of the Random Forest model using Random Search CV may be summarized as below:

Step 1 : Define a search space as a bounded domain of hyperparameter values.

Step 2 : Randomly sample points in that domain.

Step 3 : Instantiate the random search and fit it like any Scikit-Learn model. Specify the number of iterations which is the number of different combinations to try and the number of folds to use for cross validation.

Step 4 : Get the best params and best model from the optimizer. To determine if random search yielded a better model, compare the base model with the best random search model.

4.4 FLOW DIAGRAM

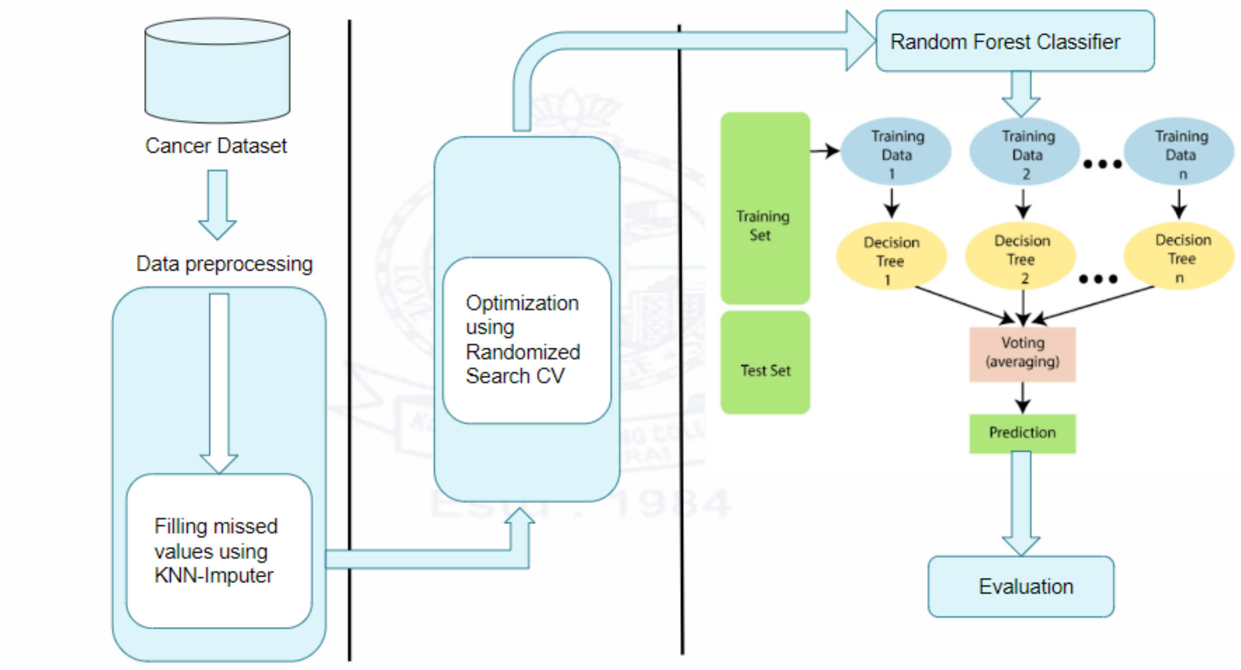


Figure 4.1 Flow Diagram

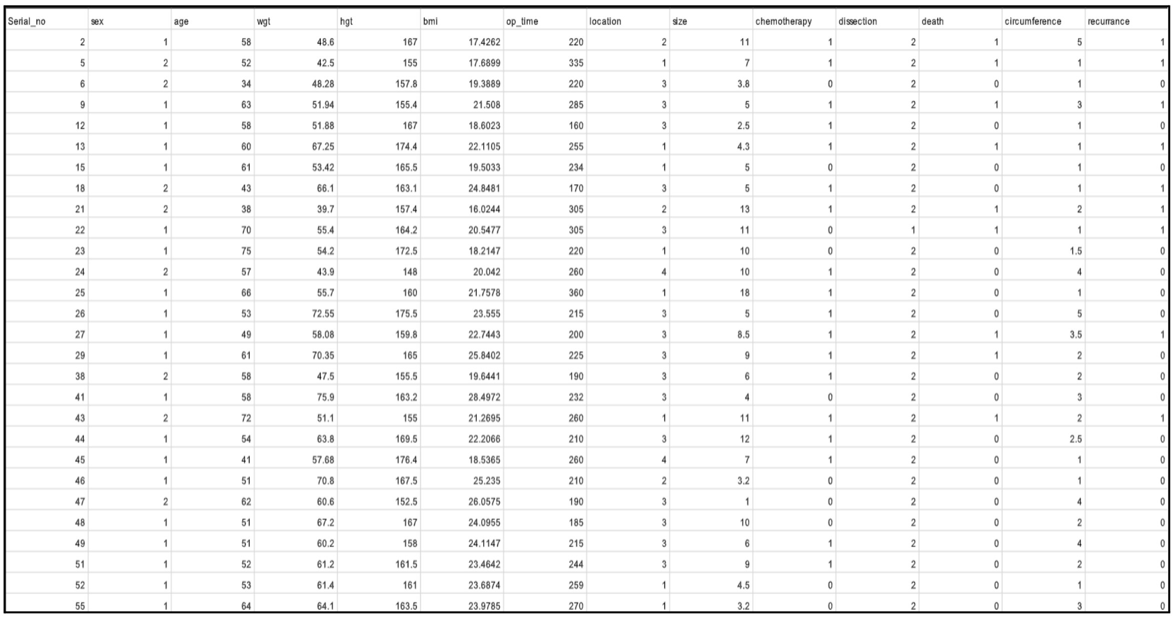
CHAPTER 5

RESULTS AND DISCUSSION

5.1 DATASET DESCRIPTION

The proposed work is evaluated on the patient’s clinicopathological dataset.It has 2012 patient records in dataset.We have selected 15 attributes to work with missing values are filled using KNN-Imputer using mean (K = 5).

5.1.1 SAMPLE DATASET



Serial_no	sex	age	wgt	hgt	bmi	op_time	location	size	chemotherapy	dissection	death	circumference	recurrance
2	1	58	48.6	167	17.4262	220	2	11	1	2	1	5	1
5	2	52	42.5	155	17.6899	335	1	7	1	2	1	1	1
6	2	34	48.28	157.8	19.3889	220	3	3.8	0	2	0	1	0
9	1	63	51.94	155.4	21.508	285	3	5	1	2	1	3	1
12	1	58	51.88	167	18.6023	160	3	2.5	1	2	0	1	0
13	1	60	67.25	174.4	22.1105	255	1	4.3	1	2	1	1	1
15	1	61	53.42	165.5	19.5033	234	1	5	0	2	0	1	0
18	2	43	66.1	163.1	24.8481	170	3	5	1	2	0	1	1
21	2	38	39.7	157.4	16.0244	305	2	13	1	2	1	2	1
22	1	70	55.4	164.2	20.5477	305	3	11	0	1	1	1	1
23	1	75	54.2	172.5	18.2147	220	1	10	0	2	0	1.5	0
24	2	57	43.9	148	20.042	260	4	10	1	2	0	4	0
25	1	66	55.7	160	21.7578	360	1	18	1	2	0	1	0
26	1	53	72.55	175.5	23.555	215	3	5	1	2	0	5	0
27	1	49	58.08	159.8	22.7443	200	3	8.5	1	2	1	3.5	1
29	1	61	70.35	165	25.8402	225	3	9	1	2	1	2	0
38	2	58	47.5	155.5	19.6441	190	3	6	1	2	0	2	0
41	1	58	75.9	163.2	28.4972	232	3	4	0	2	0	3	0
43	2	72	51.1	155	21.2695	260	1	11	1	2	1	2	1
44	1	54	63.8	169.5	22.2066	210	3	12	1	2	0	2.5	0
45	1	41	57.68	176.4	18.5365	260	4	7	1	2	0	1	0
46	1	51	70.8	167.5	25.235	210	2	3.2	0	2	0	1	0
47	2	62	60.6	152.5	26.0575	190	3	1	0	2	0	4	0
48	1	51	67.2	167	24.0955	185	3	10	0	2	0	2	0
49	1	51	60.2	158	24.1147	215	3	6	1	2	0	4	0
51	1	52	61.2	161.5	23.4642	244	3	9	1	2	0	2	0
52	1	53	61.4	161	23.6874	259	1	4.5	0	2	0	1	0
55	1	64	64.1	163.5	23.9785	270	1	3.2	0	2	0	3	0

Figure 5.1.1 Sample Dataset

5.1.2 CORRELATION ANALYSIS

Correlation describes the relationship between two or more variables. These variables could be input data features utilized to forecast our target variable.

Correlation is a statistical approach that determines how one variable moves/changes in connection to another. It offers us an idea of the strength of the relationship between the two variables. It is a bivariate analytic measure that describes the relationship between various variables. Most businesses benefit by expressing one subject in terms of its relationships with others.

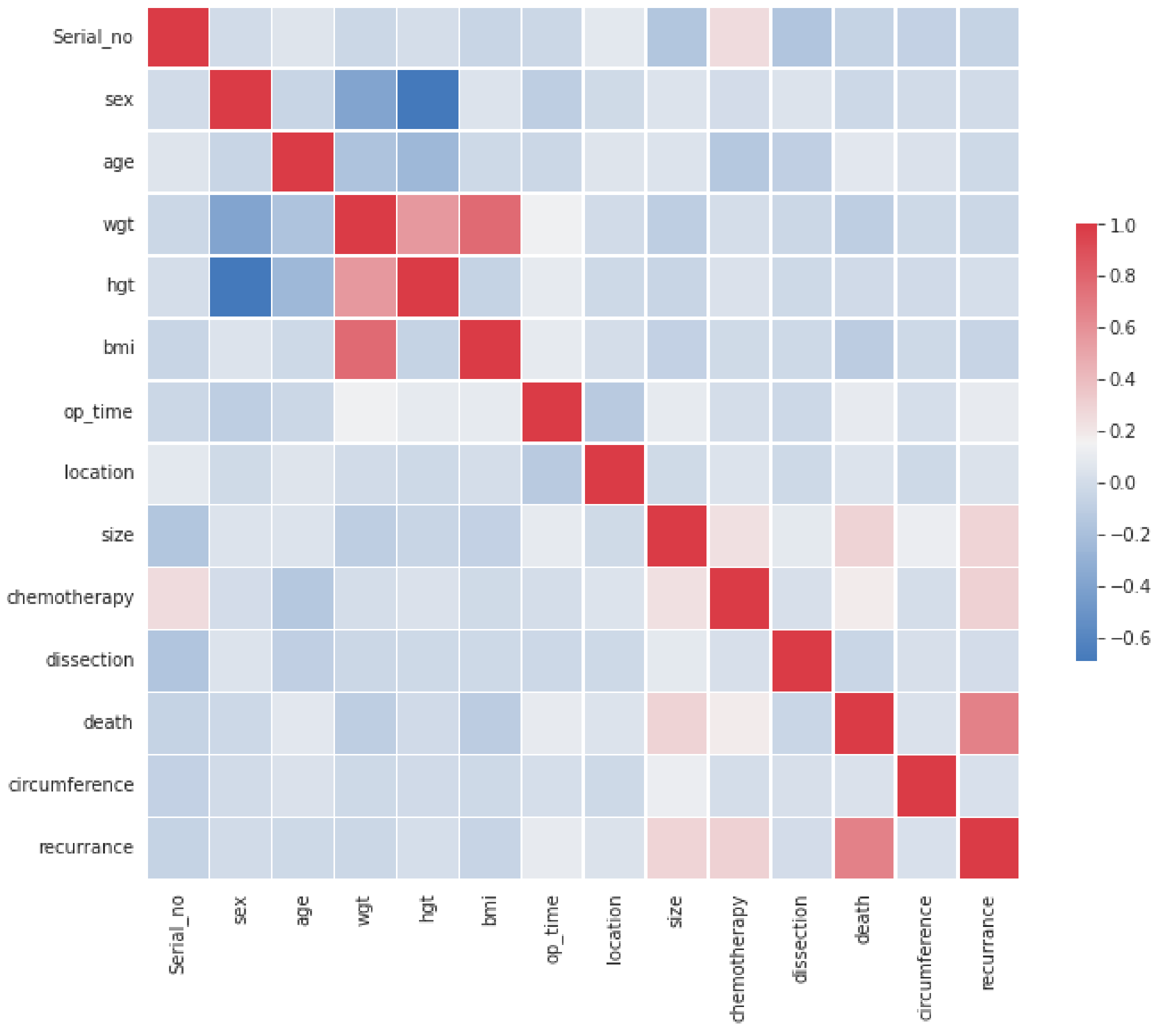


Figure 5.1.2 Correlogram

5.2 PARAMETER FOR EVALUATION

The performance is calculated in terms of following measures

Accuracy :

The percentage of accurate predictions among all predictions is how machine learning defines accuracy. This appears to be adequate as a measurement of a machine learning system's performance, but upon closer examination, it is found to be lacking.

Accuracy = (True Positive+True Negative) / (True Positive + False Positive + False Negative + True Negative)

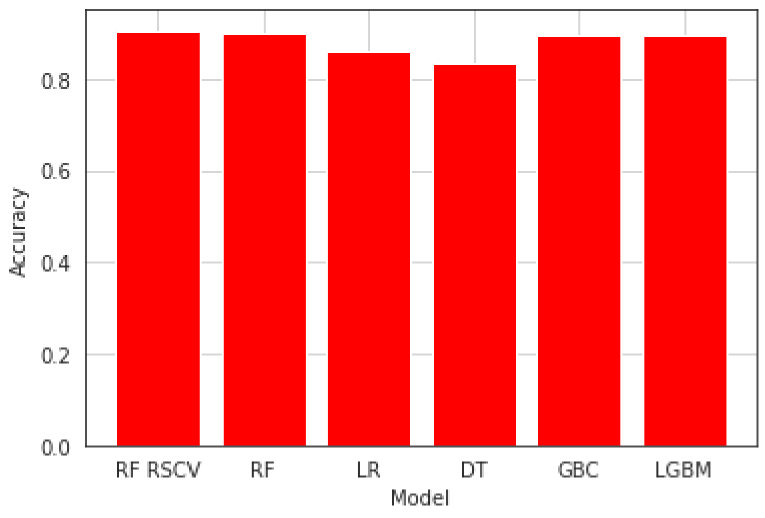


Figure 5.2.1. Accuracy Bar graph

Sensitivity :

The percentage of people without the disease who are correctly excluded by the test is referred to as specificity. These concepts are important in clinical practice for confirming or excluding disease during screening. An ideal test would have high specificity and sensitivity.

Specificity = True Negative / (True Negative + False Positive)

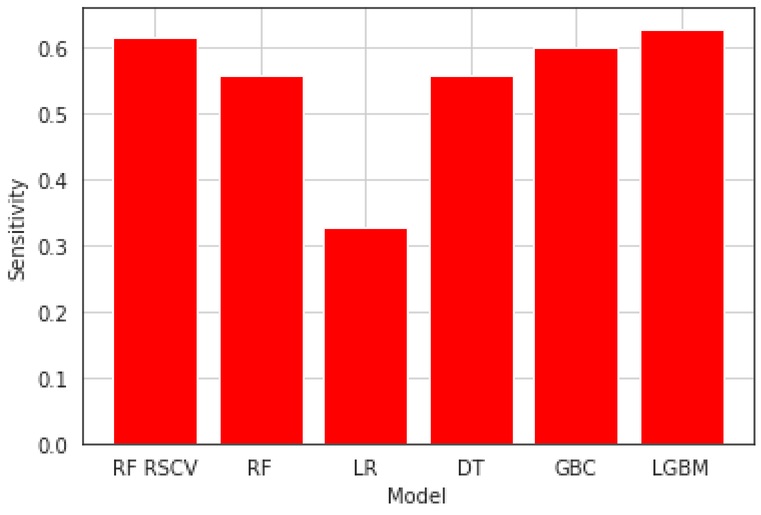


Figure 5.2.2.Sensitivity Bar graph

Specificity :

Specificity can be defined as the number of negatives returned by our ML model. It can be easily calculated it using the confusion matrix and the following formula.

Specificity=True Negative / (True Negative + False Positive)

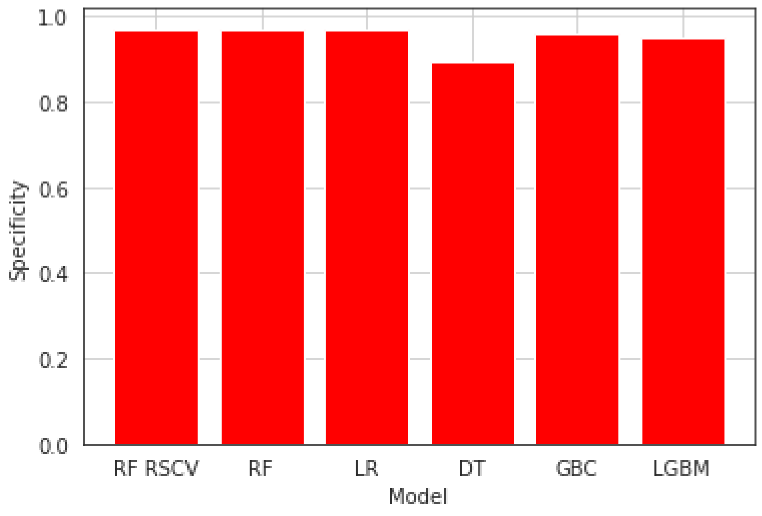


Figure 5.2.3. Specificity Bar graph

Precision:

Precision is one measure of a machine learning model's performance since it measures the accuracy of a positive prediction provided by the model. Precision is calculated by dividing the number of true positives by the total number of positive predictions.

Precision = TruePositives / (TruePositives + FalsePositives)

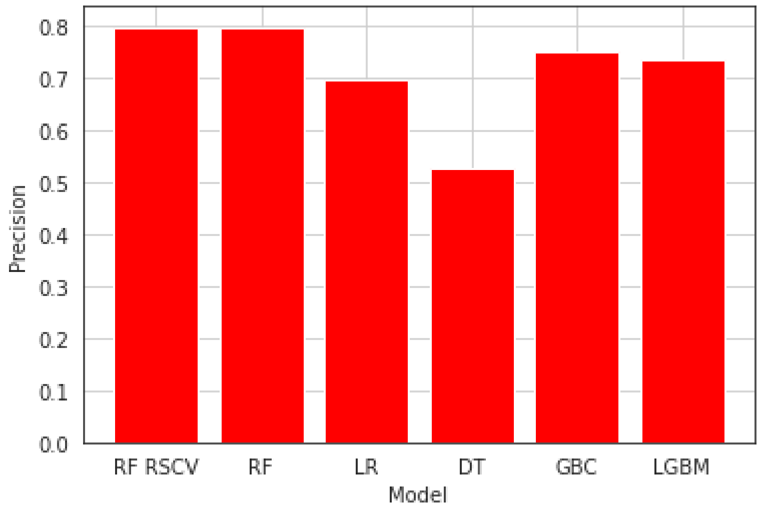


Figure 5.2.4 Precision Bar graph

AUC - ROC (Area under the receiver operating characteristic curve) :

AUC-ROC is a performance metric for classification issues that is based on varying threshold values. ROC stands for Receiver Operating Characteristic. ROC is a probability curve, and AUC is a measure of separability. In other words, the AUC-ROC metric will tell us about the model's ability to distinguish between classes. The model performs better when the AUC is higher.

Plotting TPR (True Positive Rate), also known as sensitivity or recall, vs. FPR (False Positive Rate), also known as 1-Specificity, at different threshold values can be used to create it mathematically.

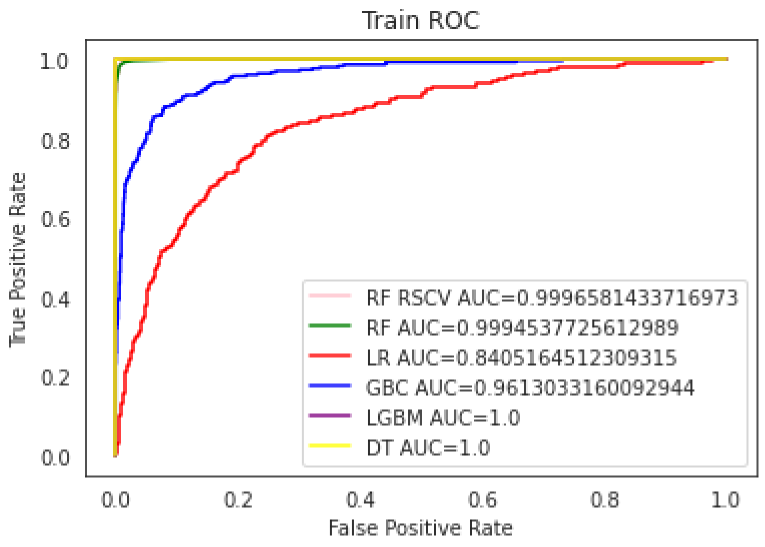


Figure 5.2.5 Train-ROC

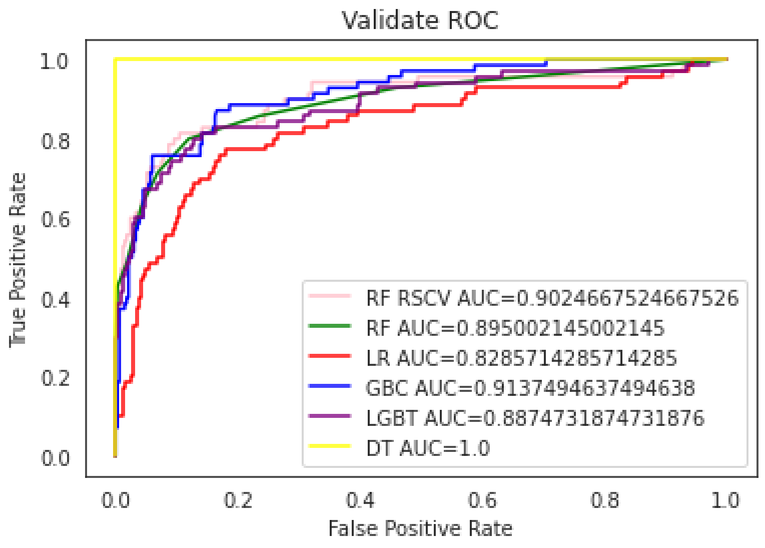


Figure 5.2.6 Validate-ROC

5.3 MODEL COMPARISON

S.No	Model	Accuracy	Specificity	Precision	Sensitivity
1	Random Forest with Random Search CV	90.57	96.70	79.63	61.43
2	Random Forest	89.83	97.00	79.59	55.71
3	Logistic Regression	85.86	97.00	69.7	32.86
4	Light Gradient Boosting Machine	89.58	95.20	73.33	62.86
5	Gradient Boosting Classifier	89.58	95.80	75.00	60.00
6	Decision Tree	83.62	89.49	52.70	55.71

Table 5.3.1 Model Comparisons

It is inferred from Table 5.3.1 that the training accuracy for all the architectures are compared. Among all the models Random Forest model has the highest accuracy. The highest accuracy obtained from this model is 89.83%.The second highest accuracy is 89.58% and that is for Light Gradient Boosting Machine and.Gradient Boosted Decision Tree. Random Forest with Random Search Cross Validation optimizer can achieve an accuracy of 90.57%.

5.4 ACCURACY WITH RANDOM SEARCH CROSS VALIDATION

The Random Forest model with Random search cross validation optimizer is displayed in the table above. The architecture displays 89.83 percent without Random search cross validation optimizer and 90.57 percent with Random search cross validation. For the Random Forest model with Random search cross validation, accuracy is greater. Random search CV is one of the most important ML optimization strategies.

CHAPTER 6

CONCLUSION AND FUTURE WORKS

In conclusion, machine learning can predict whether patients with stomach cancer would develop the disease again after surgery. Furthermore, the first four parameters influencing postoperative recurrence of gastric cancer were BMI, operation time, weight, and age. GNB, XGBoost, and Random Forest algorithms were found to be the most effective algorithms for predicting recurrent gastric cancer prediction in gastric cancer. Optimization techniques can be used to further improve the accuracy of the predictions made by the machine learning models.

In the coming years, more precise machine learning studies with more instances are required in order to identify the most accurate algorithm and maybe make individualized therapies applicable.

APPENDIX 1

```
from google.colab import drive
drive.mount("/drive/")
import pandas as pd
import matplotlib.pyplot as plt
dataset =  pd.read_csv("/drive/MyDrive/cancer-dataset.csv")
dataset["recurrence"]=dataset["recurrence"].astype("int")
X = dataset.drop(labels="recurrence",axis=1)
Y=pd.DataFrame(dataset["recurrence"],columns=["recurrence"])
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.20,random_state=42)
```

Random Forest

```
from sklearn.ensemble import RandomForestClassifier
forest_model=RandomForestClassifier(n_estimators=10)
forest_model.fit(X_train,Y_train)
forest_predictions = forest_model.predict(X_test)

from sklearn.metrics import accuracy_score
forest_acc = accuracy_score(Y_test,forest_predictions)
forest_acc

from sklearn import tree
from sklearn import metrics

def confusion_metrics (Y_test, forest_predictions):
# save confusion matrix and slice into four pieces

conf_matrix = metrics.confusion_matrix(Y_test, forest_predictions)
# Assigning columns names
cm_df = pd.DataFrame(conf_matrix,columns = ['Predicted Negative', 'Predicted Positive'],
            index = ['Actual Negative', 'Actual Positive'])

print(conf_matrix)
```

```
TP = conf_matrix[1][1]
TN = conf_matrix[0][0]
FP = conf_matrix[0][1]
FN = conf_matrix[1][0]
# calculate accuracy
conf_accuracy = (float (TP+TN) / float(TP + TN + FP + FN))

# calculate mis-classification
conf_misclassification = 1- conf_accuracy

# calculate the sensitivity
rand_conf_sensitivity = (TP / float(TP + FN))

# calculate the specificity
rand_conf_specificity = (TN / float(TN + FP))

# calculate precision
rand_conf_precision = (TN / float(TN + FP))
# calculate f_1 score
conf_f1 = 2 * ((rand_conf_precision * rand_conf_sensitivity) / (rand_conf_precision +
rand_conf_sensitivity))
print(f'Accuracy: {round(conf_accuracy,4)}')
print(f'Mis-Classification:{round(conf_misclassification,4)}')
print(f'Sensitivity: {round(rand_conf_sensitivity,4)}')
print(f'Specificity: {round(rand_conf_specificity,4)}')
print(f'Precision: {round(rand_conf_precision,4)}')
print(f'f_1 Score: {round(conf_f1,4)}')
return [round(conf_accuracy,4), rand_conf_sensitivity, rand_conf_specificity]

model_accuracy, rand_conf_sensitivity, rand_conf_specificity = confusion_metrics(Y_test,
forest_predictions)
```

RandomizedSearchCV - optimizer for random forest

```
from sklearn.model_selection import RandomizedSearchCV
import numpy as np
from pprint import pprint
```

```
# Number of trees in random forest
n_estimators = [int(x) for x in np.linspace(start = 200, stop = 2000, num = 10)]
# Number of features to consider at every split
max_features = ['auto', 'sqrt']
# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(10, 110, num = 11)]
max_depth.append(None)
# Minimum number of samples required to split a node
min_samples_split = [2, 5, 10]
# Minimum number of samples required at each leaf node
min_samples_leaf = [1, 2, 4]
# Method of selecting samples for training each tree
bootstrap = [True, False]
# Create the random grid
random_grid = {'n_estimators': n_estimators,
            'max_features': max_features,
            'max_depth': max_depth,
            'min_samples_split': min_samples_split,
            'min_samples_leaf': min_samples_leaf,
            'bootstrap': bootstrap}
pprint(random_grid)

rf_random = RandomizedSearchCV(estimator = forest_model, param_distributions =
random_grid, n_iter = 100, cv = 3, verbose=2, random_state=42, n_jobs = -1)
rf_random.fit(X_train,Y_train)
rf_random.best_params_
best_random = rf_random.best_estimator_
random_pred = best_random.predict(X_test)
random_accuracy = confusion_metrics(Y_test, random_pred)

print('Improvement of {:0.4f}%.'.format( 100 * (random_accuracy - model_accuracy) /
model_accuracy))
```

Logistic Regression

```
from sklearn.linear_model import LogisticRegression
log_model = LogisticRegression()
```

```
log_model.fit(X_train,Y_train)
log_predictions = log_model.predict(X_test)

log_acc, log_conf_sensitivity, log_conf_specificity =
confusion_metrics(Y_test,log_predictions)
log_acc
```

Decision Tree

```
from sklearn.tree import DecisionTreeClassifier
dec_model = DecisionTreeClassifier(random_state=42)
dec_model.fit(X_train,Y_train)
dec_predictions = dec_model.predict(X_test)

dec_acc, dec_conf_sensitivity, dec_conf_specificity =
confusion_metrics(Y_test,dec_predictions)
dec_acc
```

Gradient Boosting Classifier

```
from sklearn.ensemble import GradientBoostingClassifier
gbc_model = GradientBoostingClassifier(n_estimators=100,
learning_rate=1.0,max_depth=1, random_state=42)
gbc_model.fit(X_train,Y_train)
gbc_predictions = gbc_model.predict(X_test)

gbc_acc, gbc_conf_sensitivity,gbc_conf_specificity =
confusion_metrics(Y_test,gbc_predictions)
gbc_acc
```

LightGBM

```
from lightgbm import LGBMClassifier
lgbm_model = LGBMClassifier()
lgbm_model.fit(X_train,Y_train)
lgbm_predictions = lgbm_model.predict(X_test)

lgbm_acc, gbcl_conf_sensitivity, gbcl_conf_specificity =
confusion_metrics(Y_test,lgbm_predictions)
lgbm_acc
```

AUC - ROC

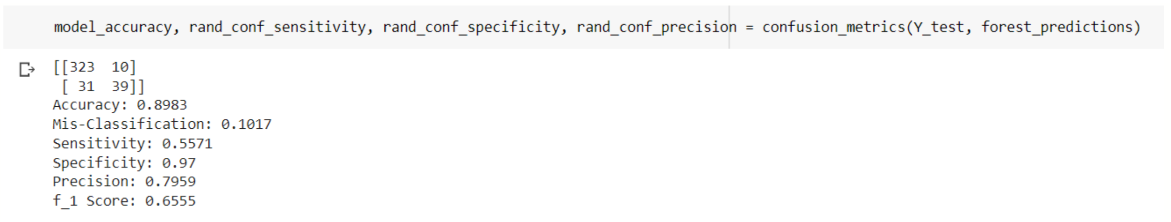
```
#define metrics

#Random Forest
forest_pred_proba = forest_model.predict_proba(X_train)[::,1]
forest_fpr, forest_tpr, _ = metrics.roc_curve(Y_train, forest_pred_proba)
forest_auc = metrics.roc_auc_score(Y_train, forest_pred_proba)

#random CV
rand_pred_proba = best_random.predict_proba(X_train)[::,1]
rand_fpr, rand_tpr, _ = metrics.roc_curve(Y_train, rand_pred_proba)
rand_auc = metrics.roc_auc_score(Y_train, rand_pred_proba)

#Logistic Regression
log_pred_proba = log_model.predict_proba(X_train)[::,1]
log_fpr, log_tpr, _ = metrics.roc_curve(Y_train, log_pred_proba)
log_auc = metrics.roc_auc_score(Y_train, log_pred_proba)

#Decision Tree
log_pred_proba = log_model.predict_proba(X_train)[::,1]
log_fpr, log_tpr, _ = metrics.roc_curve(Y_train, log_pred_proba)
log_auc = metrics.roc_auc_score(Y_train, log_pred_proba)

#Gradient Boosting Classifier
gbc_pred_proba = gbc_model.predict_proba(X_train)[::,1]
gbc_fpr, gbc_tpr, _ = metrics.roc_curve(Y_train, gbc_pred_proba)
gbc_auc = metrics.roc_auc_score(Y_train, gbc_pred_proba)

#Decision Tree
dec_pred_proba = dec_model.predict_proba(X_train)[::,1]
dec_fpr, dec_tpr, _ = metrics.roc_curve(Y_train, dec_pred_proba)
dec_auc = metrics.roc_auc_score(Y_train, dec_pred_proba)
```

```
#LightGBM
lgbm_pred_proba = lgbm_model.predict_proba(X_train)[::,1]
lgbm_fpr, lgbm_tpr, _ = metrics.roc_curve(Y_train, lgbm_pred_proba)
lgbm_auc = metrics.roc_auc_score(Y_train, lgbm_pred_proba)

#create ROC curve
plt.plot(rand_fpr,rand_tpr,label="RF RSCV AUC="+str(rand_auc),color="pink")
plt.plot(forest_fpr,forest_tpr,label="RF AUC="+str(forest_auc),color="green")
plt.plot(log_fpr,log_tpr,label="LR AUC="+str(log_auc),color="red")
plt.plot(gbc_fpr,gbc_tpr,label="GBC AUC="+str(gbc_auc),color="blue")
plt.plot(lgbm_fpr,lgbm_tpr,label="LGB AUC="+str(lgbm_auc),color="purple")
plt.plot(dec_fpr,dec_tpr,label="DEC AUC="+str(dec_auc),color="yellow")

plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.title("Train ROC")
plt.legend(loc="lower right")
plt.figure(figsize=(5,5))

#define metrics

#Random Forest
forest_pred_proba = forest_model.predict_proba(X_test)[::,1]
forest_fpr, forest_tpr, _ = metrics.roc_curve(Y_test, forest_pred_proba)
forest_auc = metrics.roc_auc_score(Y_test, forest_pred_proba)

#Random Forest with random search cv
rand_pred_proba = best_random.predict_proba(X_test)[::,1]
rand_fpr, rand_tpr, _ = metrics.roc_curve(Y_test, rand_pred_proba)
rand_auc = metrics.roc_auc_score(Y_test, rand_pred_proba)

#Logistic Regression
log_pred_proba = log_model.predict_proba(X_test)[::,1]
log_fpr, log_tpr, _ = metrics.roc_curve(Y_test, log_pred_proba)
log_auc = metrics.roc_auc_score(Y_test, log_pred_proba)
```

```
#Decision Tree
log_pred_proba = log_model.predict_proba(X_test)[::,1]
log_fpr, log_tpr, _ = metrics.roc_curve(Y_test, log_pred_proba)
log_auc = metrics.roc_auc_score(Y_test, log_pred_proba)

#Gradient Boosting Classifier
gbc_pred_proba = gbc_model.predict_proba(X_test)[::,1]
gbc_fpr, gbc_tpr, _ = metrics.roc_curve(Y_test, gbc_pred_proba)
gbc_auc = metrics.roc_auc_score(Y_test, gbc_pred_proba)

#LightGBM
lgbm_pred_proba = lgbm_model.predict_proba(X_test)[::,1]
lgbm_fpr, lgbm_tpr, _ = metrics.roc_curve(Y_test, lgbm_pred_proba)
lgbm_auc = metrics.roc_auc_score(Y_test, lgbm_pred_proba)

#create ROC curve
plt.plot(rand_fpr,rand_tpr,label="RF RSCV AUC="+str(rand_auc),color="pink")
plt.plot(forest_fpr,forest_tpr,label="RF AUC="+str(forest_auc),color="green")
plt.plot(log_fpr,log_tpr,label="LR AUC="+str(log_auc),color="red")
plt.plot(gbc_fpr,gbc_tpr,label="GBC AUC="+str(gbc_auc),color="blue")
plt.plot(lgbm_fpr,lgbm_tpr,label="LGB AUC="+str(lgbm_auc),color="purple")
plt.plot(dec_fpr,dec_tpr,label="DEC AUC="+str(dec_auc),color="yellow")

plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.title("Validate ROC")
plt.legend(loc="lower right")
plt.figure(figsize=(5,5))

#Accuracy Bar graph
plt.bar(["RF RSCV", "RF","LF","DT","GBC","LGBM"],[random_accuracy,
forest_acc,log_acc,dec_acc,gbc_acc,lgbm_acc],color="red",width=0.8)
plt.grid()
plt.show()

#Specificity Bar graph
plt.bar(["RF RSCV", "RF","LF","DT","GBC","LGBM"],[cv_spec ,rand_conf_specificity,
log_conf_specificity, dec_conf_specificity, gbc_conf_specificity,
gbcl_conf_specificity],color="red",width=0.8)
plt.grid()
plt.show()

#Sensitivity Bar graph
plt.bar(["RF RSCV", "RF","LF","DT","GBC","LGBM"],[cv_sens, rand_conf_sensitivity ,
log_conf_sensitivity , dec_conf_sensitivity , gbc_conf_sensitivity , gbcl_conf_sensitivity
],color="red",width=0.8)
plt.grid()
plt.show()
```

APPENDIX 2

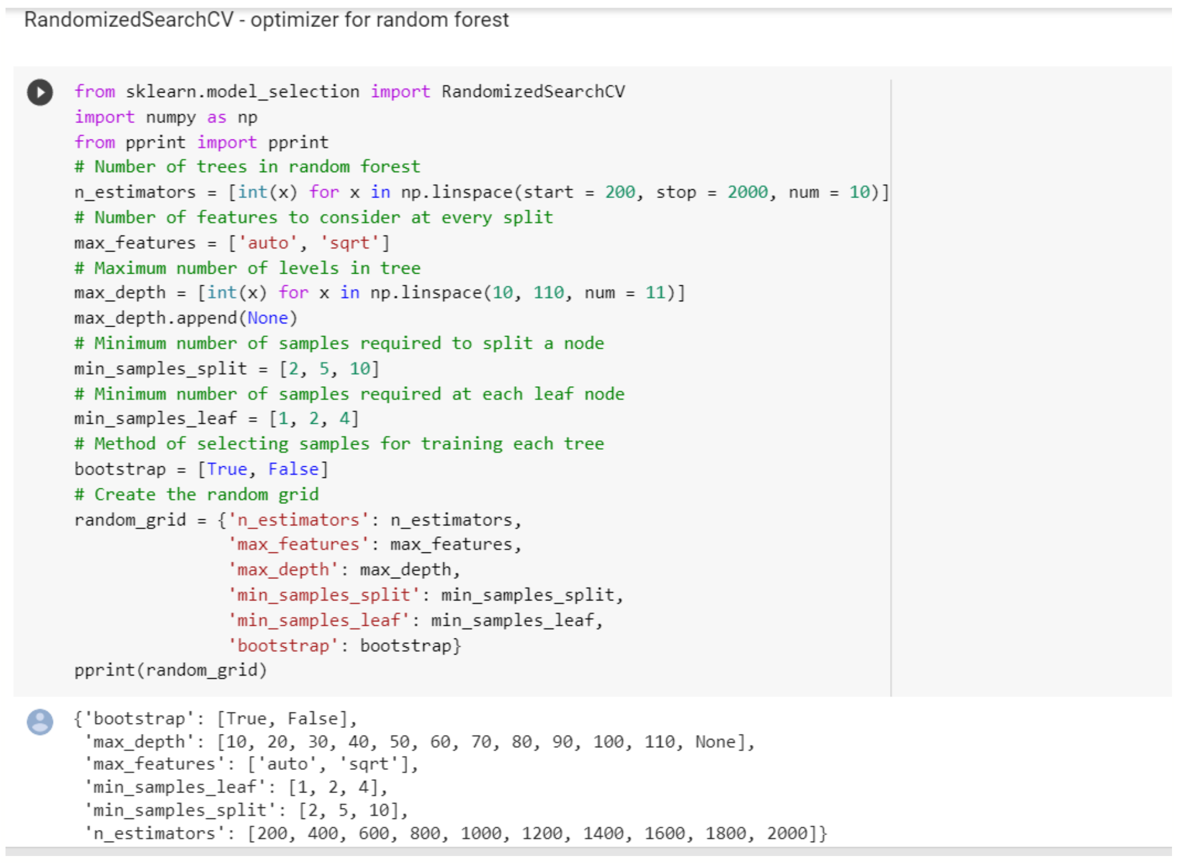
Random Forest



```
model_accuracy, rand_conf_sensitivity, rand_conf_specificity, rand_conf_precision = confusion_metrics(Y_test, forest_predictions)
```

```
[[323  10]
 [ 31  39]]
Accuracy: 0.8983
Mis-Classification: 0.1017
Sensitivity: 0.5571
Specificity: 0.97
Precision: 0.7959
f_1 Score: 0.6555
```

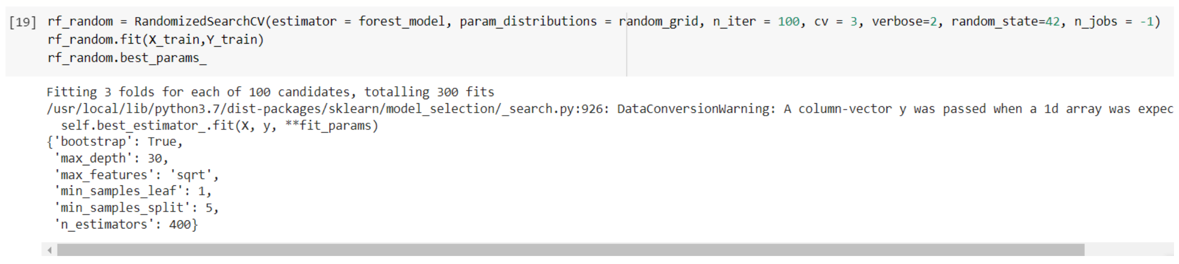
RandomizedSearchCV - optimizer for random forest



RandomizedSearchCV - optimizer for random forest

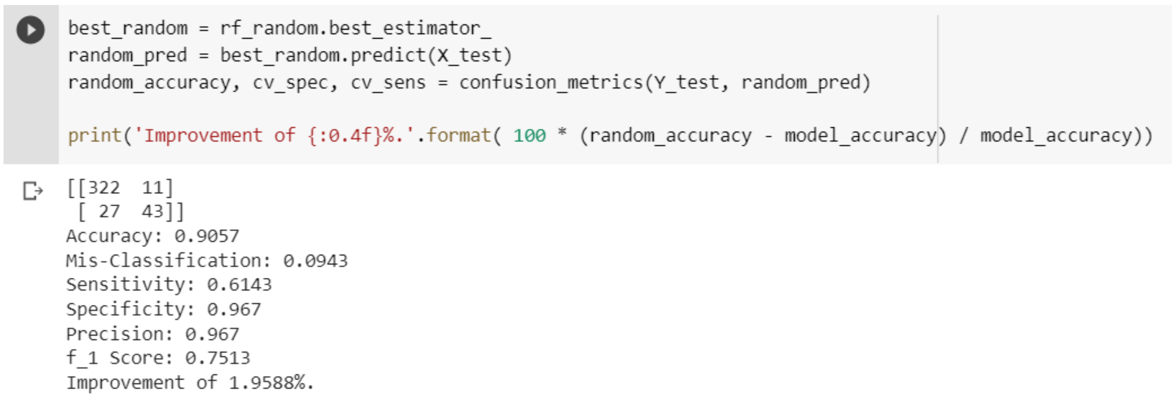
```
from sklearn.model_selection import RandomizedSearchCV
import numpy as np
from pprint import pprint
# Number of trees in random forest
n_estimators = [int(x) for x in np.linspace(start = 200, stop = 2000, num = 10)]
# Number of features to consider at every split
max_features = ['auto', 'sqrt']
# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(10, 110, num = 11)]
max_depth.append(None)
# Minimum number of samples required to split a node
min_samples_split = [2, 5, 10]
# Minimum number of samples required at each leaf node
min_samples_leaf = [1, 2, 4]
# Method of selecting samples for training each tree
bootstrap = [True, False]
# Create the random grid
random_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf,
               'bootstrap': bootstrap}
pprint(random_grid)
```

```
{'bootstrap': [True, False],
 'max_depth': [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, None],
 'max_features': ['auto', 'sqrt'],
 'min_samples_leaf': [1, 2, 4],
 'min_samples_split': [2, 5, 10],
 'n_estimators': [200, 400, 600, 800, 1000, 1200, 1400, 1600, 1800, 2000]}
```



```
[19] rf_random = RandomizedSearchCV(estimator = forest_model, param_distributions = random_grid, n_iter = 100, cv = 3, verbose=2, random_state=42, n_jobs = -1)
     rf_random.fit(X_train,Y_train)
     rf_random.best_params_
```

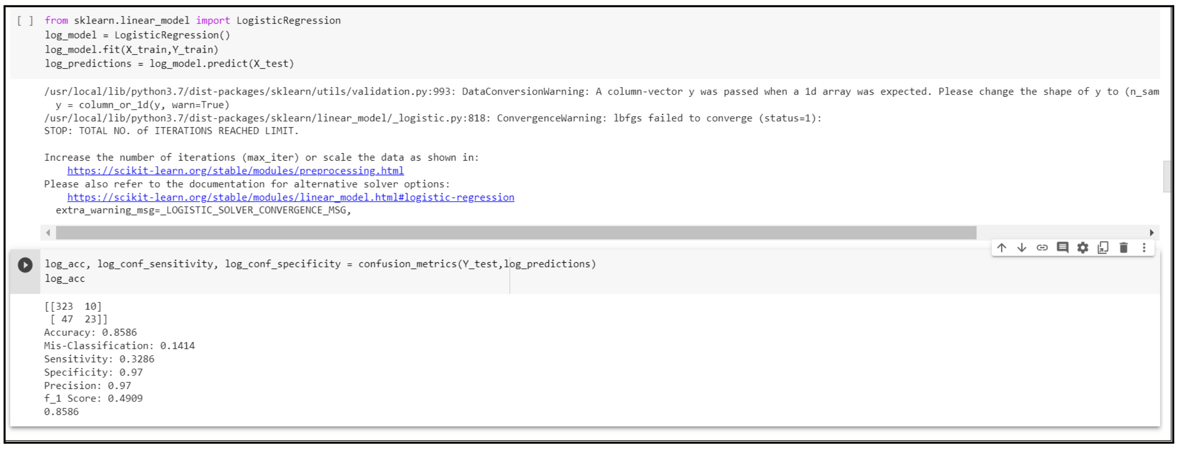
```
Fitting 3 folds for each of 100 candidates, totalling 300 fits
/usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_search.py:926: DataConversionWarning: A column-vector y was passed when a 1d array was expec
  self.best_estimator_.fit(X, y, **fit_params)
{'bootstrap': True,
 'max_depth': 30,
 'max_features': 'sqrt',
 'min_samples_leaf': 1,
 'min_samples_split': 5,
 'n_estimators': 400}
```



```
best_random = rf_random.best_estimator_
random_pred = best_random.predict(X_test)
random_accuracy, cv_spec, cv_sens = confusion_metrics(Y_test, random_pred)

print('Improvement of {:0.4f}%.'.format( 100 * (random_accuracy - model_accuracy) / model_accuracy))
```

```
[[322  11]
 [ 27  43]]
Accuracy: 0.9057
Mis-Classification: 0.0943
Sensitivity: 0.6143
Specificity: 0.967
Precision: 0.967
f_1 Score: 0.7513
Improvement of 1.9588%.
```

Logistic Regression



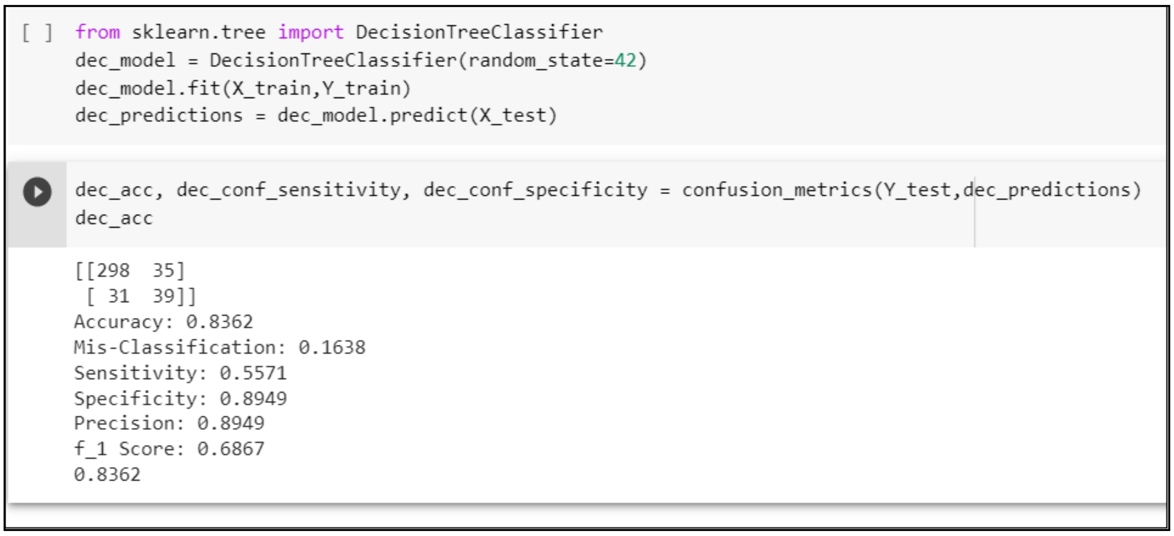
```
from sklearn.linear_model import LogisticRegression
log_model = LogisticRegression()
log_model.fit(X_train,Y_train)
log_predictions = log_model.predict(X_test)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:993: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_sam
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:818: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
```

```
log_acc, log_conf_sensitivity, log_conf_specificity = confusion_metrics(Y_test,log_predictions)
log_acc
```

```
[[323  10]
 [ 47  23]]
Accuracy: 0.8586
Mis-Classification: 0.1414
Sensitivity: 0.3286
Specificity: 0.97
Precision: 0.97
f_1 Score: 0.4909
0.8586
```

Decision Tree

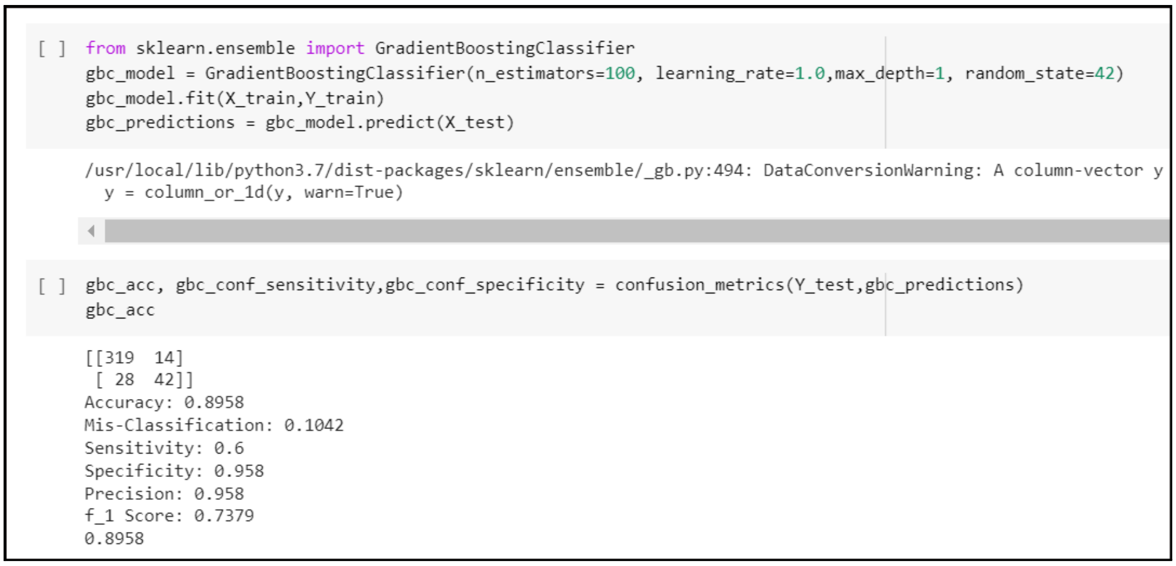


```
from sklearn.tree import DecisionTreeClassifier
dec_model = DecisionTreeClassifier(random_state=42)
dec_model.fit(X_train,Y_train)
dec_predictions = dec_model.predict(X_test)
```

```
dec_acc, dec_conf_sensitivity, dec_conf_specificity = confusion_metrics(Y_test,dec_predictions)
dec_acc
```

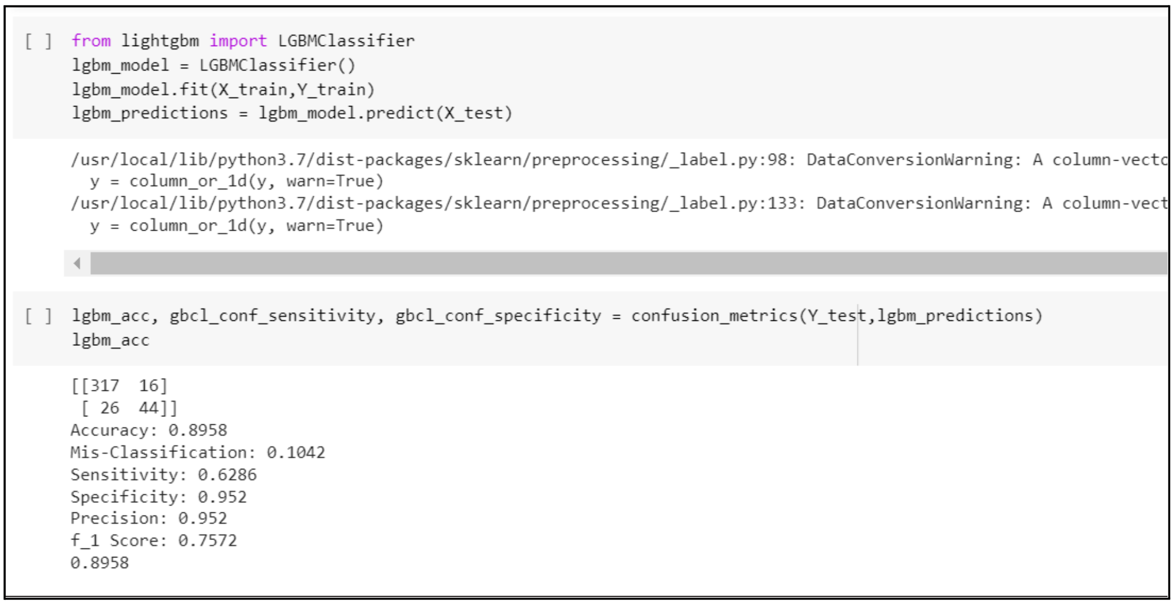
```
[[298  35]
 [ 31  39]]
Accuracy: 0.8362
Mis-Classification: 0.1638
Sensitivity: 0.5571
Specificity: 0.8949
Precision: 0.8949
f_1 Score: 0.6867
0.8362
```

Gradient Boosting Classifier



```
[ ] from sklearn.ensemble import GradientBoostingClassifier
    gbc_model = GradientBoostingClassifier(n_estimators=100, learning_rate=1.0,max_depth=1, random_state=42)
    gbc_model.fit(X_train,Y_train)
    gbc_predictions = gbc_model.predict(X_test)

    /usr/local/lib/python3.7/dist-packages/sklearn/ensemble/_gb.py:494: DataConversionWarning: A column-vector y
      y = column_or_1d(y, warn=True)

[ ] gbc_acc, gbc_conf_sensitivity,gbc_conf_specificity = confusion_metrics(Y_test,gbc_predictions)
    gbc_acc

    [[319  14]
     [ 28  42]]
    Accuracy: 0.8958
    Mis-Classification: 0.1042
    Sensitivity: 0.6
    Specificity: 0.958
    Precision: 0.958
    f_1 Score: 0.7379
    0.8958
```

LightGBM



```
[ ] from lightgbm import LGBMClassifier
    lgbm_model = LGBMClassifier()
    lgbm_model.fit(X_train,Y_train)
    lgbm_predictions = lgbm_model.predict(X_test)

    /usr/local/lib/python3.7/dist-packages/sklearn/preprocessing/_label.py:98: DataConversionWarning: A column-vecto
      y = column_or_1d(y, warn=True)
    /usr/local/lib/python3.7/dist-packages/sklearn/preprocessing/_label.py:133: DataConversionWarning: A column-vect
      y = column_or_1d(y, warn=True)

[ ] lgbm_acc, gbcl_conf_sensitivity, gbcl_conf_specificity = confusion_metrics(Y_test,lgbm_predictions)
    lgbm_acc

    [[317  16]
     [ 26  44]]
    Accuracy: 0.8958
    Mis-Classification: 0.1042
    Sensitivity: 0.6286
    Specificity: 0.952
    Precision: 0.952
    f_1 Score: 0.7572
    0.8958
```

REFERENCES

1. Sung Hoon Noh, Sook Ryun Park, Han-Kwang Yang, Hyun Cheol Chung, Ik-Joo Chung, Sang-Woon Kim, Hyung-Ho Kim, Jin-Hyuk Choi, Hoon-Kyo Kim, Wansik Yu, Jong Inn Lee, Dong Bok Shin, Jiafu Ji, Jen-Shi Chen, Yunni Lim, Stella Ha, Yung-Jue Bang, "Adjuvant capecitabine plus oxaliplatin for gastric cancer after D2 gastrectomy (CLASSIC): 5-year follow-up of an open-label, randomized phase 3 trial", The Lancet Oncology, Volume 15, Issue 12, 2014, Pages 1389-1396, ISSN 1470-2045, https://doi.org/10.1016/S1470-2045(14)70473-5.

2. Wu, B., Wu, D., Wang, M. & Wang, G. “Recurrence in patients following curative resection of early gastric carcinoma”. J. Surg. Oncol. 98, 411–414. https://doi.org/10.1002/jso.21133 (2008).

3. Tahmassebi, Amirhessam; Wengert, Georg J.; Helbich, Thomas H.; Bago-Horvath, Zsuzsanna MD; Alaei, Sousan; Bartsch, Rupert; Dubsky, Peter; Baltzer, Pascal; Clauser, Paola; Kapetas, Panagiotis; Morris, Elizabeth A.; Meyer-Baese, Anke; Pinker, Katja MD, "Impact of Machine Learning With Multiparametric Magnetic Resonance Imaging of the Breast for Early Prediction of Response to Neoadjuvant Chemotherapy and Survival Outcomes in Breast Cancer Patients". Investigative Radiology: February 2019 - Volume 54 - Issue 2 - p 110-117 doi: 10.1097/RLI.0000000000000518

4. Lo, SS., Wu, CW., Chen, JH. et al. “Surgical Results of Early Gastric Cancer and Proposing a Treatment Strategy”. Ann Surg Oncol 14, 340–347 (2007). https://doi.org/10.1245/s10434-006-9077-x

5. S. Moriguchi, Y. Maehara, D. Korenaga, K. Sugimachi, Y. Nose, "Risk factors which predict pattern of recurrence after curative surgery for patients with advanced gastric cancer", Surgical Oncology, Volume 1, Issue 5, 1992, Pages 341-346, ISSN 0960-7404, https://doi.org/10.1016/0960-7404(92)90034-I.

6. Bickenbach, K.A., Denton, B., Gonen, M. et al. “Impact of Obesity on

Perioperative Complications and Long-term Survival of Patients with Gastric Cancer". Ann Surg Oncol 20, 780–787 (2013). https://doi.org/10.1245/s10434-012-2653-3

7. Dhar D, K, Kubota H, Tachibana M, Kotoh T, Tabara H, Masunaga R, Kohno H, Nagasue N: "Body Mass Index Determines the Success of Lymph Node Dissection and Predicts the Outcome of Gastric Carcinoma Patients". Oncology 2000;59:18-23. doi: 10.1159/000012131.

8. Tokunaga, M., Hiki, N., Fukunaga, T. et al. "Better 5-Year Survival Rate Following Curative Gastrectomy in Overweight Patients". Ann Surg Oncol 16, 3245–3251 (2009). https://doi.org/10.1245/s10434-009-0645-8

9. Kruhlikava, I., Kirkegård, J., Mortensen, F. V. & Kjær, D. W. "Impact of body mass index on complications and survival after surgery for esophageal and gastro-esophageal-junction cancer". Scand. J. Surg. 106, 305–310. https://doi.org/10.1177/1457496916683097 (2017).

10. Migita, K., Takayama, T., Matsumoto, S. et al. "Impact of being underweight on the long-term outcomes of patients with gastric cancer". Gastric Cancer 19, 735–743 (2016). https://doi.org/10.1007/s10120-015-0531-y.

11. J. Kulig, M. Sierzega, P. Kolodziejczyk, J. Dadan, M. Drews, M. Fraczek, A. Jeziorski, M. Krawczyk, T. Starzynska, G. Wallner, "Implications of overweight in gastric cancer: A multicenter study in a Western patient population", European Journal of Surgical Oncology (EJSO), Volume 36, Issue 10, 2010, Pages 969-976, ISSN 0748-7983, https://doi.org/10.1016/j.ejso.2010.07.007. (2010).

12. Shoombuatong, W., Hongjaisee, S., Barin, F., Chaijaruwanich, J. & Samleerat, T. HIV-1 CRF01_AE "coreceptor usage prediction using kernel methods based on logistic model trees". Comput. Biol. Med. 42, 885–889 (2012).

13. Su, W. T., Nalini, S., Virapong, P., Chanin, N. & Watshara, S. PAAP: "A web server

for predicting antihypertensive activity of peptides.” Future Med. Chem. 10, 1749–1767 (2018).

14. Win, T. S. et al. HemoPred: “A web server for predicting the hemolytic activity of peptides”. Future Med. Chem. 9, 275–291 (2017).

15. Eom, B. W. et al. “Survival nomogram for curatively respected Korean gastric cancer patients: multicenter retrospective analysis with external validation”. PLoS ONE 10, e0119671. https://doi.org/10.1371/journal.pone.0119671 (2015).

16. Alarcon-Ruiz, C. A., Heredia, P. & Taype-Rondan, A. “Association of waiting and consultation time with patient satisfaction: secondary-data analysis of a national survey in Peruvian ambulatory care facilities”. BMC Health Serv. Res. 19, 439. https://doi. org/10.1186/s12913-019-4288-6 (2019)

17. Cuocolo, R., Caruso, M., Perillo, T., Ugga, L. & Petretta, M. “Machine learning in oncology: a clinical appraisal”. Cancer Lett. 481, 55–62 (2020).

18. Shimizu, H. & Nakayama, K. I. “Artificial intelligence in oncology”. Cancer Sci 111, 1452 (2020).