BREAST CANCER PREDICTION

-USING MACHINE LEARNING

Submitted by,

V.DESIKA

L.JOAN RESHMI

ABSTRACT

Breast Cancer 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 asignificant public health problem in today’s society , it occurs in women at any stage after puberty but with increasing rates in the later life. Therefore , Machine Learning technique can contribute to the process of prediction and early diagnosis of breast cancer. The Requirements of time is to develop the technique which gives minimum error to increase accuracy. Three Algorithm logistic regression , Random forest and KNN which predict the breast cancer outcomes have been compared in the project. All experiments are executed in JUPTYER PLATFORM. The purposed work can be used to predict the outcome of different technique and suitable technique can be used depending upon requirement. This project is carried out to predict the accuracy.

TABLE OF CONTENTS

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

ABBREVIATION	DEFINITION
SVM	Support Vector Machine
KNN	K-nearest Neighbors
MLP	Multilayer Perceptron
ANN	Artificial Neural Networks
RLP	Randomized Logistic Regression
BBO	Biogeography Based Optimization

CHAPTER 1

INTRODUCTION

The second major cause of women's death is breast cancer (after lung cancer). 246,660 of women's new cases of invasive breast cancer are expected to be diagnosed in the US during 2016 and 40,450 of women's death is estimated. Breast cancer is a type of cancer that starts in the breast. Cancer starts when cells begin to grow out of control. Breast cancer cells usually form a tumour that can often be seen on an x-ray or felt as a lump. Breast cancer can spread when the cancer cells get into the blood or lymph system and are carried to other parts of the body. The cause of Breast Cancer includes changes and mutations in DNA. There are many Different types of breast cancer and common ones include ductal carcinoma in situ (DCIS) and invasive carcinoma. Others, like phyllodes tumours and angiosarcoma are less common. There are many algorithms for classification of breast cancer outcomes. The side effects of Breast Cancer are – Fatigue, Headaches, Pain and numbness (peripheral neuropathy), Bone loss and osteoporosis. There are many algorithms for classification and prediction of breast cancer outcomes. The present paper gives a comparison between the performance of four classifiers: SVM, Logistic Regression , Random Forest and kNN which are among the most influential data mining algorithms. It can be medically detected early during a screening examination through mammography or by portable cancer diagnostic tool. Cancerous breast tissues change with the progression of the disease, which can be directly linked to cancer staging. The stage of breast cancer (I–IV) describes how far a patient's cancer has proliferated. Statistical indicators such as tumour size, lymph node metastasis, and distant metastasis and so on are used to determine stages. To prevent cancer from spreading, patients have to undergo breast cancer surgery, chemotherapy, radiotherapy and endocrine. The goal of the research is to identify and classify Malignant and Benign patients and intending how to parametrize our classification techniques hence to achieve high accuracy. We are looking into many datasets and how further Machine Learning algorithms can be used to characterize Breast Cancer. We want to reduce the error rates with maximum accuracy. 10-fold cross validation test which is a Machine Learning Technique is used in JUPYTER to evaluate the data and analyse data in terms of effectiveness and efficiency.

1.1 METHODS USED:

(1) Logistic Regression

Logistic regression was introduced by statistician DR Cox in 1958 and so predates the field of machine learning. It is a supervised machine learning technique, employed in classification jobs (for predictions based on training data). Logistic Regression uses an equation like Linear Regression, but the outcome of logistic regression is a categorical variable whereas it is a value for other regression models. Binary outcomes can be predicted from the independent variables.

The general workflow is:

(1) get a dataset

(2) train a classifier

(3) make a prediction using such classifier

(2) k-Nearest Neighbour (k-NN)

K-Nearest Neighbour is a supervised machine learning algorithm as the data given to it is labelled. It is a nonparametric method as the classification of test data point relies upon the nearest training data points rather than considering the dimensions (parameters) of the dataset.

ALGORITHM:

(1) Input the dataset and split it into a training and testing set.

(2) Pick an instance from the testing sets and calculate its distance with the training set.

(3) List distances in ascending order.

(4) The class of the instance is the most common class of the 3 first trainings instances (k=3).

(3) Random Forest

Random forest, like its name implies, consists of many individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction.

1.2.HEURISTIC:

The heuristic systematic model is used to investigate how ability, motivation, and heuristic message cues predict knowledge scores for individuals receiving messages written for different literacy levels about 3 environmental risk factors for breast cancer. The 3 risk factors were the roles of genetics, progesterone, and ingesting perfluorooctanoic acid in breast cancer risk. In this study, more than 4,000 women participated in an online survey. The results showed support for the hypotheses that ability (measured as education, number of science courses, and confidence in scientific ability) predict knowledge gain and that those individuals who presented with the lower literacy level message had significantly higher knowledge scores across all 3 message topics. There was little support for motivation or heuristic cues as direct predictors of knowledge gain across the 3 message topics, although they served as moderators for the perfluorooctanoic acid topic. The authors provide implications for health communication practitioners. This research was supported by the Breast Cancer and the Environment Research Centers grant number U01 ES012800 from the National Institute of Environmental Health Sciences, and the National Cancer Institute, National Institutes of Health, Department of Health and Human Services. Its contents are solely the responsibility of the authors and do not necessarily represent the official views of the National Institute of Environmental Health Sciences or of the National Cancer Institute/National Institutes of Health. Breast Cancer is the most affected disease present in women worldwide. 246,660 of women's new cases of invasive breast cancer are expected to be diagnosed in the U.S during 2016 and 40,450 of women's death is estimated. The development in Breast Cancer and its prediction fascinated. The UCI Wisconsin Machine Learning Repository Breast Cancer Dataset attracted as large patients with multivariate attributes were taken as sample set.

1.3. SYSTEM OVERVIEW:

It is a process of collecting and interpreting facts, identifying the problems, and decomposition of a system into its components.System analysis is conducted for the purpose of studying a system or its parts in order to identify its objectives. It is a problem solving technique that improves the system and ensures that all the components of the system work efficiently to accomplish their purpose. It is a process of planning a new business system or replacing an existing system by defining its components or modules to satisfy the specific requirements. Before planning, you need to understand the old system thoroughly and determine how computers can best be used in order to operate efficiently. The word System is derived from Greek word Systema, which means an organized relationship between any set of components to achieve some common cause or objective.

A system is "an orderly grouping of interdependent components linked together according to a plan to achieve a specific goal."

1.4. SCOPE OF THE PROJECT:

This work reports the impact of dominance-based filtering approach on performances of major state-of-the-art classifiers used in machine learning paradigm. WBCD database was utilized for RH, 5-fold and 10-fold cross validation protocols using sensitivity, specificity, accuracy and AUC parameters. ANN has emerged as the best classifier with classification accuracies of 98.9% for four dominant features (feature set #4), 99.6% for five dominant features (feature set #5) and 99.7% for all features (feature set #9) subjected to the 10-fold crossvalidation protocol. It is interesting to note that the accuracies and AUCs conceived by ANN classifier for four and five abridged features of dominance-based filtering technique are almost equivalent to the values for all nine features (feature set #9) with less than 2% error. There are some limitations of this work which can be looked out in future by researchers. As this empirical study was done in WBCD (original) dataset consists of narrow feature space (n =9), the reliability of the research can be increased by introducing a live breast- cancer dataset with a large number of feature vectors. Also, though dominance-based filtering technique is computationally fast, the feature rank is algorithm liberated. This can be prevented by adding an algorithm-relative block with proposed tactic for rank calculation. Furthermore, the proposed approach can be applied for other cancer datasets to test its generalized performance capability of the algorithm.

CHAPTER 2

SURVEY

Siyabend Turgut et al., "Microarray Breast Cancer Data Classification Using Machine Learning Methods" [IEEE 2018]

The paper uses microarray breast cancer data for classification of the patients using machine learning methods. In the first case, eight different machine learning algorithms are applied to the dataset and the results of classification were noted. Then in the second case, two different feature selection methods such as Recursive Feature Elimination (RFE) and Randomized Logistic Regression (RLR) were applied on the microarray breast cancer dataset and 50 features were chosen as stop criterion. Again, the same eight machine learning algorithms were applied on the modified dataset. The results of the classifications are compared with each other and with the results of the first case. The methods applied are SVM, KNN, MLP, Decision Trees, Random Forest, Logistic Regression, Ad boost and Gradient Boosting Machines. After applying the two different feature selection methods, SVM gave the best results. MLP is applied using different number of layers and neurons to examine the effect of the number of layers and neurons on the classification accuracy .

Varalatchoumy M et al., "Four Novel Approaches for Detection of Region of Interest in Mammograms - A Comparative Study" [ICISS 2017]

The paper compares Four Novel approaches used for detection of Region of Interest in Mammographic images based on database and Real time images. In Approach I histogram equalization and dynamic thresholding techniques were used for preprocessing. Region of InteresT (ROI) was partitioned from the preprocessed image by using particle swarm optimization and k- means clustering methods. In Approach II preprocessing was done using various morphological operations like erosion followed by dilation. For the identification of ROI, a modified approach of watershed segmentation was used. Approach III uses histogram equalization for preprocessing and an advanced level set approach for performing segmentation. Approach IV, which is considered to be the most efficient approach that uses different morphological operations and contrast limited adaptive histogram equalization for image preprocessing. A very novel algorithm was developed for detection of Region of Interest. Approaches I and II were applicable for Mammographic Image Analysis Society (MIAS) database images alone. Approaches III and IV were applicable for MIAS and Real time hospital images. The various graphs presented in the comparative study, clearly depicts that the novel approach that used a novel algorithm for detection of ROI is proved to be the most efficient, accurate and highly reliable approach that can be used by radiologists to detect tumors in MRM images.

Ammu P K et al., "Review on Feature Selection Techniques of DNAMicroarray Data" [IJCA 2013]

This paper reviews few major feature selection techniques employed in microarray data and points out the merits and demerits of various approaches. Feature selection from DNA microarray data is one of the most important procedures in bioinformatics. Biogeography Based Optimization (BBO) is an optimization algorithm which works on the basis of migration of species between different habitats and the process of mutation. Particle Swarm Optimization (PSO) is an algorithm which works on the basis of movement of particles in a search space. Redundancy based feature selection approaches can be used to remove redundant genes from the selected genes as the resultant gene set can achieve a better representation of the target class. A two-stage hybrid filter wrapper method where, in the first stage a subset of the original feature set is obtained by applying information gain as the filtering criteria. In the second stage the genetic algorithm is applied to the set of filtered genes. Gene selection based on dependency of features where the features are classified as independent, half dependent and dependent features. Independent features are those features that doesn't depend on any other features. Half dependent features are more relevant in correlation with other features and dependent features are fully dependent on other features.

Bing Lan Li et al., "Integrating spatial fuzzy clustering with level set methods for automated medical image segmentation" [ELSEVIER 2010]

A new Fuzzy Level Set algorithm is proposed in this paper to facilitate automated medical image segmentation. It can directly evolve from the initial segmentation by spatial fuzzy clustering where centroid and the scope of each subclass are estimated adaptively in order to minimize a pre-defined cost function. The controlling parameters of Level Set evolution are also estimated from the results of fuzzy clustering. The level set methods utilize dynamic variational boundaries for image segmentation. The new Fuzzy Level Set algorithm automates the initialization and parameter configuration of the level set segmentation, using spatial fuzzy clustering. It employs a Fuzzy-C means (FCM) with spatial restrictions to determine the approximate contours of interest in a medical image. Moreover, the Fuzzy Level Set algorithm is enhanced with locally regularized evolution. Such improvements facilitate level set manipulation and lead to more robust segmentation. Performance evaluation of the proposed algorithm was carried on medical images from different modalities. The results confirm its effectiveness for image segmentation.

CHAPTER 3

SYSTEM ANALYSIS

3.1. EXISTING SYSTEM:

Many existing models use the concept of mammography image dataset to build models and do predictions. A mammogram is an x-ray picture of the breast. It is also useful if you have a lump or other major sign of cancer. Screening mammography is the type of mammogram that checks patients when they don’t have any symptoms of the breast cancer. It helped in reducing the number of deaths from breast cancer among women of age from 40 to 70. But fewtumors cannot be spottedby a mammogram due to the position of the cancer or the thickness of the chest tissue. About 25 % of cancers in female agingfrom 40 to 49 are not detectable by a screening mammogram, compared to 10% in women older than 50. And that’s the major drawback of using dataset built on mammography image. There are some other models that do not use image dataset but uses the faulty or improper data to make predictions which often results to be wrong.

3.1.1. DISADVANTAGES OF EXISTING SYSTEM:

The major limitations of existing systems are:

1. Use of faulty dataset
2. Using less and unimportant features to determine output.
3. More man power
4. Consumes large volume of paper work.
5. Needs manual calculations.

3.2. PROPOSED SYSTEM:

The proposed system develops a classification and predictive model that will perform accurate classification grouping and prediction of Breast Cancer. This proposed approach will focus on prediction of the disease in the early stage by taking 10 real world value parameters for every cancer cell nucleus. A combination of learning algorithms of classification and ensemble learning are used to implement and develop the proposed model.

3.2.1. ADVANTAGES OF PROPOSED SYSTEM:

1. The accuracy will be more due to use proper dataset consisting of parameters of cell nucleus.

2. Proper use of features and normalization of the data.

3.Free from unnecessary tunning of model.

3.3. REQUIREMENTS SPECIFICATIONS:

3.3.1. HARDWARE REQUIREMENTS:

1. Processor	: Intel(R) Core(TM) i3-7100U CPU @ 2.40GHz 2.40 GHz
2. RAM	: 8.00 GB
3. System type	: 64-bit Operating System
4. Operating system:	Windows10

3.3.2. SOFTWARE REQUIREMENTS:

1. Jupyter Notebook.

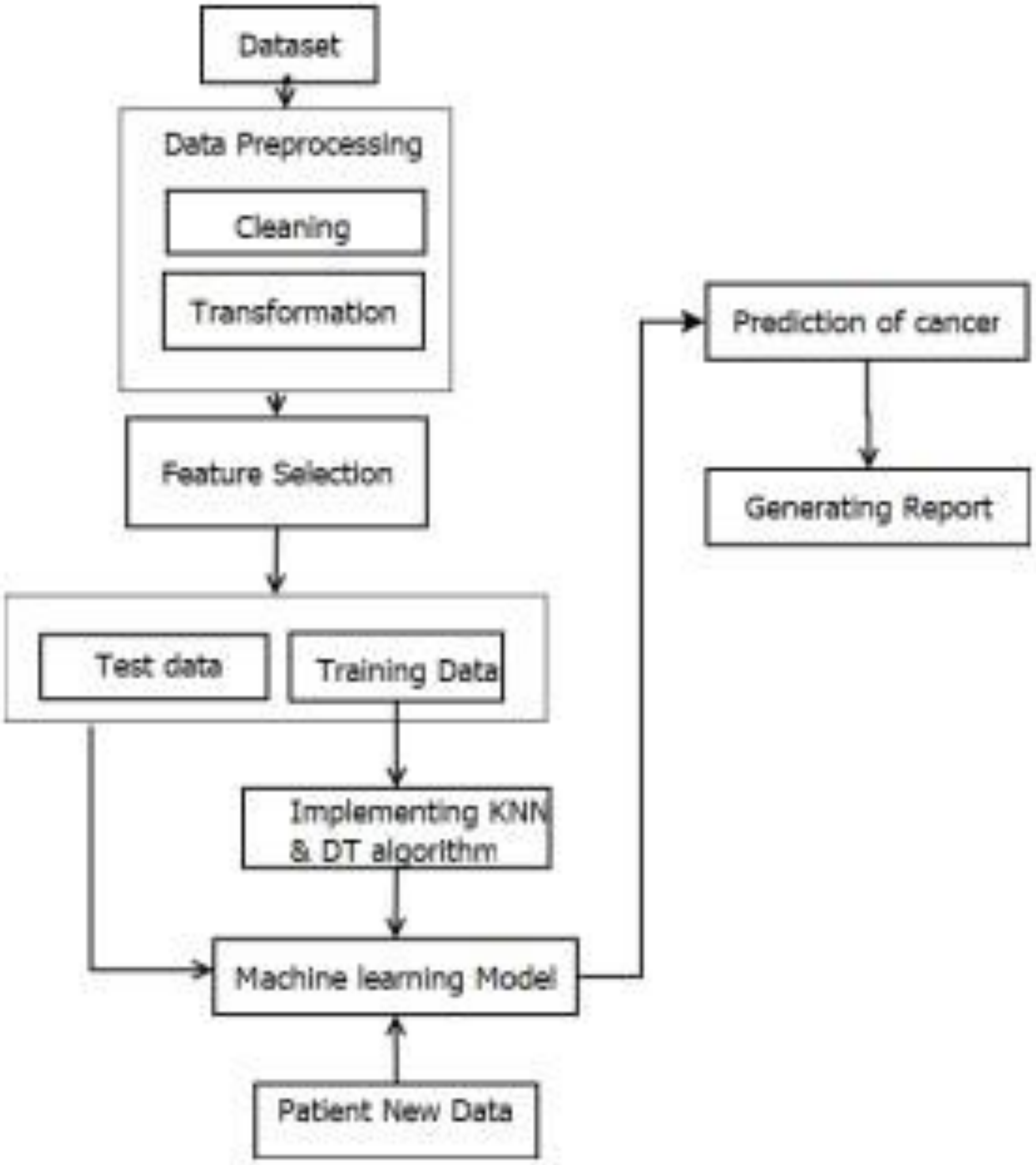
3.4. LANGUAGE SPECIFICATION:

Python leads the pack, with 57% of data scientists and machine learning developers using it and 33% prioritizing it for development. Little wonder, given all the evolution in the deep learning Python frameworks over the past 2 years, including the release of Tensor Flow and a wide selection of other libraries. Python is often compared to R, but they are nowhere near comparable in terms of popularity: R comes fourth in overall usage (31%) and fifth in prioritization (5%). R is in fact the language with the lowest prioritization -to-usage ratio among the five, with only 17% of developers who use it prioritising it. This means that in most cases R is a complementary language, not a first choice. The same ratio for Python is at 58%, the highest by far among the five languages, a clear indication that the usage trends of Python are the exact opposite to those of R. Not only is Python the most widely used language , it is also the primary choice for the majority of its users. C/C++ is a distant second to Python, both in usage (44%) and prioritization (19%). Java follows C/C++ very closely, while JavaScript comes fifth in usage, although with a slightly better prioritisation performance than R (7%). We asked our respondents about other languages used in machine learning, including the usual suspects of Julia, Scala , Ruby, Octave, MATLAB and SAS, but they all fall below the 5% mark of prioritisation and below 26% of usage. We therefore focused our attention on the top-5 language.

CHAPTER 4

SYSTEM DESIGN

4.1. SYSTEM ARCHITECTURE:



A Dataset is a collection of data. Load Dataset into the program. Data Pre-processing is a important step in machine learning process. Data pre processing is a technique that is used to convert the raw data into a clean dataset. Data is clean through process such as handle the missing value ,noisy data or resolving the inconsistancies in the data.

Data transformation is the process of converting data from one format or structure into another format or structure. It is primarily involves mapping how source data element will be changed for the destination. Machine learning uses so called features (i.e. variables or attributes) to generate predictive models.

Using a suitable combination of features is essential for obtaining high precision and accuracy. Because too many (unspecific) features pose the problem of overfitting the model, we generally want to restrict the features in our models to those, that are most relevant for the response variable we want to predict.

Using as few features as possible will also reduce the complexity of our models, which means it needs less time and computer power to run and is easier to understand.

Here select the features on the basis of Principal Component Analysis(PCA). After feature selection dataset is split into training data and test data. Apply training dataset to the K NN or DT algorithms. Applying the K-NN and DT algorithms generate the machine learning model.

Send test data to the model and first check the performance. Identify which algorithm is best for our system and then enter new test data to that model and predict the cancer stage.

CHAPTER 5

MODULE DESCRIPTION

5.1.MODULES

The system consists of the following modules.

1. Data Collection.
2. Data Exploration.
3. Data Visualization.
4. Data Filtering.
5. Train model.
6. Performance Analysis.
7. Features Selection.

6.1. MODULES DESCRIPTION

1. DATA COLLECTION:

The first phase we do is to collect the data that we are interested in collecting for pre-processing and to apply classification and Regression methods. Data pre-processing is a data mining technique that involves transforming raw data into an understandable format. Real world data is often incomplete, inconsistent, and lacking certain to contain many errors. Data pre-processing is a proven method of resolving such issues. Data pre-processing prepares raw data for further processing. For pre-processing we have used standardization method to pre-process the UCI dataset. This step is very important because the quality and quantity of data that you gather will directly determine how good your predictive model can be. In this case we collect the Breast Cancer samples which are Benign and Malignant. This will be our training data.

After collecting data, we need to know what are the shape of this dataset, Here we have attribute(property) called data.shape

For that we have 2 type of methods to show the shape of the datasets.

1. len(data.index), len(data.columns)

   data.shape

Both methods are giving us the same output, As you can see in the cells.

2. DATA EXPLORATION:

Data exploration is the first step of data analysis used to explore and visualize data to uncover insights from the start or identify areas or patterns to dig into more. Using interactive dashboards and point-and-click data exploration, users can better understand the bigger picture and get to insights faster. Starting with data exploration helps users to make better decisions on where to dig deeper into the data and to take a broad understanding of the business when asking more detailed questions later. With a user-friendly interface, anyone across an organization can familiarize themselves with the data, discover patterns, and generate thoughtful questions that may spur on deeper, valuable analysis. Data exploration and visual analytics tools build understanding, empowering users to explore data in any visualization. This approach speeds up time to answers and deepens users' understanding by covering more ground in less time. Data exploration is important for this reason because it democratizes access to data and provides governed self-service analytics. Furthermore, businesses can accelerate data exploration by provisioning and delivering data through visual data marts that are easy to explore and use.

3. DATA VISUALIZATION:

Data visualization is the practice of translating information into a visual context, such as a map or graph, to make data easier for the human brain to understand and pull insights from. The main goal of data visualization is to make it easier to identify patterns, trends and outliers in large data sets. The term is often used interchangeably with others, including information graphics, information visualization and statistical graphics .Data visualization is one of the steps of the data science process, which states that after data has been collected, processed and modeled, it must be visualized for conclusions to be made. Data visualization is also an element of the broader data presentation architecture (DPA) discipline, which aims to identify, locate, manipulate, format and deliver data in the most efficient way possible .Data visualization is important for almost every career. It can be used by teachers to display student test results, by computer scientists exploring advancements in artificial intelligence (AI) or by executives looking to share information with stakeholders. It also plays an important role in big data projects. As businesses accumulated massive collections of data during the early years of the big data trend, they needed a way to quickly and easily get an overview of their data. Visualization tools were a natural fit. Visualization is central to advanced analytics for similar reasons. When a data scientist is writing advanced predictive analytics or machine learning (ML) algorithms, it becomes important to visualize the outputs to monitor results and ensure that models are performing as intended. This is because visualizations of complex algorithms are generally easier to interpret than numerical outputs.

4. Data Filtering :

Data filtering in IT can refer to a wide range of strategies or solutions for refining data sets. This means the data sets are refined into simply what a user (or set of users) needs, without including other data that can be repetitive, irrelevant or even sensitive. Different types of data filters can be used to amend reports, query results, or other kinds of information results.Typically, data filtering will involve taking out information that is useless to a reader or information that can be confusing. Generated reports and query results from database tools often result in large and complex data sets. Redundant or impartial pieces of data can confuse or disorient a user. Filtering data can also make results more efficient.In some other cases, data filters work to prevent wider access to sensitive information. For example, a data filtering program could scrub Social Security numbers, credit card numbers and other identifiers from complex client data sets coming into an employee’s workstation or, even more importantly, onto his or her mobile device. With the "bring your own device" (BYOD) movement emerging within the business world, data filtering can solve some security problems related to the information that employees need to do their jobs.

Now, we have one categorical feature, so we need to convert it into numeric values using LabelEncoder from sklearn.preprocessing packages.

5. TRAIN MODEL:

A training model is a dataset that is used to train an ML algorithm. It consists of the sample output data and the corresponding sets of input data that have an influence on the output. The training model is used to run the input data through the algorithm to correlate the processed output against the sample output. The result from this correlation is used to modify the model. This iterative process is called “model fitting”. The accuracy of the training dataset or the validation dataset is critical for the precision of the model. Model training in machine language is the process of feeding an ML algorithm with data to help identify and learn good values for all attributes involved. There are several types of machine learning models, of which the most common ones are supervised and unsupervised learning. Supervised learning is possible when the training data contains both the input and output values. Each set of data that has the inputs and the expected output is called a supervisory signal. The training is done based on the deviation of the processed result from the documented result when the inputs are fed into the model. Unsupervised learning involves determining patterns in the data. Additional data is then used to fit patterns or clusters. This is also an iterative process that improves the accuracy based on the correlation to the expected patterns or clusters. There is no reference output dataset in this method.

6.PERFORMANCE ANALYSIS:

Machine learning model performance is relative and ideas of what score a good model can achieve only make sense and can only be interpreted in the context of the skill scores of other models also trained on the same data.

Perform Feature Standerd Scalling :
Standardize features by removing the mean and scaling to unit variance
The standard score of a sample x is calculated as:

$$z = (x - u) / s$$

7.FEATURES SELECTION:

The input variables that we give to our machine learning models are called features. Each column in our dataset constitutes a feature. To train an optimal model, we need to make sure that we use only the essential features. If we have too many features, the model can capture the unimportant patterns and learn from noise. The method of choosing the important parameters of our data is called Feature Selection. In this article titled 'Everything you need to know about Feature Selection', we will teach you all you need to know about feature selection.

Machine learning models follow a simple rule: whatever goes in, comes out. If we put garbage into our model, we can expect the output to be garbage too. In this case, garbage refers to noise in our data.

To train a model, we collect enormous quantities of data to help the machine learn better. Usually, a good portion of the data collected is noise, while some of the columns of our dataset might not contribute significantly to the performance of our model. Further, having a lot of data can slow down the training process and cause the model to be slower. The model may also learn from this irrelevant data and be inaccurate.

Feature Selection is the method of reducing the input variable to your model by using only relevant data and getting rid of noise in data.

It is the process of automatically choosing relevant features for your machine learning model based on the type of problem you are trying to solve. We do this by including or excluding important features without changing them. It helps in cutting down the noise in our data and reducing the size of our input data.

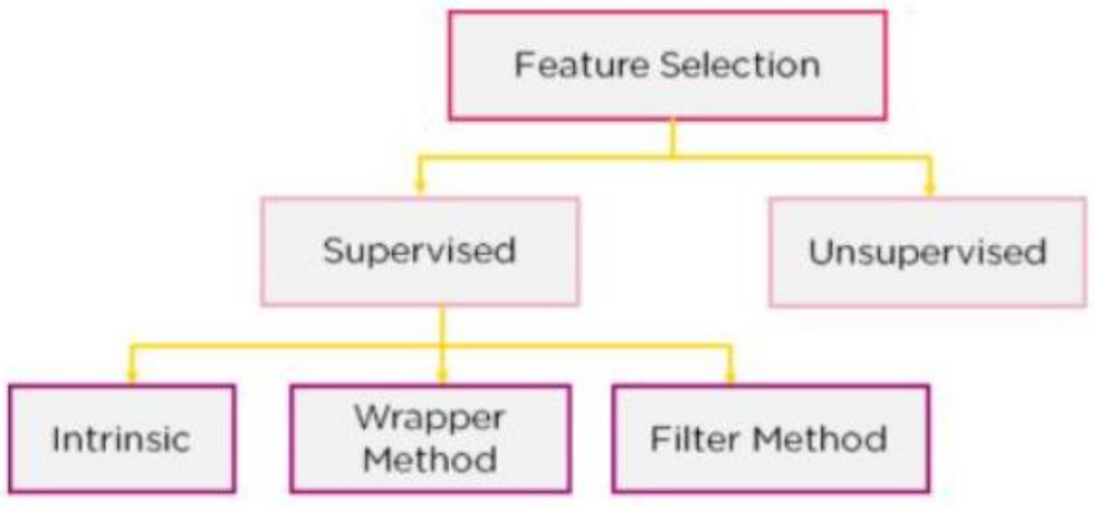


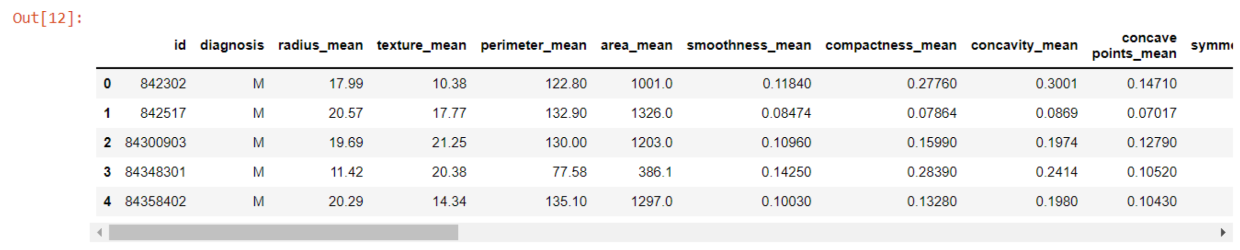
Figure 4: Feature Selection Models

1. Supervised Models: Supervised feature selection refers to the method which uses the output label class for feature selection. They use the target variables to identify the variables which can increase the efficiency of the model
2. Unsupervised Models: Unsupervised feature selection refers to the method which does not need the output label class for feature selection. We use them for unlabelled data.

APPENDIX 1

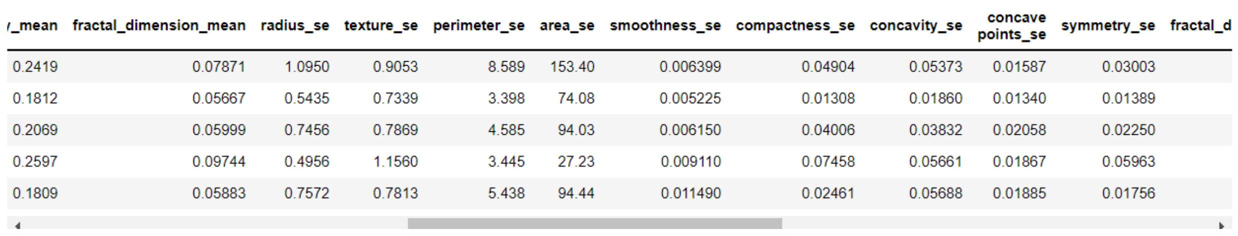
SAMPLE CODE

```
import numpy as np # linear algebra

import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

pd.options.display.max_columns = 100

import os

for dirname, _, filenames in os.walk('/kaggle/input'):

    for filename in filenames:

        print(os.path.join(dirname, filename))

data = pd.read_csv("C://Users//LAWRENCE//Documents//data.csv")

# Cell 1

len(data.index), len(data.columns)

OUT: (569, 32)
data.shape()
OUT: (569, 32)
data.head()
```

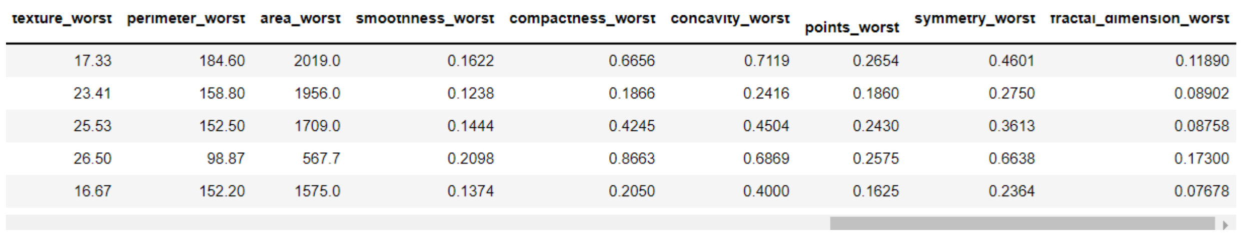


Out[12]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symm
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	

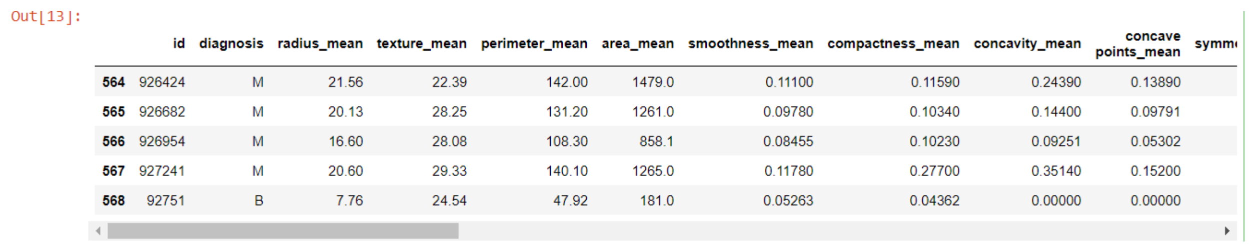


_mean	fractal_dimension_mean	radius_se	texture_se	perimeter_se	area_se	smoothness_se	compactness_se	concavity_se	concave points_se	symmetry_se	fractal_d
0.2419	0.07871	1.0950	0.9053	8.589	153.40	0.006399	0.04904	0.05373	0.01587	0.03003	
0.1812	0.05667	0.5435	0.7339	3.398	74.08	0.005225	0.01308	0.01860	0.01340	0.01389	
0.2069	0.05999	0.7456	0.7869	4.585	94.03	0.006150	0.04006	0.03832	0.02058	0.02250	
0.2597	0.09744	0.4956	1.1560	3.445	27.23	0.009110	0.07458	0.05661	0.01867	0.05963	
0.1809	0.05883	0.7572	0.7813	5.438	94.44	0.011490	0.02461	0.05688	0.01885	0.01756	



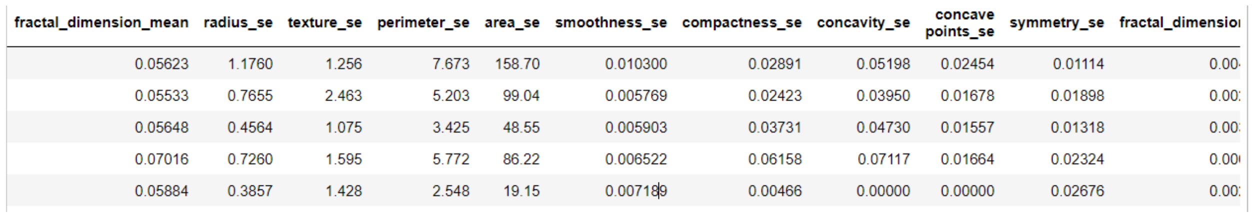
texture_worst	perimeter_worst	area_worst	smoothness_worst	compactness_worst	concavity_worst	points_worst	symmetry_worst	fractal_dimension_worst
17.33	184.60	2019.0	0.1622	0.6656	0.7119	0.2654	0.4601	0.11890
23.41	158.80	1956.0	0.1238	0.1866	0.2416	0.1860	0.2750	0.08902
25.53	152.50	1709.0	0.1444	0.4245	0.4504	0.2430	0.3613	0.08758
26.50	98.87	567.7	0.2098	0.8663	0.6869	0.2575	0.6638	0.17300
16.67	152.20	1575.0	0.1374	0.2050	0.4000	0.1625	0.2364	0.07678

data.tail()

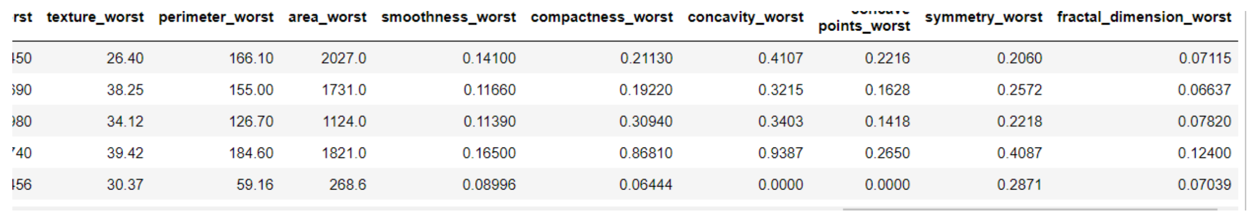


Out[13]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symm
564	926424	M	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	
565	926682	M	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	
566	926954	M	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	
567	927241	M	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	
568	92751	B	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	

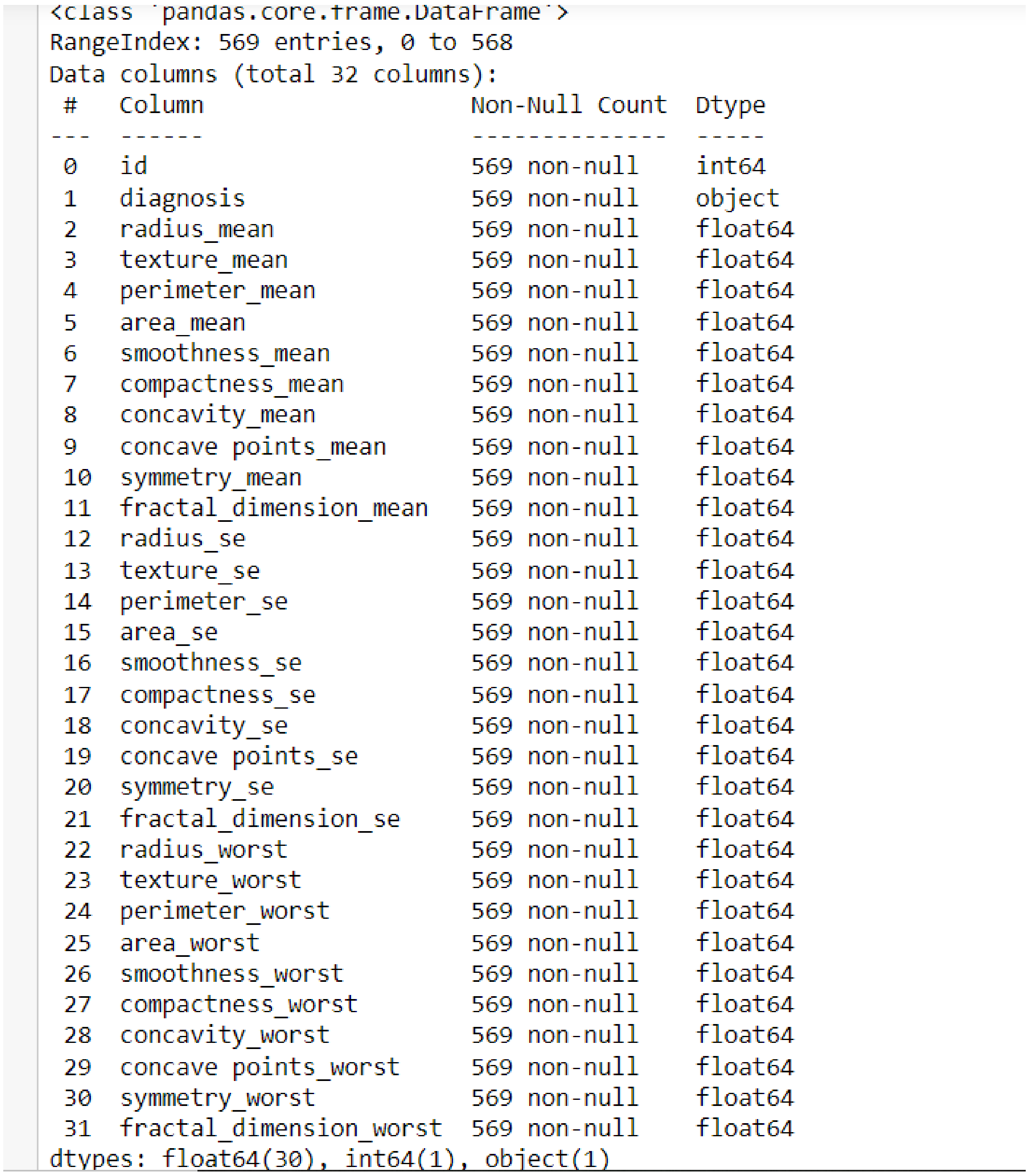


fractal_dimension_mean	radius_se	texture_se	perimeter_se	area_se	smoothness_se	compactness_se	concavity_se	concave points_se	symmetry_se	fractal_dimensio
0.05623	1.1760	1.256	7.673	158.70	0.010300	0.02891	0.05198	0.02454	0.01114	0.00
0.05533	0.7655	2.463	5.203	99.04	0.005769	0.02423	0.03950	0.01678	0.01898	0.00
0.05648	0.4564	1.075	3.425	48.55	0.005903	0.03731	0.04730	0.01557	0.01318	0.00
0.07016	0.7260	1.595	5.772	86.22	0.006522	0.06158	0.07117	0.01664	0.02324	0.00
0.05884	0.3857	1.428	2.548	19.15	0.007189	0.00466	0.00000	0.00000	0.02676	0.00



rst	texture_worst	perimeter_worst	area_worst	smoothness_worst	compactness_worst	concavity_worst	points_worst	symmetry_worst	fractal_dimension_worst
50	26.40	166.10	2027.0	0.14100	0.21130	0.4107	0.2216	0.2060	0.07115
90	38.25	155.00	1731.0	0.11660	0.19220	0.3215	0.1628	0.2572	0.06637
80	34.12	126.70	1124.0	0.11390	0.30940	0.3403	0.1418	0.2218	0.07820
40	39.42	184.60	1821.0	0.16500	0.86810	0.9387	0.2650	0.4087	0.12400
56	30.37	59.16	268.6	0.08996	0.06444	0.0000	0.0000	0.2871	0.07039

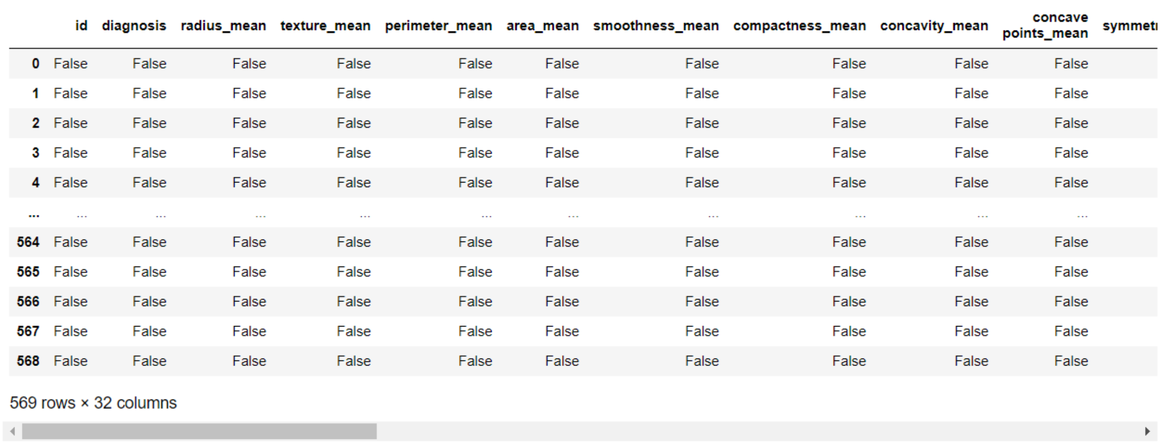
data.info()



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 32 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   id                       569 non-null    int64  
 1   diagnosis                569 non-null    object 
 2   radius_mean              569 non-null    float64
 3   texture_mean             569 non-null    float64
 4   perimeter_mean           569 non-null    float64
 5   area_mean                569 non-null    float64
 6   smoothness_mean          569 non-null    float64
 7   compactness_mean         569 non-null    float64
 8   concavity_mean           569 non-null    float64
 9   concave points_mean      569 non-null    float64
 10  symmetry_mean            569 non-null    float64
 11  fractal_dimension_mean   569 non-null    float64
 12  radius_se                569 non-null    float64
 13  texture_se               569 non-null    float64
 14  perimeter_se             569 non-null    float64
 15  area_se                  569 non-null    float64
 16  smoothness_se            569 non-null    float64
 17  compactness_se           569 non-null    float64
 18  concavity_se             569 non-null    float64
 19  concave points_se        569 non-null    float64
 20  symmetry_se              569 non-null    float64
 21  fractal_dimension_se     569 non-null    float64
 22  radius_worst             569 non-null    float64
 23  texture_worst            569 non-null    float64
 24  perimeter_worst          569 non-null    float64
 25  area_worst               569 non-null    float64
 26  smoothness_worst         569 non-null    float64
 27  compactness_worst        569 non-null    float64
 28  concavity_worst          569 non-null    float64
 29  concave points_worst     569 non-null    float64
 30  symmetry_worst           569 non-null    float64
 31  fractal_dimension_worst  569 non-null    float64
dtypes: float64(30), int64(1), object(1)
```

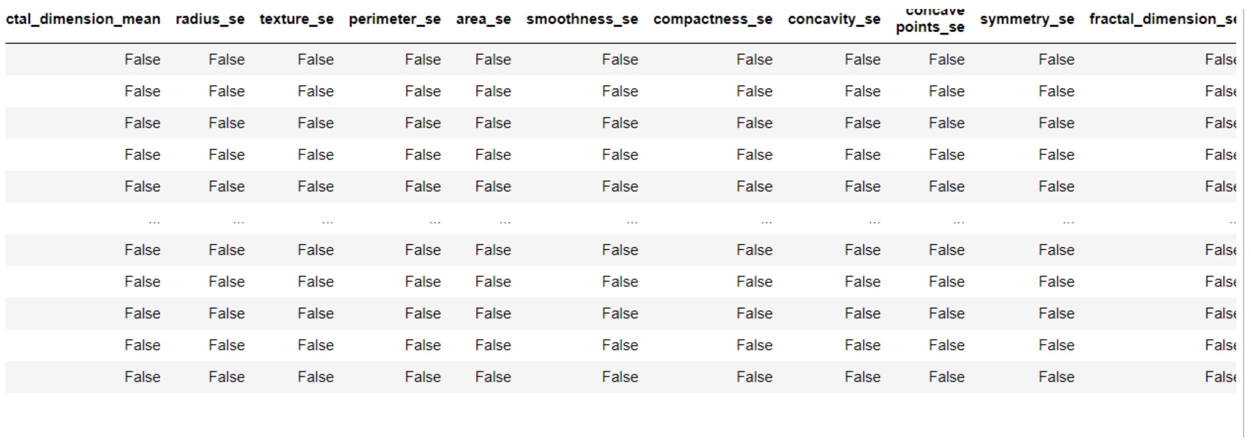
data.isna()

Out[15]:

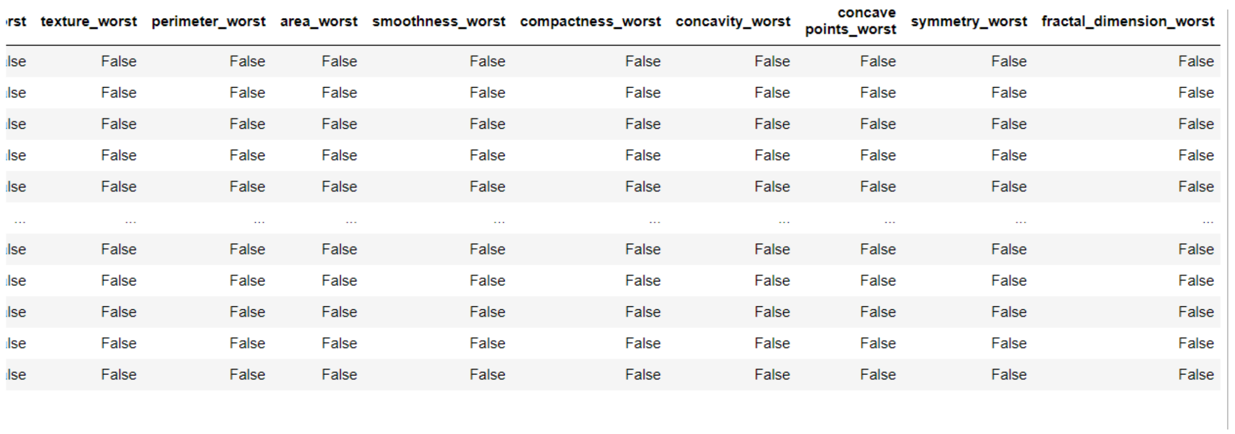


	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetr
0	False	False	False	False	False	False	False	False	False	False	
1	False	False	False	False	False	False	False	False	False	False	
2	False	False	False	False	False	False	False	False	False	False	
3	False	False	False	False	False	False	False	False	False	False	
4	False	False	False	False	False	False	False	False	False	False	
...	...	...	...	...	...	...	...	...	...	...	
564	False	False	False	False	False	False	False	False	False	False	
565	False	False	False	False	False	False	False	False	False	False	
566	False	False	False	False	False	False	False	False	False	False	
567	False	False	False	False	False	False	False	False	False	False	
568	False	False	False	False	False	False	False	False	False	False	

569 rows × 32 columns

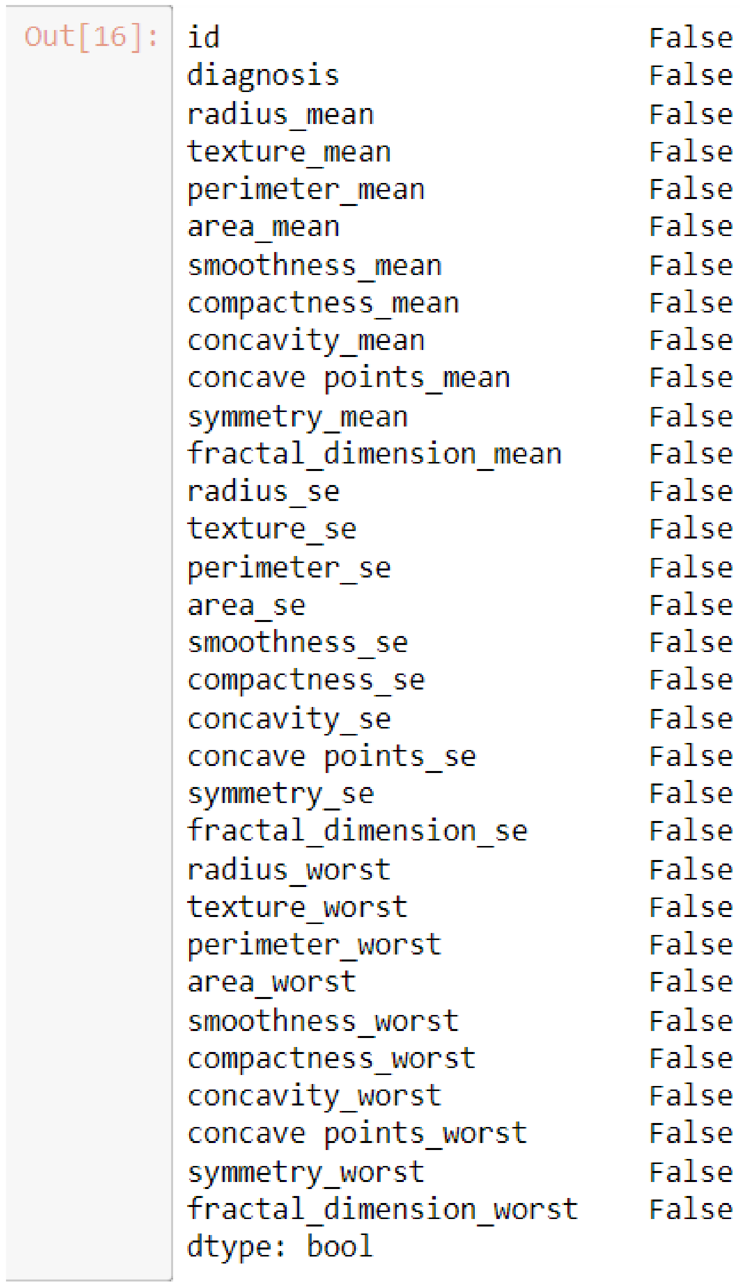


ctal_dimension_mean	radius_se	texture_se	perimeter_se	area_se	smoothness_se	compactness_se	concavity_se	concave points_se	symmetry_se	fractal_dimension_s
False	False	False	False	False	False	False	False	False	False	Fals
False	False	False	False	False	False	False	False	False	False	Fals
False	False	False	False	False	False	False	False	False	False	Fals
False	False	False	False	False	False	False	False	False	False	Fals
False	False	False	False	False	False	False	False	False	False	Fals
...	...	...	...	...	...	...	...	...	...	..
False	False	False	False	False	False	False	False	False	False	Fals
False	False	False	False	False	False	False	False	False	False	Fals
False	False	False	False	False	False	False	False	False	False	Fals
False	False	False	False	False	False	False	False	False	False	Fals
False	False	False	False	False	False	False	False	False	False	Fals



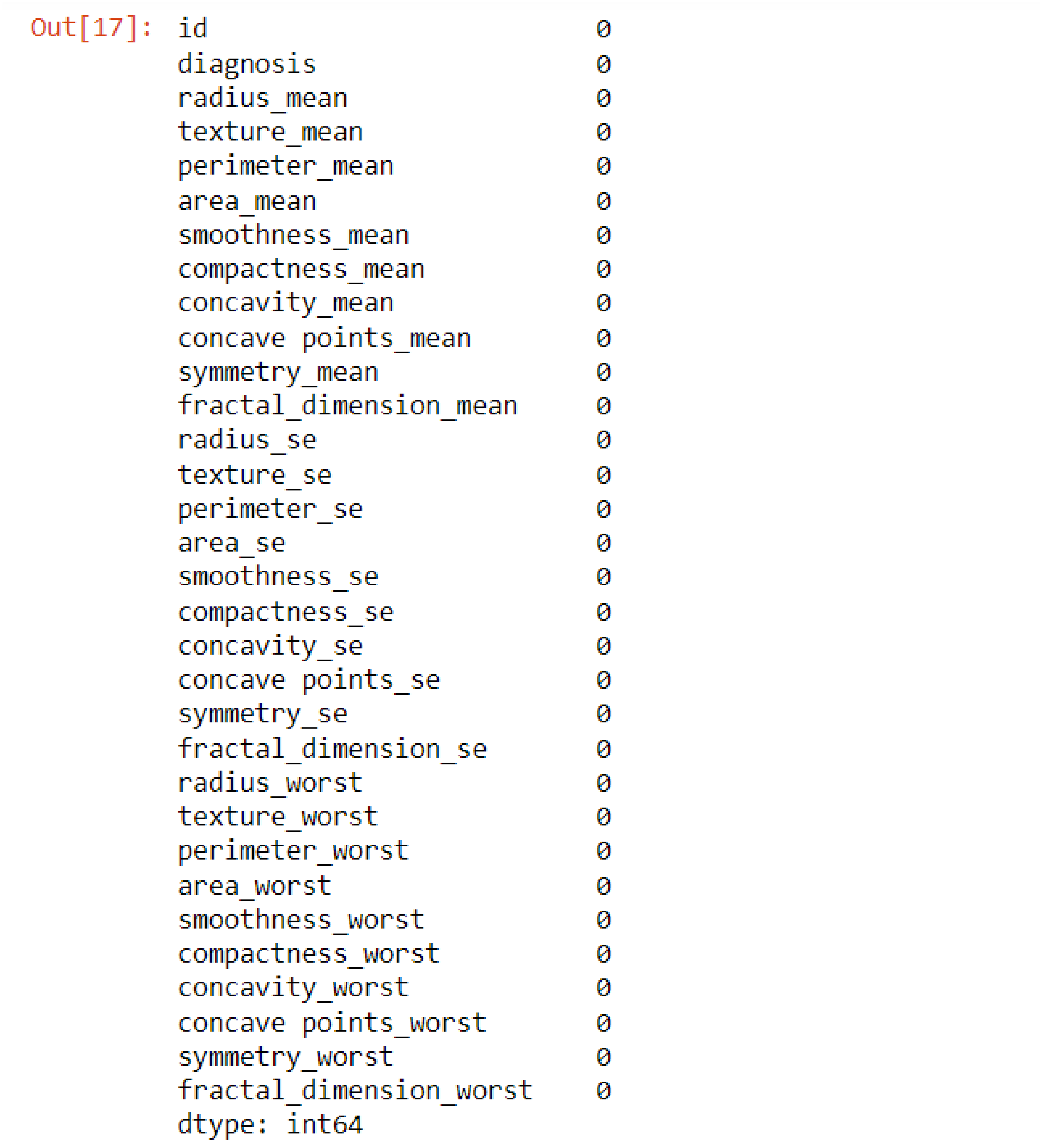
rst	texture_worst	perimeter_worst	area_worst	smoothness_worst	compactness_worst	concavity_worst	concave points_worst	symmetry_worst	fractal_dimension_worst
lse	False	False	False	False	False	False	False	False	False
lse	False	False	False	False	False	False	False	False	False
lse	False	False	False	False	False	False	False	False	False
lse	False	False	False	False	False	False	False	False	False
lse	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...
lse	False	False	False	False	False	False	False	False	False
lse	False	False	False	False	False	False	False	False	False
lse	False	False	False	False	False	False	False	False	False
lse	False	False	False	False	False	False	False	False	False
lse	False	False	False	False	False	False	False	False	False

data.isna().any()



```
Out[16]: id                         False
         diagnosis                  False
         radius_mean                False
         texture_mean               False
         perimeter_mean             False
         area_mean                  False
         smoothness_mean            False
         compactness_mean           False
         concavity_mean             False
         concave points_mean        False
         symmetry_mean              False
         fractal_dimension_mean     False
         radius_se                  False
         texture_se                 False
         perimeter_se               False
         area_se                    False
         smoothness_se              False
         compactness_se             False
         concavity_se               False
         concave points_se          False
         symmetry_se                False
         fractal_dimension_se       False
         radius_worst               False
         texture_worst              False
         perimeter_worst            False
         area_worst                 False
         smoothness_worst           False
         compactness_worst          False
         concavity_worst            False
         concave points_worst       False
         symmetry_worst             False
         fractal_dimension_worst    False
         dtype: bool
```

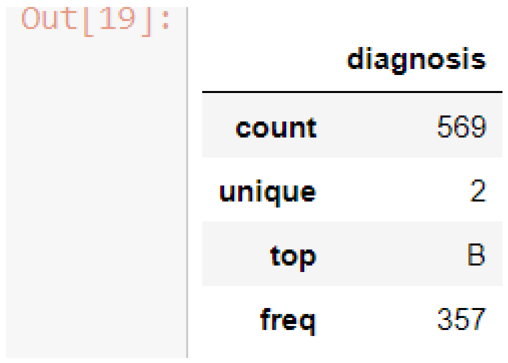
data.isna().sum()



```
Out[17]: id                         0
         diagnosis                  0
         radius_mean                0
         texture_mean               0
         perimeter_mean             0
         area_mean                  0
         smoothness_mean            0
         compactness_mean           0
         concavity_mean             0
         concave points_mean        0
         symmetry_mean              0
         fractal_dimension_mean     0
         radius_se                  0
         texture_se                 0
         perimeter_se               0
         area_se                    0
         smoothness_se              0
         compactness_se             0
         concavity_se               0
         concave points_se          0
         symmetry_se                0
         fractal_dimension_se       0
         radius_worst               0
         texture_worst              0
         perimeter_worst            0
         area_worst                 0
         smoothness_worst           0
         compactness_worst          0
         concavity_worst            0
         concave points_worst       0
         symmetry_worst             0
         fractal_dimension_worst    0
         dtype: int64
```

```
data = data.dropna(axis='columns')
```

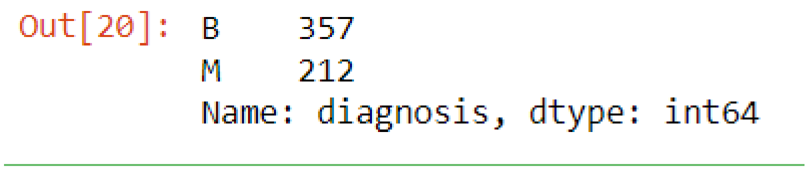
```
data.describe(include="O")
```



```
Out[19]:
```

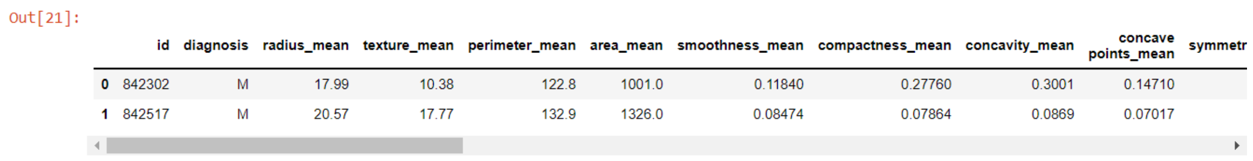
	diagnosis
count	569
unique	2
top	B
freq	357

data.diagnosis.value_counts()



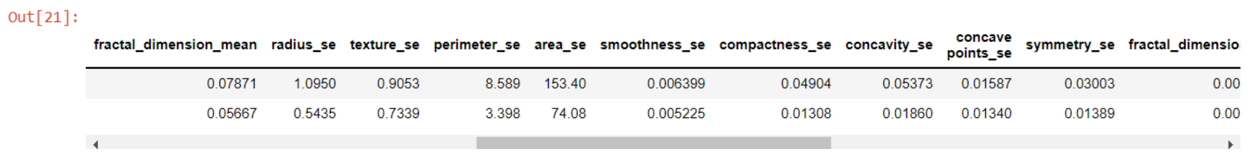
```
Out[20]: B    357
         M    212
         Name: diagnosis, dtype: int64
```

data.head(2)



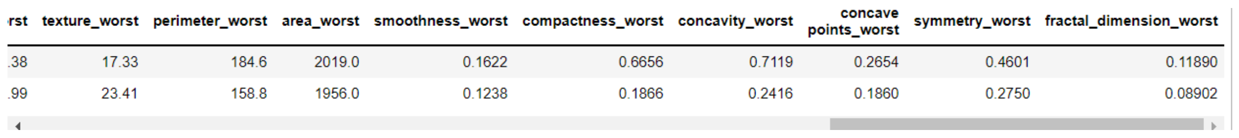
```
Out[21]:
```

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetr
0	842302	M	17.99	10.38	122.8	1001.0	0.11840	0.27760	0.3001	0.14710	
1	842517	M	20.57	17.77	132.9	1326.0	0.08474	0.07864	0.0869	0.07017	



```
Out[21]:
```

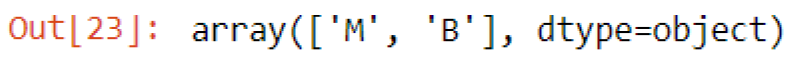
fractal_dimension_mean	radius_se	texture_se	perimeter_se	area_se	smoothness_se	compactness_se	concavity_se	concave points_se	symmetry_se	fractal_dimensio
0.07871	1.0950	0.9053	8.589	153.40	0.006399	0.04904	0.05373	0.01587	0.03003	0.00
0.05667	0.5435	0.7339	3.398	74.08	0.005225	0.01308	0.01860	0.01340	0.01389	0.00



rst	texture_worst	perimeter_worst	area_worst	smoothness_worst	compactness_worst	concavity_worst	concave points_worst	symmetry_worst	fractal_dimension_worst
.38	17.33	184.6	2019.0	0.1622	0.6656	0.7119	0.2654	0.4601	0.11890
.99	23.41	158.8	1956.0	0.1238	0.1866	0.2416	0.1860	0.2750	0.08902

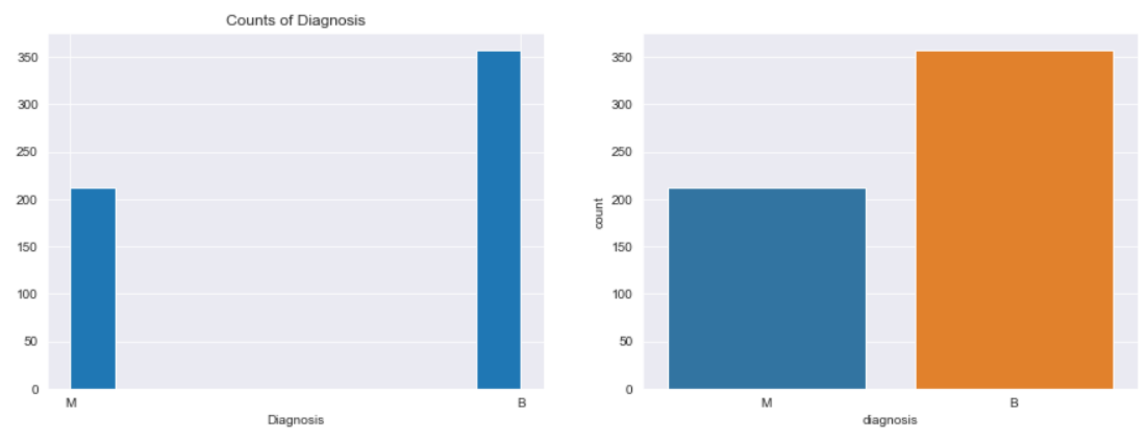
diagnosis_unique = data.diagnosis.unique()

diagnosis_unique

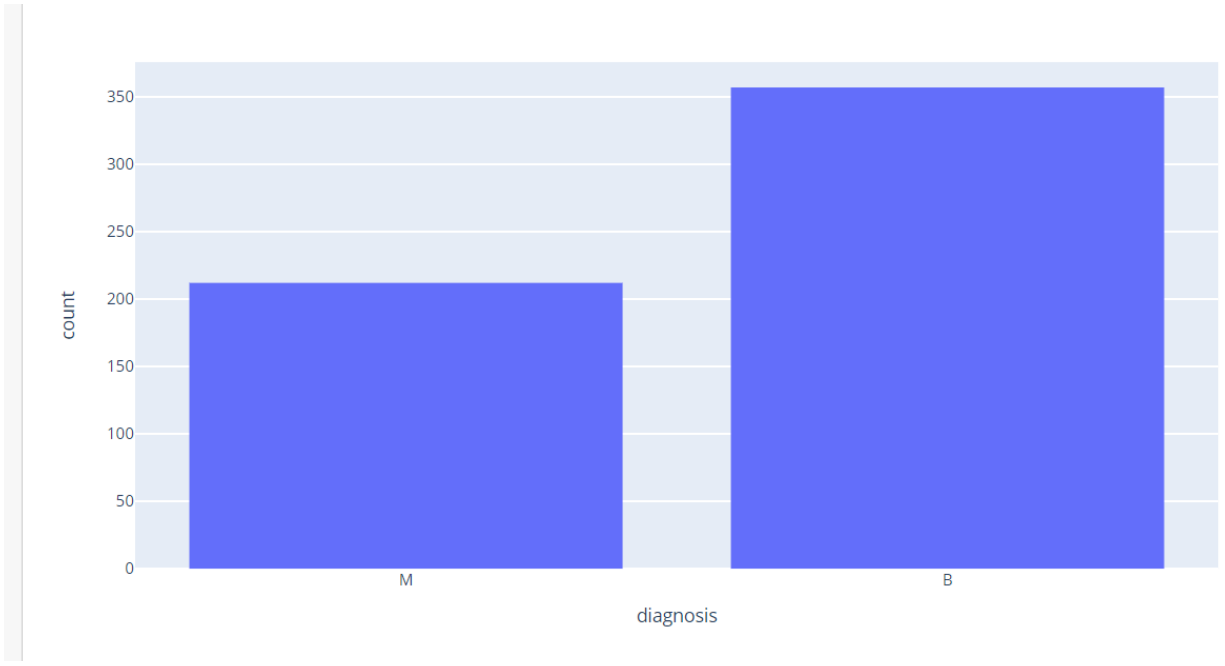


```
Out[23]: array(['M', 'B'], dtype=object)
```

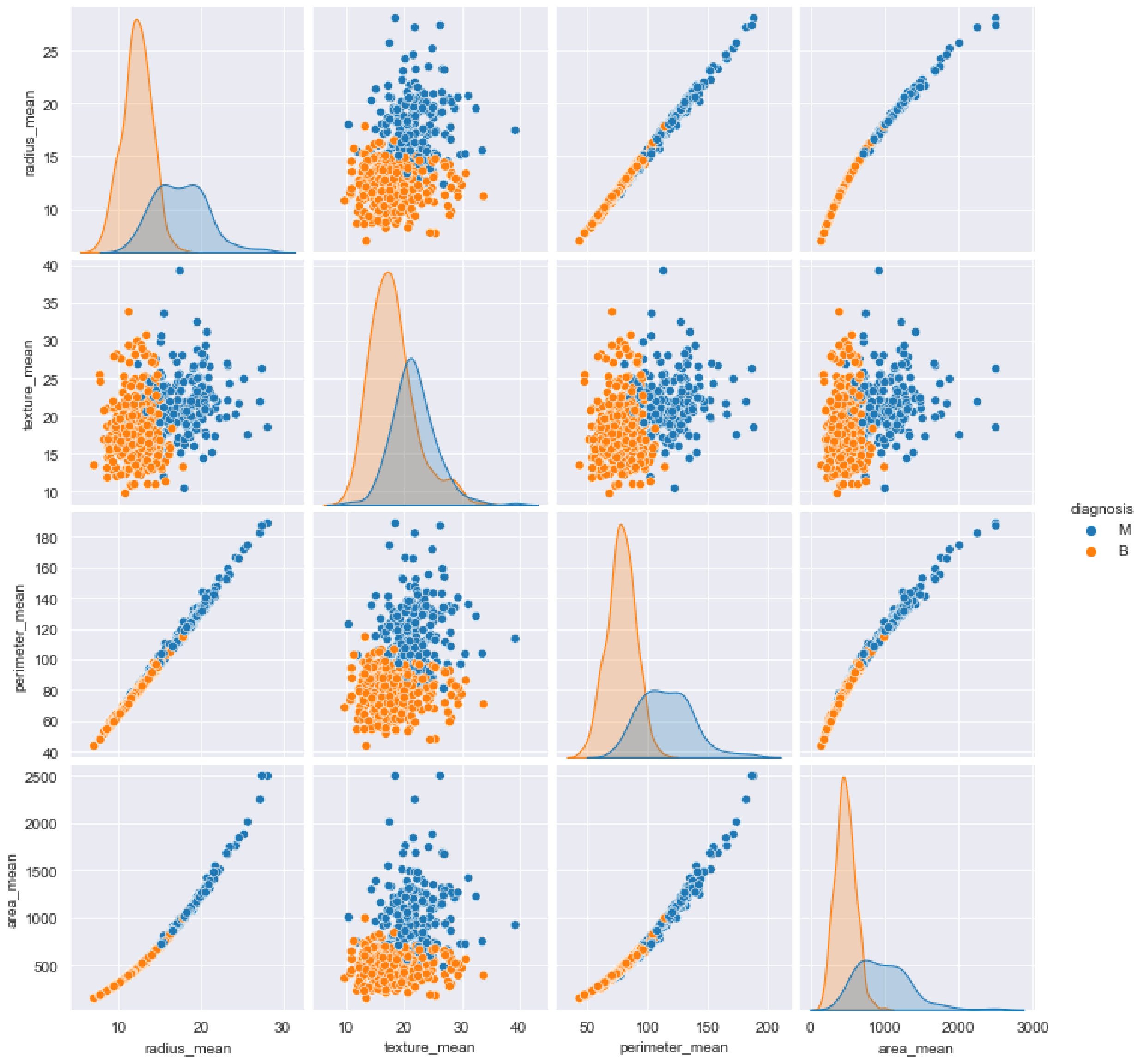
```
import matplotlib.pyplot as plt

import seaborn as sns

import plotly.express as px

import plotly.graph_objects as go%matplotlib inline sns.set_style('darkgrid')

  plt.figure(figsize=(15, 5))

plt.subplot(1, 2, 1)

plt.hist( data.diagnosis)

# plt.legend()

plt.title("Counts of Diagnosis")

plt.xlabel("Diagnosis")

plt.subplot(1, 2, 2)

sns.countplot('diagnosis', data=data); # ";" to remove output like this >
<matplotlib.axes._subplots.AxesSubplot at 0x7f3a1dddba50>
```



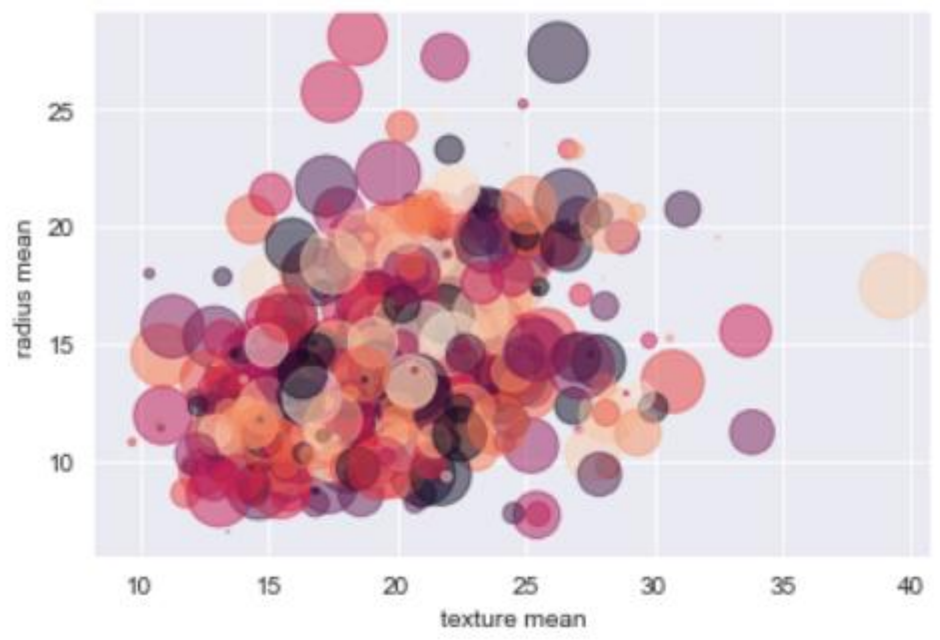
```
# plt.figure(figsize=(7,12))

px.histogram(data, x='diagnosis')

# plt.show()
```



```
cols = ["diagnosis", "radius_mean", "texture_mean", "perimeter_mean",
"area_mean"]

sns.pairplot(data[cols], hue="diagnosis")

plt.show()
```

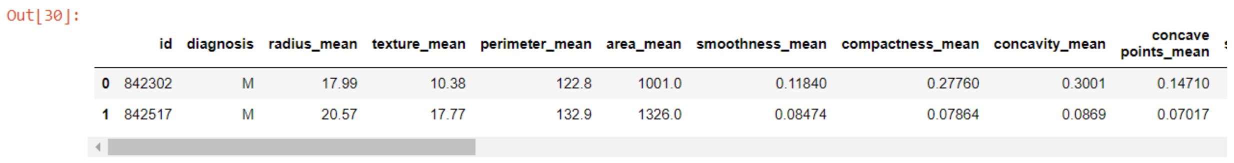


```
size = len(data['texture_mean'])

area = np.pi * (15 * np.random.rand( size ))**2

colors = np.random.rand( size )

plt.xlabel("texture mean")

plt.ylabel("radius mean")

plt.scatter(data['texture_mean'], data['radius_mean'], s=area, c=colors, alpha=0.5);
```



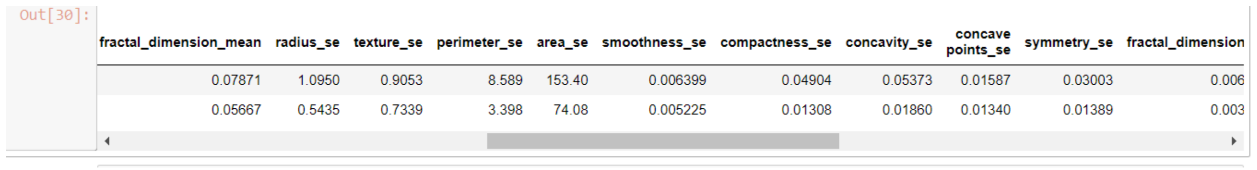
from sklearn.preprocessing import LabelEncoder

data.head(2)



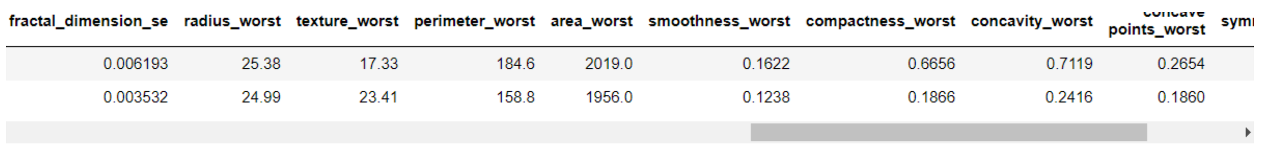
Out[30]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean
0	842302	M	17.99	10.38	122.8	1001.0	0.11840	0.27760	0.3001	0.14710
1	842517	M	20.57	17.77	132.9	1326.0	0.08474	0.07864	0.0869	0.07017

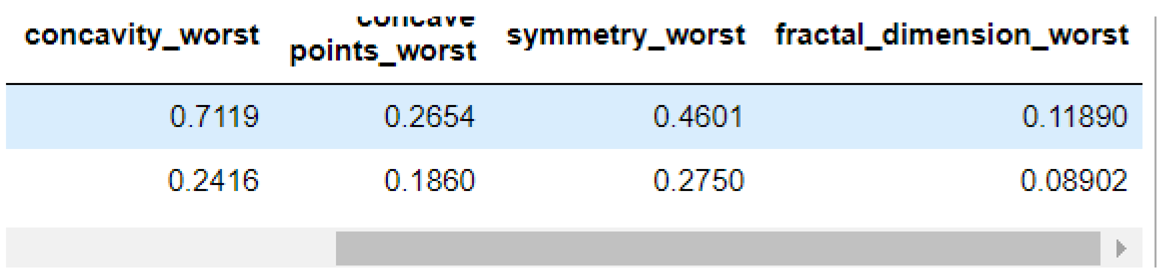


Out[30]:

fractal_dimension_mean	radius_se	texture_se	perimeter_se	area_se	smoothness_se	compactness_se	concavity_se	concave points_se	symmetry_se	fractal_dimension
0.07871	1.0950	0.9053	8.589	153.40	0.006399	0.04904	0.05373	0.01587	0.03003	0.006
0.05667	0.5435	0.7339	3.398	74.08	0.005225	0.01308	0.01860	0.01340	0.01389	0.003



fractal_dimension_se	radius_worst	texture_worst	perimeter_worst	area_worst	smoothness_worst	compactness_worst	concavity_worst	concave points_worst
0.006193	25.38	17.33	184.6	2019.0	0.1622	0.6656	0.7119	0.2654
0.003532	24.99	23.41	158.8	1956.0	0.1238	0.1866	0.2416	0.1860

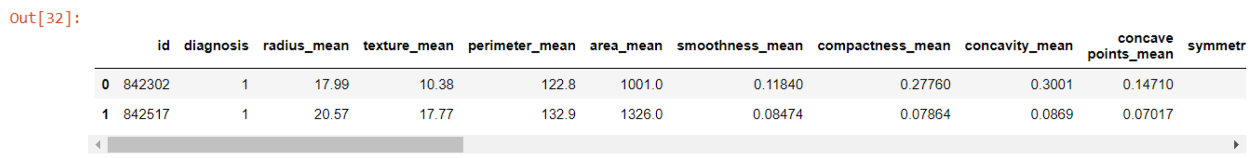


concavity_worst	concave points_worst	symmetry_worst	fractal_dimension_worst
0.7119	0.2654	0.4601	0.11890
0.2416	0.1860	0.2750	0.08902

labelencoder_Y = LabelEncoder()

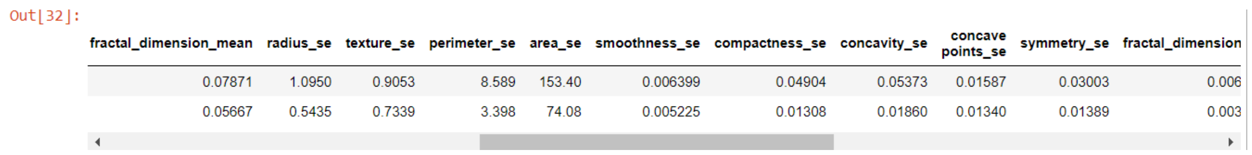
data.diagnosis = labelencoder_Y.fit_transform(data.diagnosis)

data.head(2)



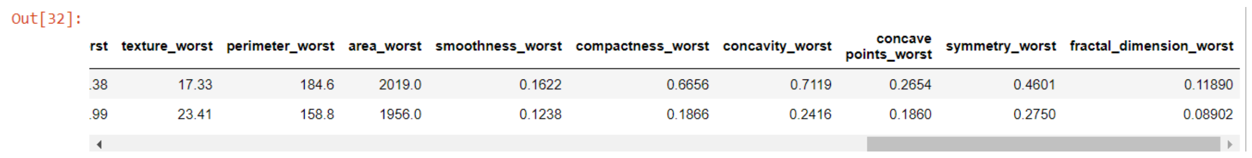
Out[32]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetr
0	842302	1	17.99	10.38	122.8	1001.0	0.11840	0.27760	0.3001	0.14710	
1	842517	1	20.57	17.77	132.9	1326.0	0.08474	0.07864	0.0869	0.07017	



Out[32]:

fractal_dimension_mean	radius_se	texture_se	perimeter_se	area_se	smoothness_se	compactness_se	concavity_se	concave points_se	symmetry_se	fractal_dimension
0.07871	1.0950	0.9053	8.589	153.40	0.006399	0.04904	0.05373	0.01587	0.03003	0.006
0.05667	0.5435	0.7339	3.398	74.08	0.005225	0.01308	0.01860	0.01340	0.01389	0.003

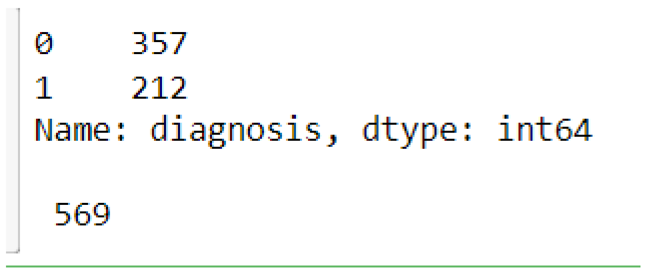


Out[32]:

rst	texture_worst	perimeter_worst	area_worst	smoothness_worst	compactness_worst	concavity_worst	concave points_worst	symmetry_worst	fractal_dimension_worst
.38	17.33	184.6	2019.0	0.1622	0.6656	0.7119	0.2654	0.4601	0.11890
.99	23.41	158.8	1956.0	0.1238	0.1866	0.2416	0.1860	0.2750	0.08902

print(data.diagnosis.value_counts())

print("\n", data.diagnosis.value_counts().sum())

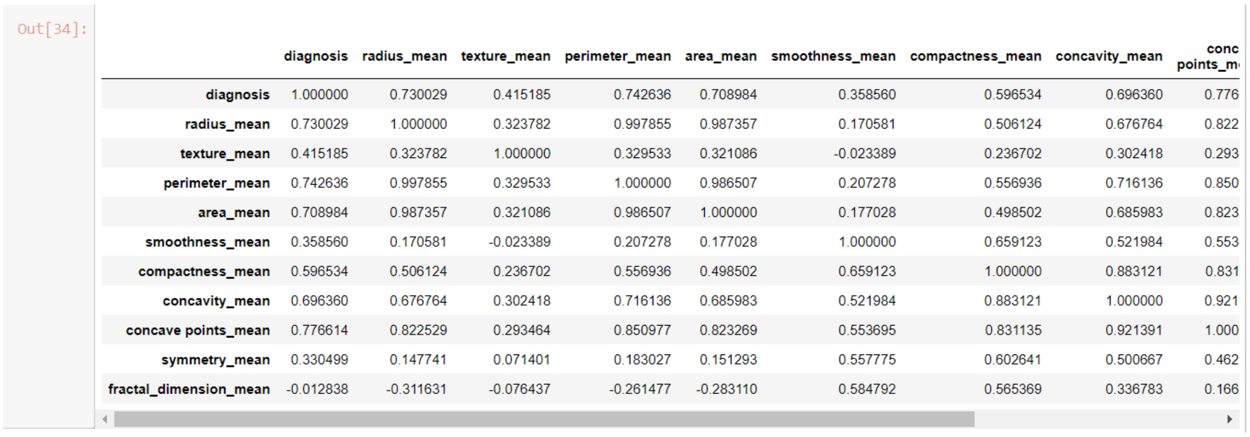


```
0    357
1    212
Name: diagnosis, dtype: int64

 569
```

cols = ['diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean' 'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean','concave points_mean', 'symmetry_mean', 'fractal_dimension_mean']

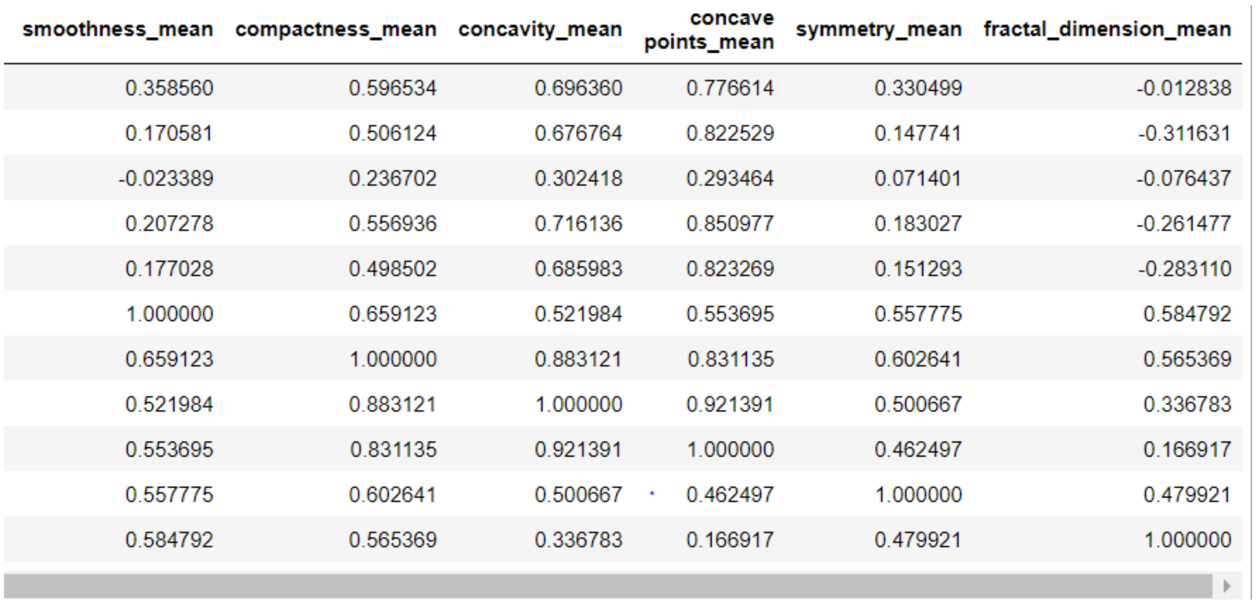
print(len(cols))

data[cols].corr()



Out[34]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	conc points_m
diagnosis	1.000000	0.730029	0.415185	0.742636	0.708984	0.358560	0.596534	0.696360	0.776
radius_mean	0.730029	1.000000	0.323782	0.997855	0.987357	0.170581	0.506124	0.676764	0.822
texture_mean	0.415185	0.323782	1.000000	0.329533	0.321086	-0.023389	0.236702	0.302418	0.293
perimeter_mean	0.742636	0.997855	0.329533	1.000000	0.986507	0.207278	0.556936	0.716136	0.850
area_mean	0.708984	0.987357	0.321086	0.986507	1.000000	0.177028	0.498502	0.685983	0.823
smoothness_mean	0.358560	0.170581	-0.023389	0.207278	0.177028	1.000000	0.659123	0.521984	0.553
compactness_mean	0.596534	0.506124	0.236702	0.556936	0.498502	0.659123	1.000000	0.883121	0.831
concavity_mean	0.696360	0.676764	0.302418	0.716136	0.685983	0.521984	0.883121	1.000000	0.921
concave points_mean	0.776614	0.822529	0.293464	0.850977	0.823269	0.553695	0.831135	0.921391	1.000
symmetry_mean	0.330499	0.147741	0.071401	0.183027	0.151293	0.557775	0.602641	0.500667	0.462
fractal_dimension_mean	-0.012838	-0.311631	-0.076437	-0.261477	-0.283110	0.584792	0.565369	0.336783	0.166

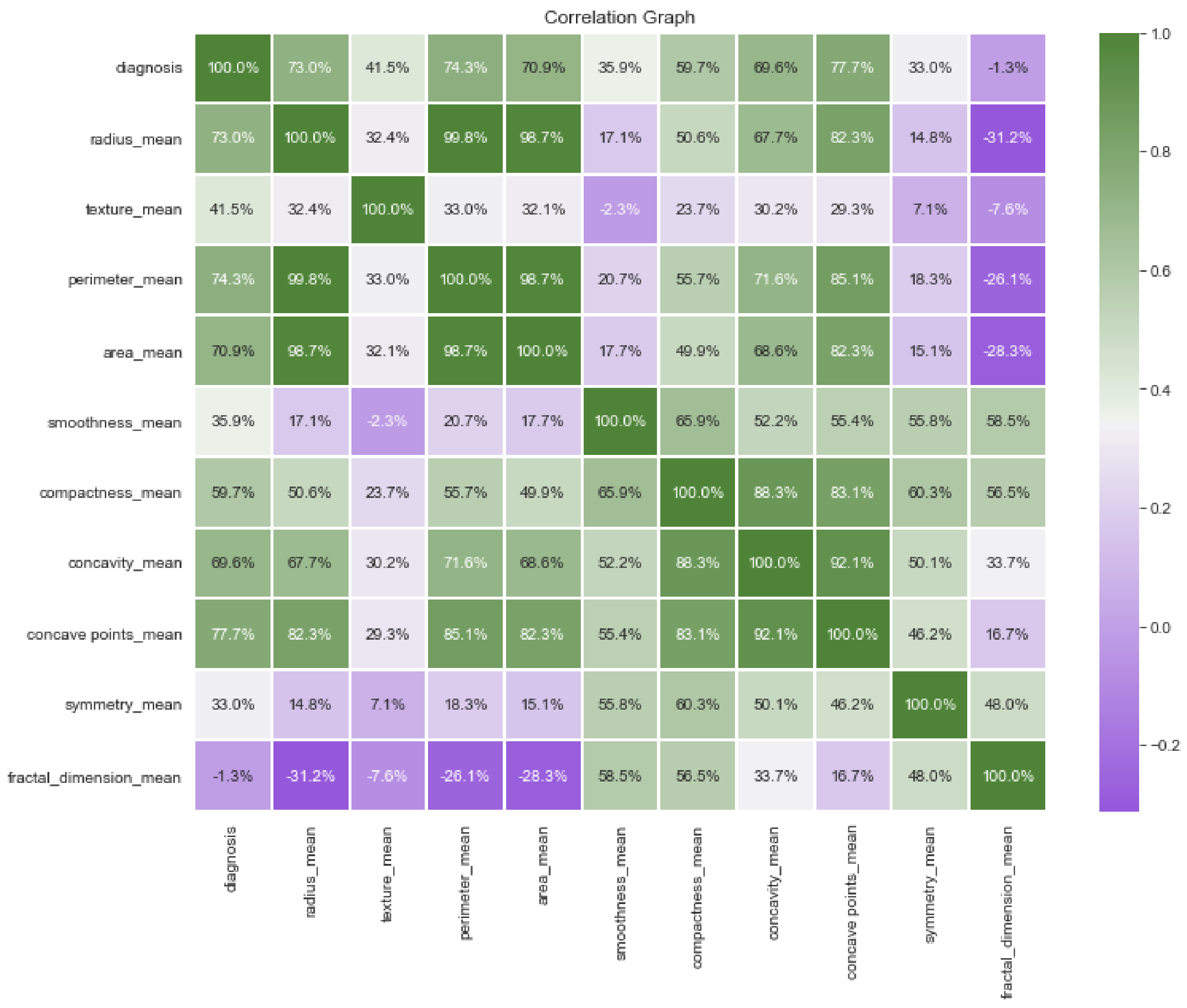


smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_mean	fractal_dimension_mean
0.358560	0.596534	0.696360	0.776614	0.330499	-0.012838
0.170581	0.506124	0.676764	0.822529	0.147741	-0.311631
-0.023389	0.236702	0.302418	0.293464	0.071401	-0.076437
0.207278	0.556936	0.716136	0.850977	0.183027	-0.261477
0.177028	0.498502	0.685983	0.823269	0.151293	-0.283110
1.000000	0.659123	0.521984	0.553695	0.557775	0.584792
0.659123	1.000000	0.883121	0.831135	0.602641	0.565369
0.521984	0.883121	1.000000	0.921391	0.500667	0.336783
0.553695	0.831135	0.921391	1.000000	0.462497	0.166917
0.557775	0.602641	0.500667	0.462497	1.000000	0.479921
0.584792	0.565369	0.336783	0.166917	0.479921	1.000000

plt.figure(figsize=(12, 9))

plt.title("Correlation Graph")

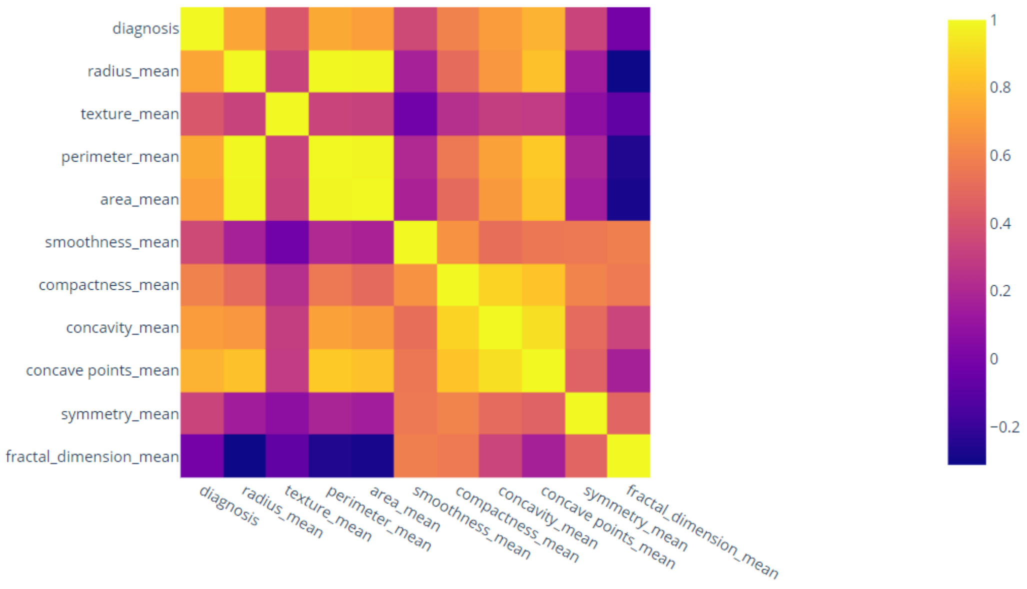
cmap = sns.diverging_palette(1000, 120, as_cmap=True) sns.heatmap(data[cols].corr(), annot=True, fmt='.1%', linewidths=.05, cmap=cmap);



plt.figure(figsize=(15, 10))

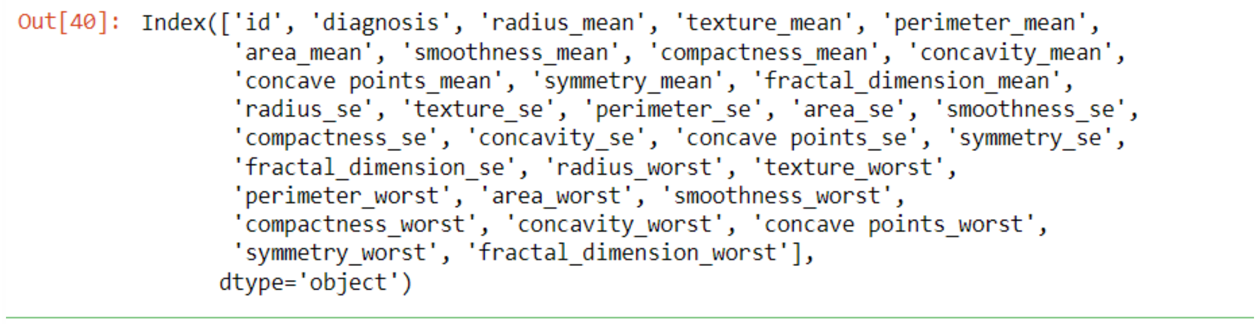
fig = px.imshow(data[cols].corr());

fig.show()



```
from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler

from sklearn.linear_model import LogisticRegression

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.naive_bayes import GaussianNB

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy_score, confusion_matrix, f1_score

from sklearn.metrics import classification_report

from sklearn.model_selection import KFold

from sklearn.model_selection import cross_validate, cross_val_score

from sklearn.svm import SVC

from sklearn import metrics
```

data.columns



```
Out[40]: Index(['id', 'diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean',
                'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean',
                'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean',
                'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_se',
                'compactness_se', 'concavity_se', 'concave points_se', 'symmetry_se',
                'fractal_dimension_se', 'radius_worst', 'texture_worst',
                'perimeter_worst', 'area_worst', 'smoothness_worst',
                'compactness_worst', 'concavity_worst', 'concave points_worst',
                'symmetry_worst', 'fractal_dimension_worst'],
               dtype='object')
```

prediction_feature = ["radius_mean", 'perimeter_mean', 'area_mean', 'symmetry_mean', 'compactness_mean', 'concave points_mean']

targeted_feature = 'diagnosis'

len(prediction_feature)

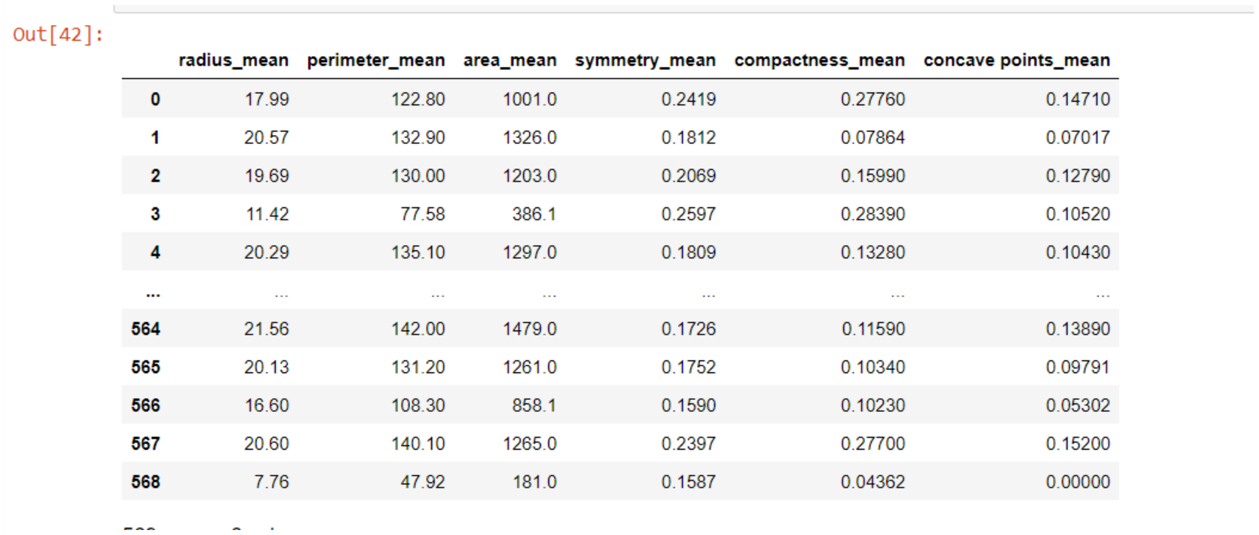
OUT : 6

X = data[prediction_feature]

X

# print(X.shape)

# print(X.values)

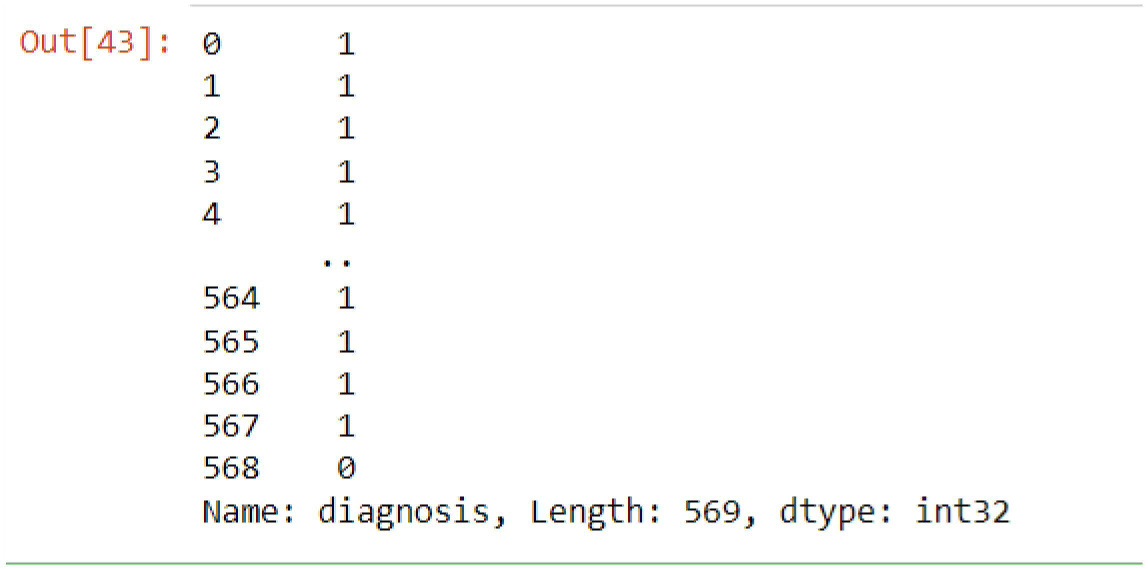


Out[42]:

	radius_mean	perimeter_mean	area_mean	symmetry_mean	compactness_mean	concave points_mean
0	17.99	122.80	1001.0	0.2419	0.27760	0.14710
1	20.57	132.90	1326.0	0.1812	0.07864	0.07017
2	19.69	130.00	1203.0	0.2069	0.15990	0.12790
3	11.42	77.58	386.1	0.2597	0.28390	0.10520
4	20.29	135.10	1297.0	0.1809	0.13280	0.10430
...	...	...	...	...	...	...
564	21.56	142.00	1479.0	0.1726	0.11590	0.13890
565	20.13	131.20	1261.0	0.1752	0.10340	0.09791
566	16.60	108.30	858.1	0.1590	0.10230	0.05302
567	20.60	140.10	1265.0	0.2397	0.27700	0.15200
568	7.76	47.92	181.0	0.1587	0.04362	0.00000

y = data.diagnosis

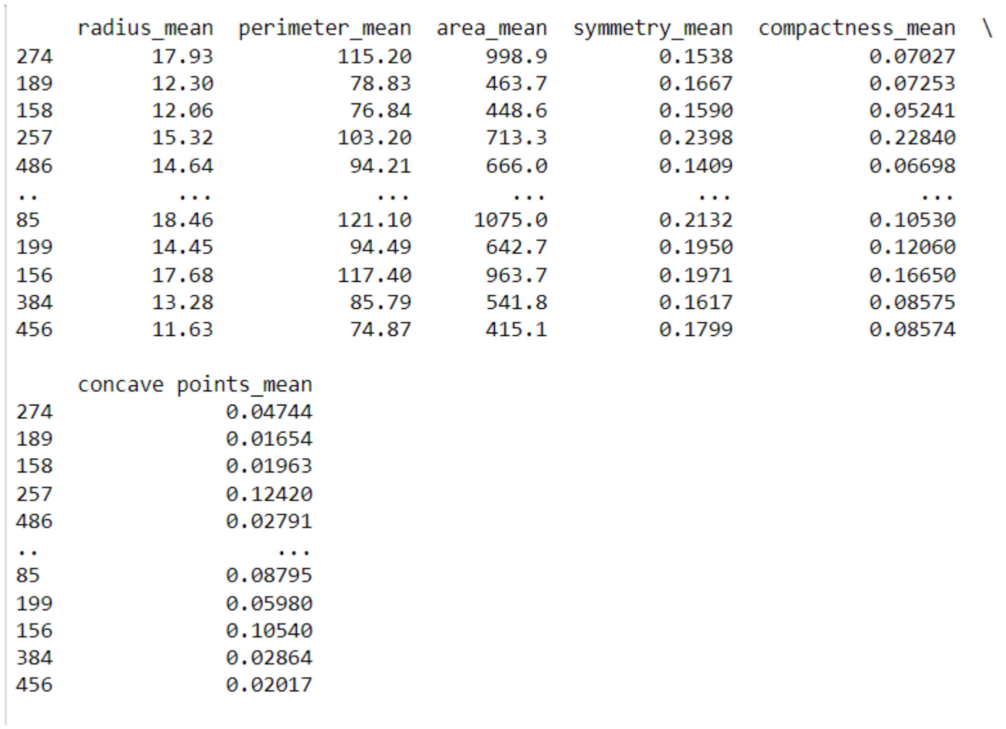
y



```
Out[43]: 0      1
         1      1
         2      1
         3      1
         4      1
               ..
         564    1
         565    1
         566    1
         567    1
         568    0
         Name: diagnosis, Length: 569, dtype: int32
```

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=15)

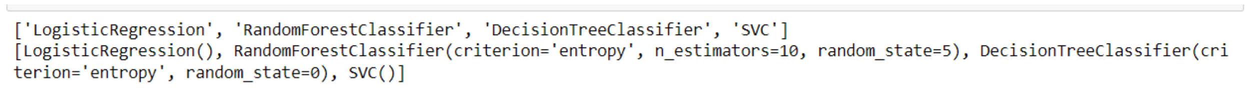
print(X_train)



```
     radius_mean  perimeter_mean  area_mean  symmetry_mean  compactness_mean  \
274        17.93          115.20      998.9         0.1538           0.07027
189        12.30           78.83      463.7         0.1667           0.07253
158        12.06           76.84      448.6         0.1590           0.05241
257        15.32          103.20      713.3         0.2398           0.22840
486        14.64           94.21      666.0         0.1409           0.06698
..           ...             ...        ...            ...               ...
85         18.46          121.10     1075.0         0.2132           0.10530
199        14.45           94.49      642.7         0.1950           0.12060
156        17.68          117.40      963.7         0.1971           0.16650
384        13.28           85.79      541.8         0.1617           0.08575
456        11.63           74.87      415.1         0.1799           0.08574

     concave points_mean
274              0.04744
189              0.01654
158              0.01963
257              0.12420
486              0.02791
..                   ...
85               0.08795
199              0.05980
156              0.10540
384              0.02864
456              0.02017
```

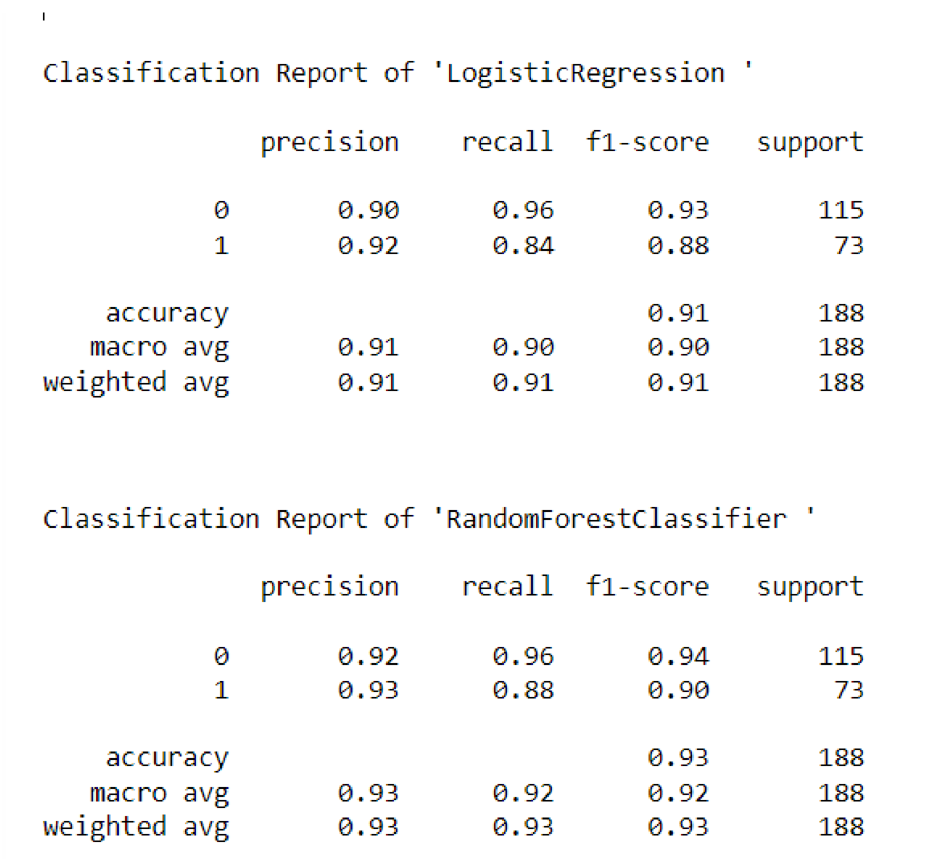
```
sc = StandardScaler()


X_train = sc.fit_transform(X_train)

X_test = sc.fit_transform(X_test)


def model_building(model, X_train, X_test, y_train, y_test):

    """

    Model Fitting, Prediction And Other stuff   return ('score', 'accuracy_score', 'predictions' )

    """

     model.fit(X_train, y_train)

    score = model.score(X_train, y_train)

    predictions = model.predict(X_test)

    accuracy = accuracy_score(predictions, y_test)

    return (score, accuracy, predictions)

models_list = {

"LogisticRegression" :  LogisticRegression(),

"RandomForestClassifier":RandomForestClassifier(n_estimators=10,
criterion='entropy', random_state=5),

 "DecisionTreeClassifier":DecisionTreeClassifier(criterion='entropy',
random_state=0),

"SVC" :  SVC(),

}
```

```
print(list(models_list.keys()))

print(list(models_list.values()))
```

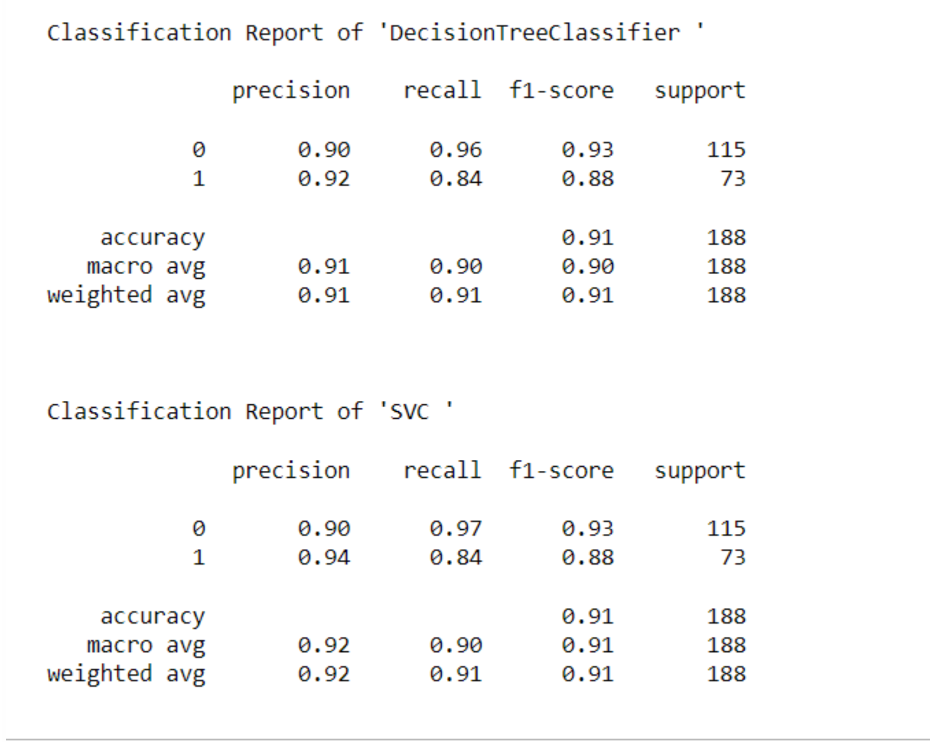


```
['LogisticRegression', 'RandomForestClassifier', 'DecisionTreeClassifier', 'SVC']
[LogisticRegression(), RandomForestClassifier(criterion='entropy', n_estimators=10, random_state=5), DecisionTreeClassifier(cri
terion='entropy', random_state=0), SVC()]
```

```
def cm_metrix_graph(cm):


    sns.heatmap(cm,annot=True,fmt="d")

    plt.show()

df_prediction = []

confusion_matrixs = []

df_prediction_cols = [ 'model_name', 'score', 'accuracy_score' ,
"accuracy_percentage"]

for name, model in zip(list(models_list.keys()), list(models_list.values())):

    (Score, accuracy, predictions) = model_building(model, X_train, X_test,
y_train, y_test )

     print("\n\nClassification Report of '"+ str(name), "'\n")

   print(classification_report(y_test, predictions))

df_prediction.append([name, score, accuracy, "{0:.2%}".format(accuracy)])

   # For Showing Metrics

    confusion_matrixs.append(confusion_matrix(y_test, predictions))

   df_pred = pd.DataFrame(df_prediction, columns=df_prediction_cols)
```

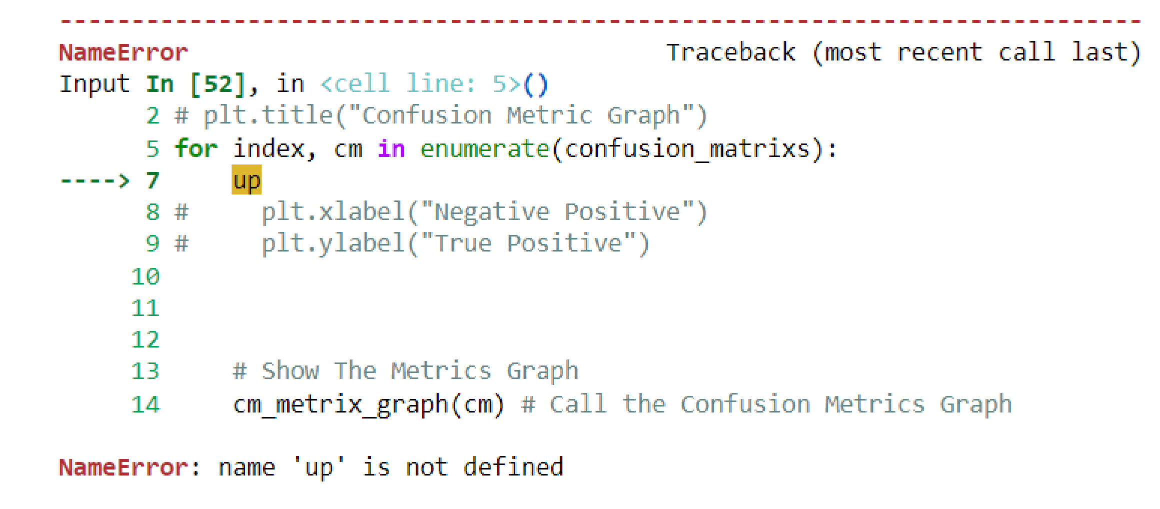


```
Classification Report of 'LogisticRegression '

              precision    recall  f1-score   support

           0       0.90      0.96      0.93       115
           1       0.92      0.84      0.88        73

    accuracy                           0.91       188
   macro avg       0.91      0.90      0.90       188
weighted avg       0.91      0.91      0.91       188


Classification Report of 'RandomForestClassifier '

              precision    recall  f1-score   support

           0       0.92      0.96      0.94       115
           1       0.93      0.88      0.90        73

    accuracy                           0.93       188
   macro avg       0.93      0.92      0.92       188
weighted avg       0.93      0.93      0.93       188
```



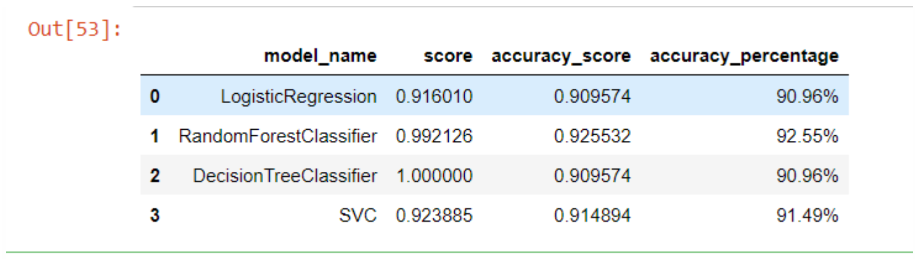
```
Classification Report of 'DecisionTreeClassifier '

              precision    recall  f1-score   support

           0       0.90      0.96      0.93       115
           1       0.92      0.84      0.88        73

    accuracy                           0.91       188
   macro avg       0.91      0.90      0.90       188
weighted avg       0.91      0.91      0.91       188


Classification Report of 'SVC '

              precision    recall  f1-score   support

           0       0.90      0.97      0.93       115
           1       0.94      0.84      0.88        73

    accuracy                           0.91       188
   macro avg       0.92      0.90      0.91       188
weighted avg       0.92      0.91      0.91       188
```

```
print(len(confusion_matrixs))

4

plt.figure(figsize=(10, 2))

# plt.title("Confusion Metric Graph")

for index, cm in enumerate(confusion_matrixs):

up

#     plt.xlabel("Negative Positive")

#     plt.ylabel("True Positive")

 # Show The Metrics Graph

   cm_metrix_graph(cm) # Call the Confusion Metrics Graph

   plt.tight_layout(pad=True)
```



```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Input In [52], in <cell line: 5>()
      2 # plt.title("Confusion Metric Graph")
      5 for index, cm in enumerate(confusion_matrixs):
----> 7     up
      8 #     plt.xlabel("Negative Positive")
      9 #     plt.ylabel("True Positive")
     10 
     11 
     12 
     13     # Show The Metrics Graph
     14     cm_metrix_graph(cm) # Call the Confusion Metrics Graph

NameError: name 'up' is not defined
```

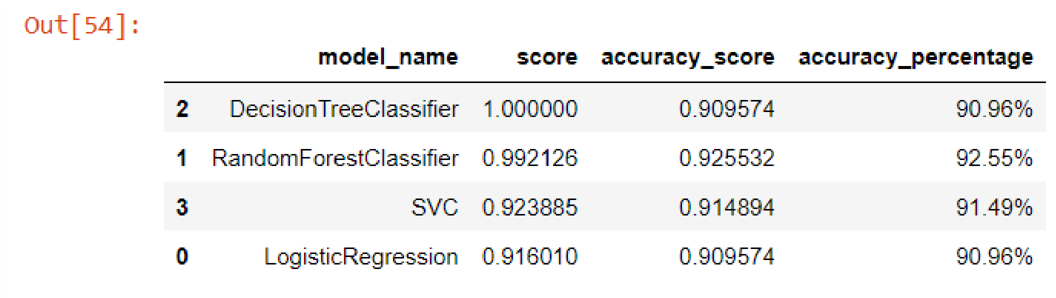
df_pred



Out[53]:

	model_name	score	accuracy_score	accuracy_percentage
0	LogisticRegression	0.916010	0.909574	90.96%
1	RandomForestClassifier	0.992126	0.925532	92.55%
2	DecisionTreeClassifier	1.000000	0.909574	90.96%
3	SVC	0.923885	0.914894	91.49%

```
df_pred.sort_values('score', ascending=False)
```



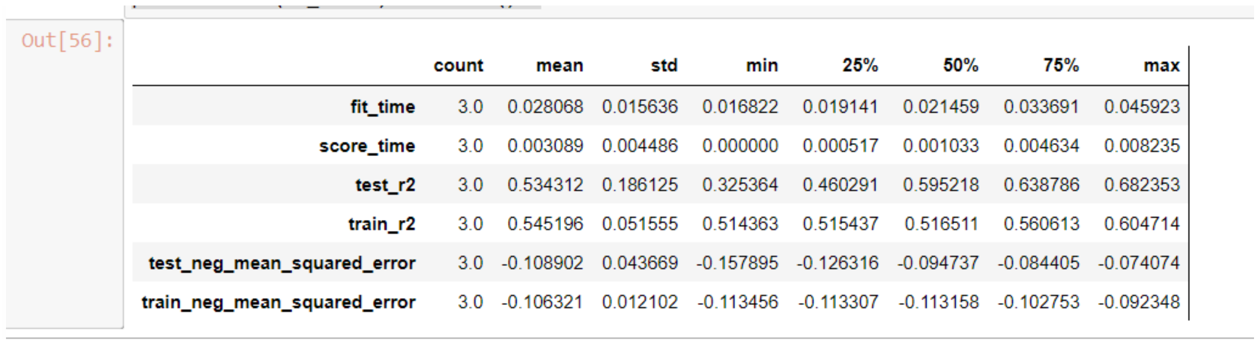
Out[54]:

	model_name	score	accuracy_score	accuracy_percentage
2	DecisionTreeClassifier	1.000000	0.909574	90.96%
1	RandomForestClassifier	0.992126	0.925532	92.55%
3	SVC	0.923885	0.914894	91.49%
0	LogisticRegression	0.916010	0.909574	90.96%

```
len(data)
```

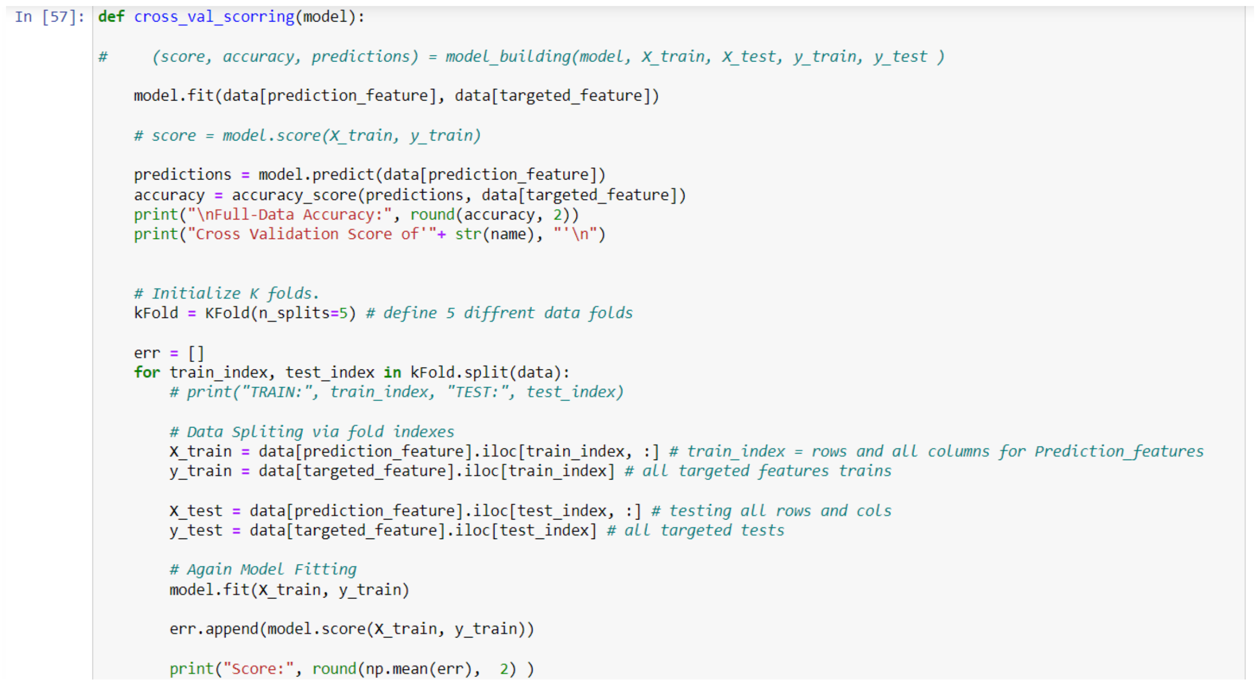
```
OUT: 569
```

```
cv_score = cross_validate(LogisticRegression(), X, y, cv=3,
 scoring=('r2', 'neg_mean_squared_error'),
return_train_score=True)
pd.DataFrame(cv_score).describe().T
```



Out[56]:

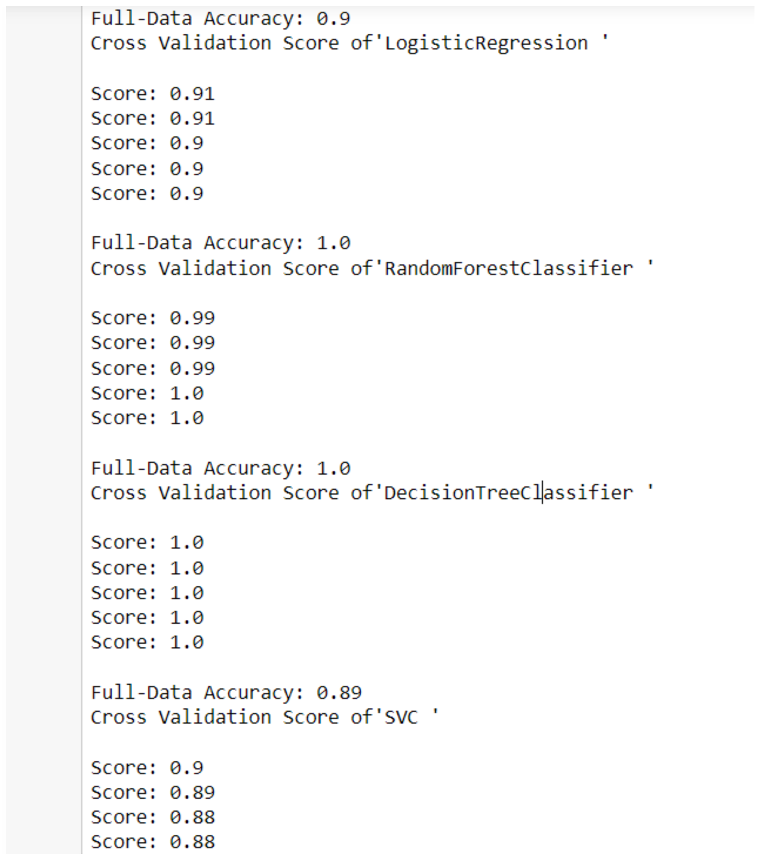
	count	mean	std	min	25%	50%	75%	max
fit_time	3.0	0.028068	0.015636	0.016822	0.019141	0.021459	0.033691	0.045923
score_time	3.0	0.003089	0.004486	0.000000	0.000517	0.001033	0.004634	0.008235
test_r2	3.0	0.534312	0.186125	0.325364	0.460291	0.595218	0.638786	0.682353
train_r2	3.0	0.545196	0.051555	0.514363	0.515437	0.516511	0.560613	0.604714
test_neg_mean_squared_error	3.0	-0.108902	0.043669	-0.157895	-0.126316	-0.094737	-0.084405	-0.074074
train_neg_mean_squared_error	3.0	-0.106321	0.012102	-0.113456	-0.113307	-0.113158	-0.102753	-0.092348



```
In [57]: def cross_val_scorring(model):

         #     (score, accuracy, predictions) = model_building(model, X_train, X_test, y_train, y_test )

             model.fit(data[prediction_feature], data[targeted_feature])

             # score = model.score(X_train, y_train)

             predictions = model.predict(data[prediction_feature])
             accuracy = accuracy_score(predictions, data[targeted_feature])
             print("\nFull-Data Accuracy:", round(accuracy, 2))
             print("Cross Validation Score of'"+ str(name), "'\n")


             # Initialize K folds.
             kFold = KFold(n_splits=5) # define 5 diffrent data folds

             err = []
             for train_index, test_index in kFold.split(data):
                 # print("TRAIN:", train_index, "TEST:", test_index)

                 # Data Spliting via fold indexes
                 X_train = data[prediction_feature].iloc[train_index, :] # train_index = rows and all columns for Prediction_features
                 y_train = data[targeted_feature].iloc[train_index] # all targeted features trains

                 X_test = data[prediction_feature].iloc[test_index, :] # testing all rows and cols
                 y_test = data[targeted_feature].iloc[test_index] # all targeted tests

                 # Again Model Fitting
                 model.fit(X_train, y_train)

                 err.append(model.score(X_train, y_train))

                 print("Score:", round(np.mean(err),  2) )
```

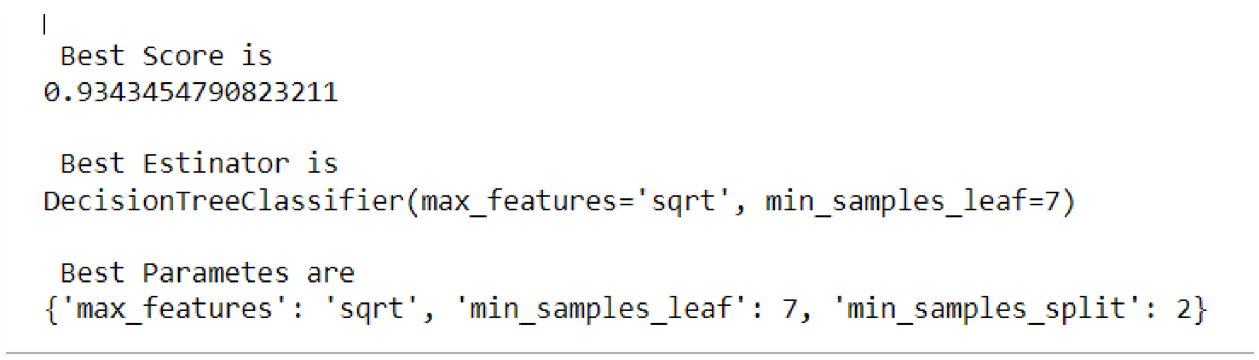
for name, model in zip(list(models_list.keys()), list(models_list.values())):

cross_val_scorring(model)



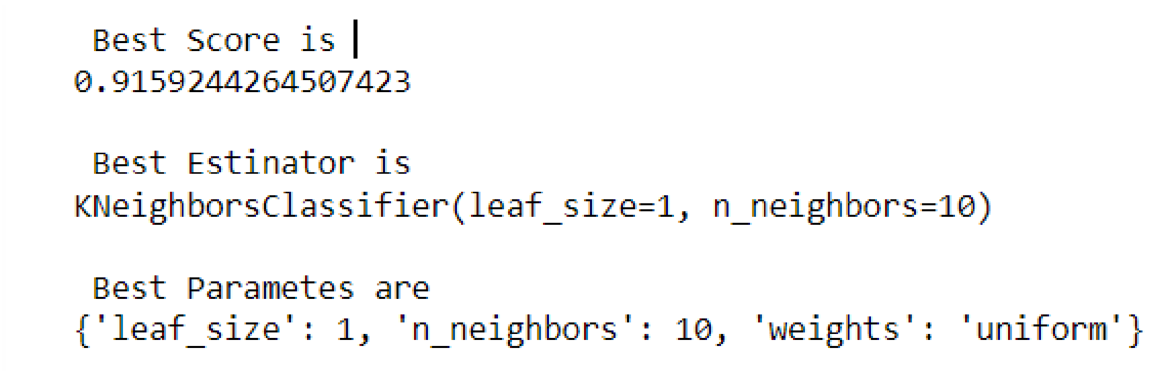
```
Full-Data Accuracy: 0.9
Cross Validation Score of'LogisticRegression '

Score: 0.91
Score: 0.91
Score: 0.9
Score: 0.9
Score: 0.9

Full-Data Accuracy: 1.0
Cross Validation Score of'RandomForestClassifier '

Score: 0.99
Score: 0.99
Score: 0.99
Score: 1.0
Score: 1.0

Full-Data Accuracy: 1.0
Cross Validation Score of'DecisionTreeClassifier '

Score: 1.0
Score: 1.0
Score: 1.0
Score: 1.0
Score: 1.0

Full-Data Accuracy: 0.89
Cross Validation Score of'SVC '

Score: 0.9
Score: 0.89
Score: 0.88
Score: 0.88
```

```
from  sklearn.model_selection import GridSearchCV


model = DecisionTreeClassifier()

# Tunning Params

param_grid = {'max_features': ['auto', 'sqrt', 'log2'],

              'min_samples_split': [2,3,4,5,6,7,8,9,10],

              'min_samples_leaf':[2,3,4,5,6,7,8,9,10] }

# Implement GridSearchCV

gsc = GridSearchCV(model, param_grid, cv=10) # For 10 Cross-Validation

gsc.fit(X_train, y_train) # Model Fitting

print("\n Best Score is ")

print(gsc.best_score_)

print("\n Best Estinator is ")

print(gsc.best_estimator_)

print("\n Best Parametes are")

print(gsc.best_params_)
```



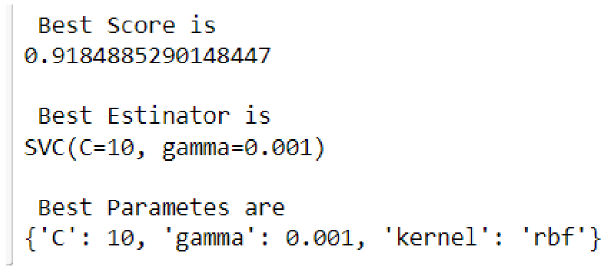
```
 Best Score is
0.9343454790823211

 Best Estinator is
DecisionTreeClassifier(max_features='sqrt', min_samples_leaf=7)

 Best Parametes are
{'max_features': 'sqrt', 'min_samples_leaf': 7, 'min_samples_split': 2}
```

```
model = KNeighborsClassifier()

# Tunning Params

param_grid = {

    'n_neighbors': list(range(1, 30)),

    'leaf_size': list(range(1,30)),

    'weights': [ 'distance', 'uniform' ]

}

# Implement GridSearchCV

gsc = GridSearchCV(model, param_grid, cv=10)

# Model Fitting

gsc.fit(X_train, y_train)

print("\n Best Score is ")

print(gsc.best_score_)

print("\n Best Estinator is ")

print(gsc.best_estimator_)

print("\n Best Parametes are")

print(gsc.best_params_)
```



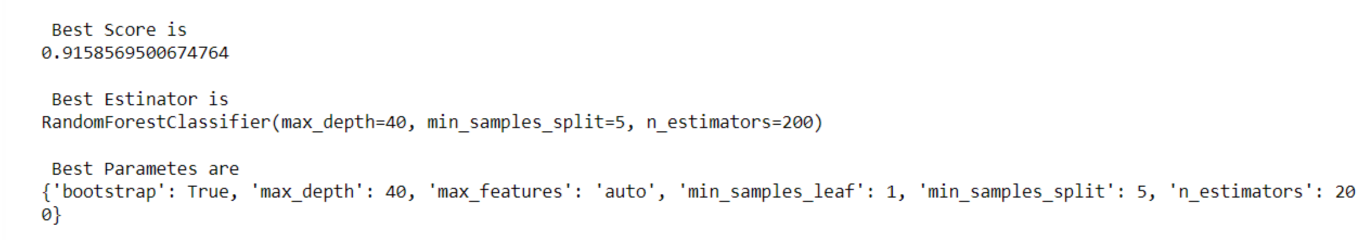
```
 Best Score is
0.9159244264507423

 Best Estinator is
KNeighborsClassifier(leaf_size=1, n_neighbors=10)

 Best Parametes are
{'leaf_size': 1, 'n_neighbors': 10, 'weights': 'uniform'}
```

```
model = SVC()
param_grid = [
            {'C': [1, 10, 100, 1000],
             'kernel': ['linear']
            },
            {'C': [1, 10, 100, 1000],
             'gamma': [0.001, 0.0001],
             'kernel': ['rbf']
            }
]
gsc = GridSearchCV(model, param_grid, cv=10) # 10 Cross Validation
# Model Fitting
gsc.fit(X_train, y_train)
print("\n Best Score is ")
print(gsc.best_score_)
print("\n Best Estinator is ")
print(gsc.best_estimator_)
print("\n Best Parametes are")
print(gsc.best_params_)
```



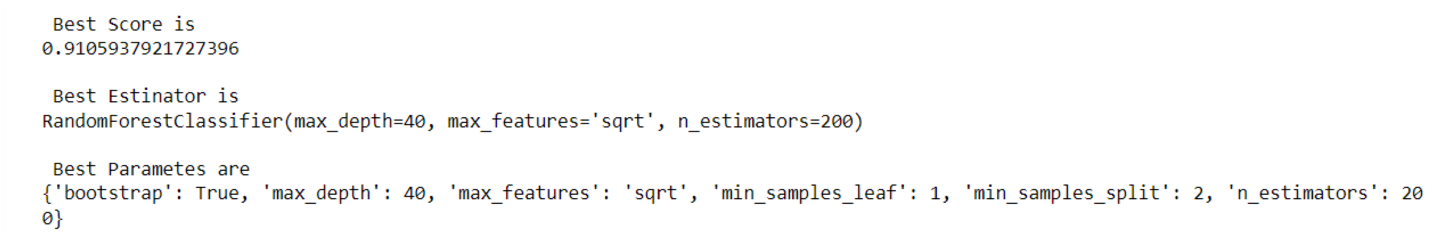
```
 Best Score is 
0.9184885290148447

 Best Estinator is 
SVC(C=10, gamma=0.001)

 Best Parametes are
{'C': 10, 'gamma': 0.001, 'kernel': 'rbf'}
```

```
model = RandomForestClassifier()
random_grid = {'bootstrap': [True, False],
 'max_depth': [40, 50, None], # 10, 20, 30, 60, 70, 100,
 '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
# Implement GridSearchCV
gsc = GridSearchCV(model, random_grid, cv=10) # 10 Cross Validation
# Model Fitting
gsc.fit(X_train, y_train)
print("\n Best Score is ")
print(gsc.best_score_)
print("\n Best Estinator is ")
print(gsc.best_estimator_)
print("\n Best Parametes are")
print(gsc.best_params_)
```



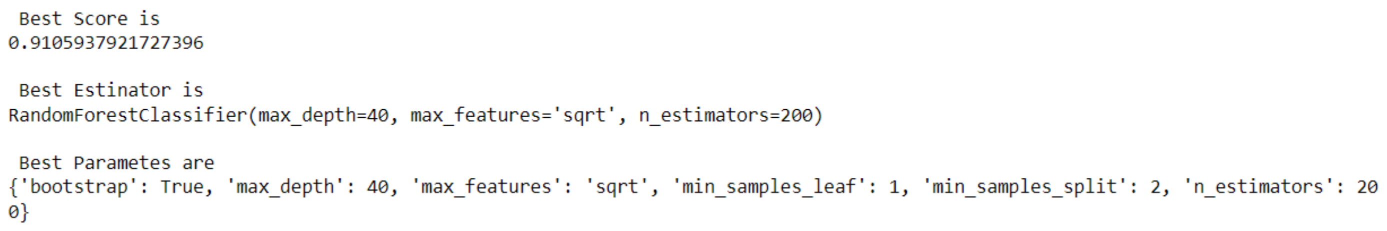
```
 Best Score is
0.9158569500674764

 Best Estinator is
RandomForestClassifier(max_depth=40, min_samples_split=5, n_estimators=200)

 Best Parametes are
{'bootstrap': True, 'max_depth': 40, 'max_features': 'auto', 'min_samples_leaf': 1, 'min_samples_split': 5, 'n_estimators': 20
0}
```

```
random_grid = {'bootstrap': [True, False],
 'max_depth': [40, 50, None], # 10, 20, 30, 60, 70, 100,
 '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
# Implement GridSearchCV
gsc = GridSearchCV(model, random_grid, cv=10) # 10 Cross Validation
# Model Fitting
gsc.fit(X_train, y_train)
print("\n Best Score is ")
print(gsc.best_score_)
print("\n Best Estinator is ")
print(gsc.best_estimator_)
print("\n Best Parametes are")
print(gsc.best_params_)
```

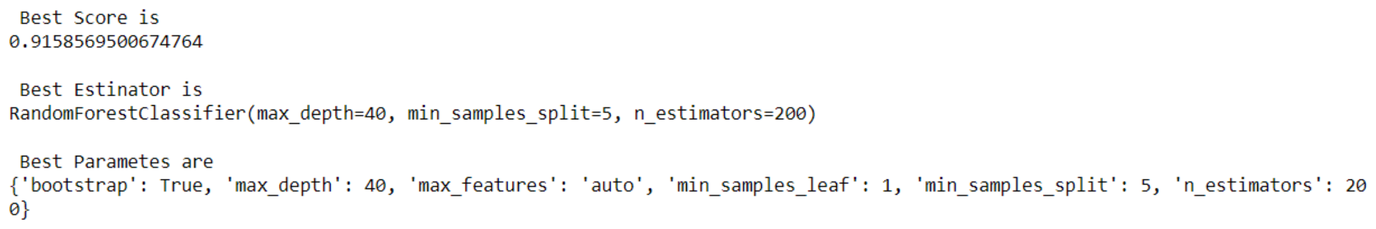


```
 Best Score is 
0.9105937921727396

 Best Estinator is 
RandomForestClassifier(max_depth=40, max_features='sqrt', n_estimators=200)

 Best Parametes are
{'bootstrap': True, 'max_depth': 40, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 20
0}
```

APPENDIX 2



```
 Best Score is
0.9105937921727396

 Best Estinator is
RandomForestClassifier(max_depth=40, max_features='sqrt', n_estimators=200)

 Best Parametes are
{'bootstrap': True, 'max_depth': 40, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 20
0}
```



```
 Best Score is
0.9158569500674764

 Best Estinator is
RandomForestClassifier(max_depth=40, min_samples_split=5, n_estimators=200)

 Best Parametes are
{'bootstrap': True, 'max_depth': 40, 'max_features': 'auto', 'min_samples_leaf': 1, 'min_samples_split': 5, 'n_estimators': 20
0}
```

CONCLUSION:

This work reports the impact of dominance-based filtering approach on performances of major state-of-the-art classifiers used in machine learning paradigm. WBCD database was utilized for RH, 5-fold and 10-fold cross validation protocols using sensitivity, specificity, accuracy and AUC parameters. ANN has emerged as the best classifier with classification accuracies of 98.9% for four dominant features (feature set #4), 99.6% for five dominant features (feature set #5) and 99.7% for all features (feature set #9) subjected to the 10-fold cross validation protocol. Also, though dominance-based filtering technique is computationally fast, the feature rank is algorithm liberated. This can be prevented by adding an algorithm-relative block with proposed tactic for rank calculation. Furthermore, the proposed approach can be applied for other cancer datasets to test its generalized performance capabilities.A decision support system for predicting breast cancer helps and assist physician in making optimum, accurate and timely decision, and reduce the overall cost of treatment. Different classifiers have been used to conduct experiments on the standard WBCD. It is been observed KNN classifier yields the highest classification accuracies when used with most predictive variables. The proposed system greatly reduces the cost of treatment and improves the quality of life by predicting breast cancer at early stage of development. The future work will focus on exploring more of the dataset values and yielding more interesting outcomes. This study can help in making more effective and reliable disease prediction and diagnostic system which will contribute towards developing better healthcare system by reducing overall cost, time and mortality rate.

REFERENCES

1. Jemal A, Murray T, Ward E, Samuels A, Tiwari RC, Ghafoor A, Feuer EJ, Thun MJ. Cancer statistics, 2005. CA: a cancer journal for clinicians. 2005 Jan 1;55(1):10-30.
2. Polat K, Güneş S. Breast cancer diagnosis using least square support vector machine. Digital Signal Processing. 2007 Jul 1;17(4):694- 701.
3. Akay MF. Support vector machines combined with feature selection for breast cancer diagnosis. Expert systems with applications. 2009 Mar 1;36(2):3240-7.
4. Yeh WC, Chang WW, Chung YY. A new hybrid approach for mining breast cancer pattern using discrete particle swarm optimization and statistical method. Expert Systems with Applications. 2009 May 1;36(4):8204-11.
5. Marcano-Cedeño A, Quintanilla-Domínguez J, Andina D. WBCD breast cancer database classification applying artificial metaplasticity neural network. Expert Systems with Applications. 2011 Aug 1;38(8):9573-9.
6. Kaya Y, Uyar M. A hybrid decision support system based on rough set and extreme learning machine for diagnosis of hepatitis disease. Applied Soft Computing. 2013 Aug 1;13(8):3429-38.
7. Nahato KB, Harichandran KN, Arputharaj K. Knowledge mining from clinical datasets using rough sets and backpropagation neural network. Computational and mathematical methods in medicine. 2015;2015.
8. Liu L, Deng M. An evolutionary artificial neural network approach for breast cancer diagnosis. InKnowledge Discovery and Data Mining, 2010. WKDD'10. Third International Conference on 2010 Jan 9 (pp. 593-596). IEEE.
9. Chen HL, Yang B, Liu J, Liu DY. A support vector machine classifier with rough set-based feature selection for breast cancer diagnosis. Expert Systems with Applications. 2011 Jul 1;38(7):9014-22.