**PREDICTING Malignancy of  
 BREAST CANCER**

**Group Members:**

|  |
| --- |
| Susmita Goswami,GNIT, regno-171430110112  Sayan Chowdhury ,GNIT,regno-171430110085  Sankha Mondal ,GNIT, regno-171430110083  UdayadityaSasmal,GNIT,regno-171430110122  Parlay Ghosh ,GNIT, regno-171430110065  Anik Dutta ,GNIT, regno-17143011012 |

**Project Objective:**

The project focuses on a prediction model to predict malignancy of breast cancer i.e. if a tumor is of malignant type or benign type based on 30 features of the cell nuclei. A dataset of 569 records that we have used is a sample dataset where we are assuming all the data in the dataset are correct and sufficient enough for our model. The dataset is used for the training, test, and cross validation steps for our Learner Model by inductive learning.

All available predictor versions are included in our model (Mean, Standard Error, and Worst) for the 15 real valued predictors (Radius, Texture, Perimeter, Area, Smoothness, Compactness, Concavity, Concave Points, Symmetry, and Fractal Dimension). We have used machine learning classification method to fit a function that can predict the discreet class of new input features.

**Project scope:**

Breast cancer is the most common cancer in women. It is estimated that worldwide over 5,08,000 women died in 2011 due to breast cancer (Global Health Estimates, WHO 2013). Although breast cancer is thought to be a disease of the developed world, almost 50% of breast cancer cases and 58% of deaths occur in less developed countries (GLOBOCAN 2008).

Breast cancer is detected by Mammogram which costs between 1,500 Rs to 8,000 Rs and takes around 15 minutes to detect malignancy.

But our predicting model can detect malignancy within 5 mins at a low cost. Because of its unique advantages in critical features detection from complex Breast Cancer datasets, this model can be used in breast cancer pattern classification and forecasting.

**Data Description:**

Diagnosis (M = malignant, B = benign)

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

a) radius (mean of distances from centre to points on the perimeter)

b) texture (standard deviation of gray-scale values)

c) perimeter

d) area

e) smoothness (local variation in radius lengths)

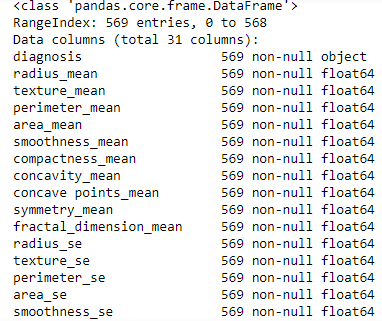
f) compactness (perimeter^2 / area - 1.0)

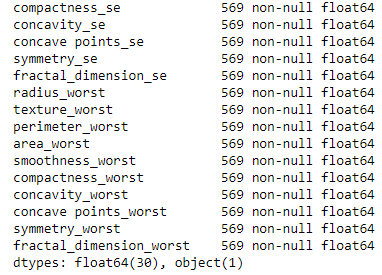
g) concavity (severity of concave portions of the contour)

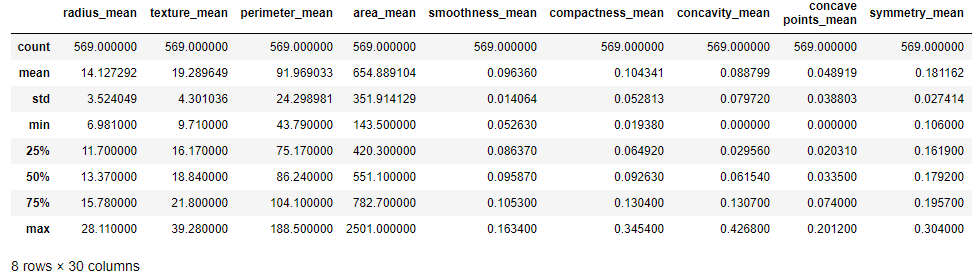
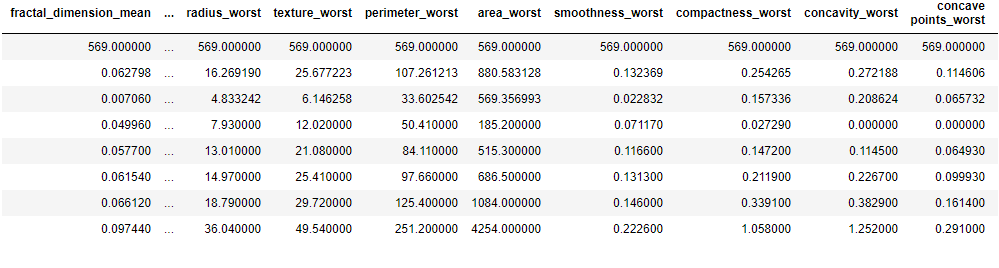
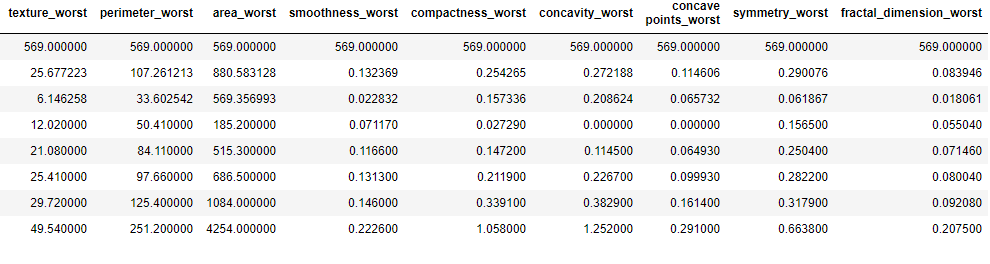
h) concave points (number of concave portions of the contour)

i) symmetry

j) fractal dimension

The mean, standard error and "worst" or largest (mean of the three largest values) of these features were computed for each image, resulting in 30 features e.g. column 3 is Mean Radius, column 13 is Radius-SE, column 23 is Worst Radius.

****

****

**Exploratory Data Analysis(EDA):**

We need to do EDA in order to understand types of data, removing skewness, outliers treatment and transformation of the data(if required). For the following analysis we’ve to go through some steps.

1. **Univariate Analysis:**

At first we import pandas module to read dataset(.csv file).In dataset there is a column named ‘Unnamed: 32’ which is full of NaN values.

A screenshot of a social media post

Description automatically generated

**A screenshot of a cell phone

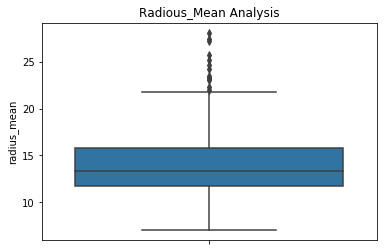
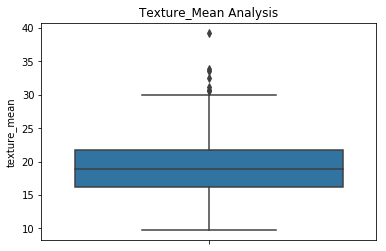
Description automatically generated**So, we droped the column using drop() function.A screenshot of a social media post

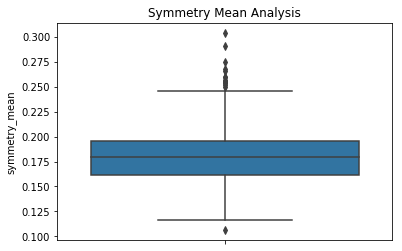
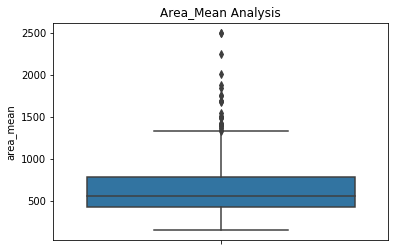
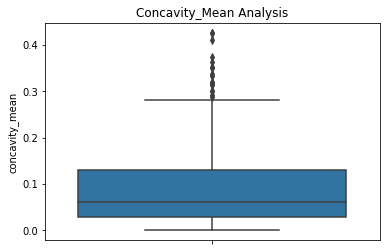
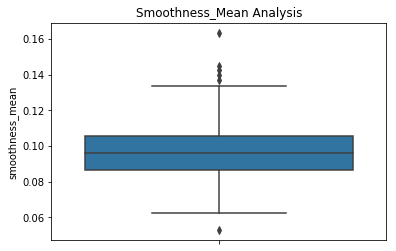
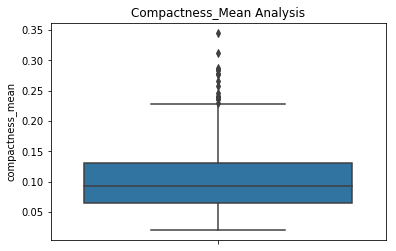
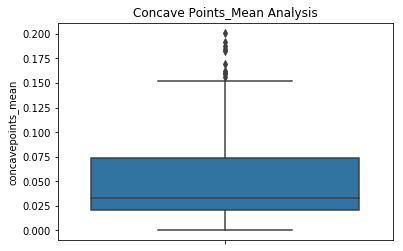
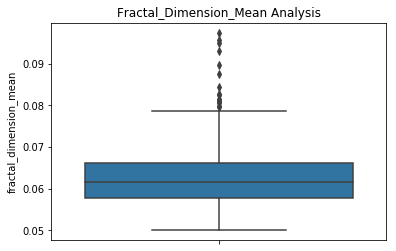
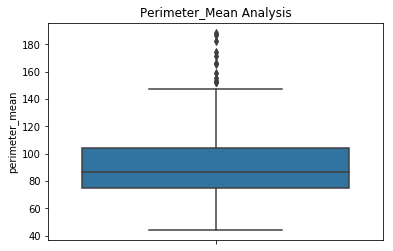
Description automatically generated

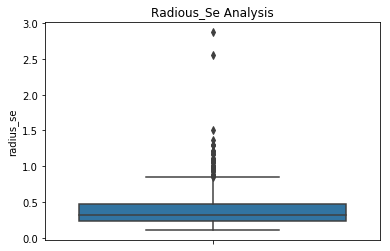
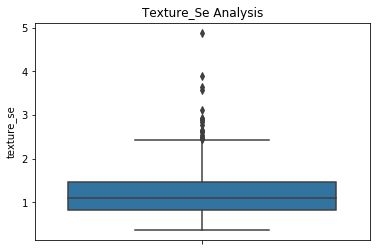
Now, we will plot boxplot to determine the outliers of the input variables. In order to do that, here we’ve to import ‘seaborn’ module as sns and ‘matplotlib.pyplot’ module as plt.

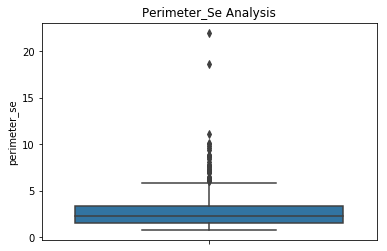
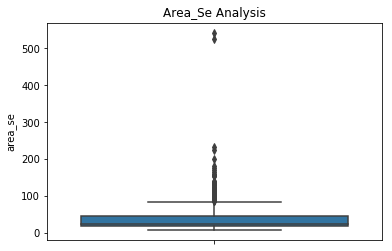
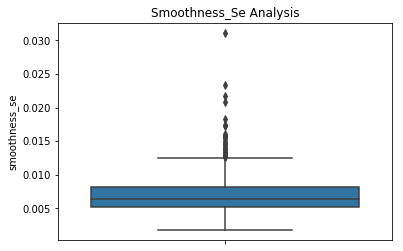
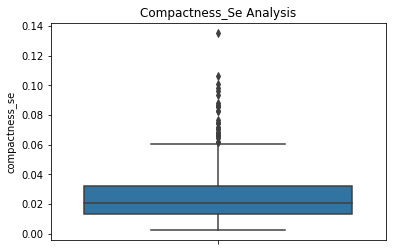
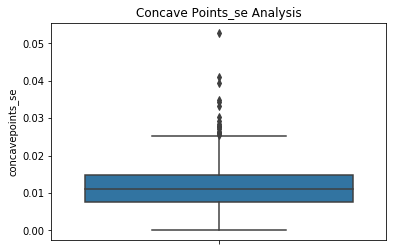
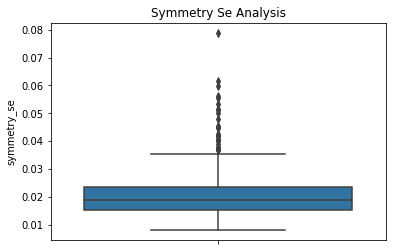
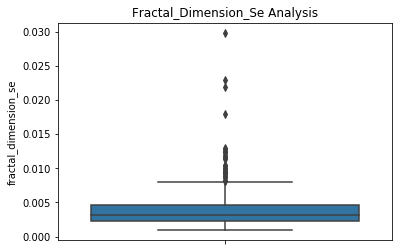
import seaborn as sns  
 import matplotlib.pyplot as plt  
 %matplotlib inline #this will show the plot in same tab

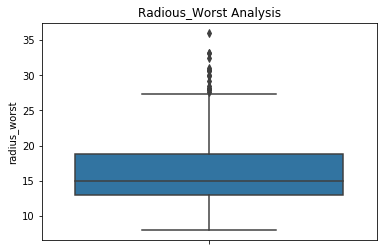
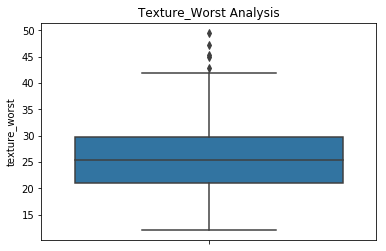
plt.title("Variable Name")  
 sns.boxplot(y=df.Variable Name)  
 plt.show()

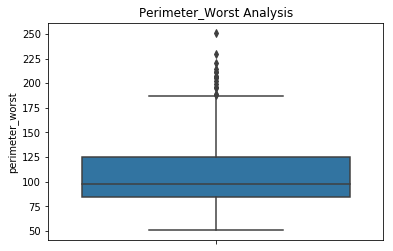
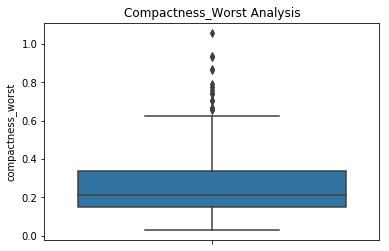
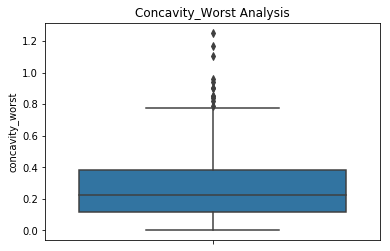
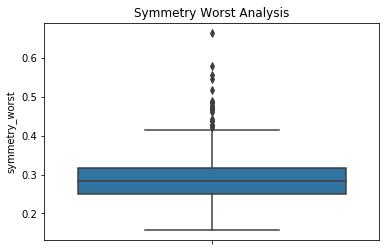
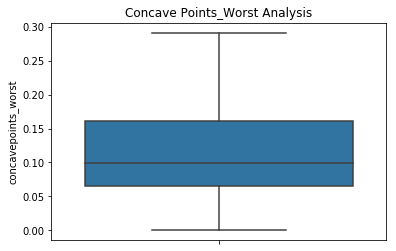
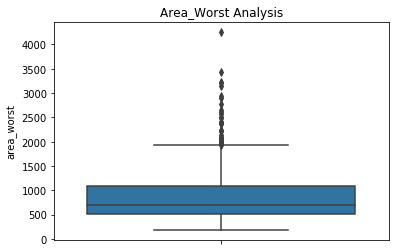
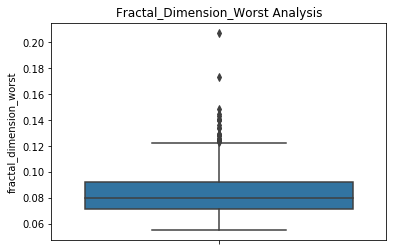
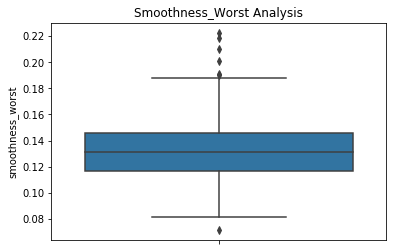












**1.1) Outliers Treatment:**

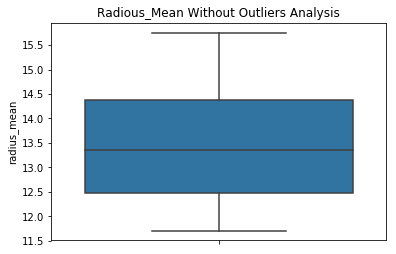
From the above boxplots we can see there are few numbers of outliers are present in every input feature. So, we need to first remove the outliers.

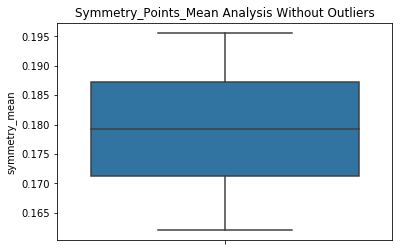
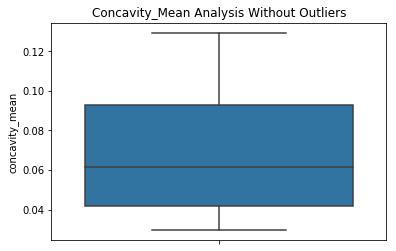
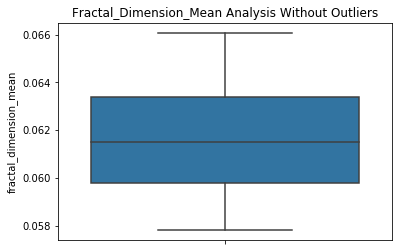
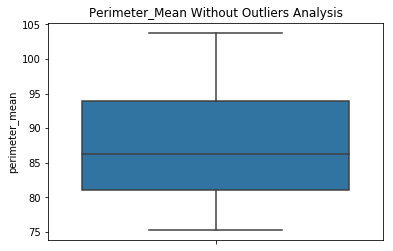
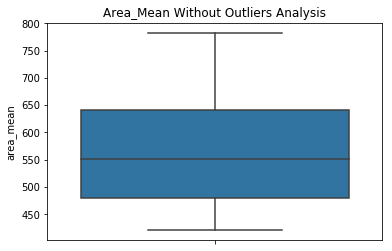
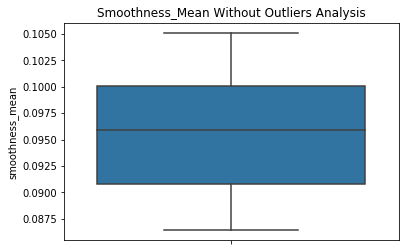
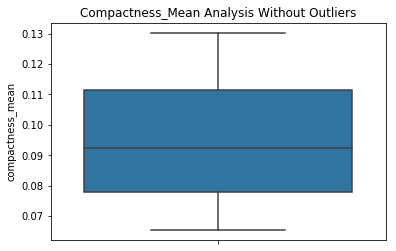
For that we should know the value of 1st Quartile and 3rd Quartile that means value at 25% and value at 75%. So, here we write the following code to describe the value of Quartiles.

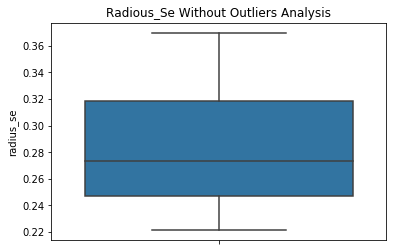
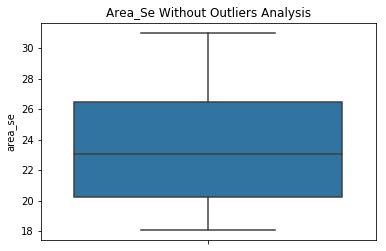
df[‘Variable Name’].describe()

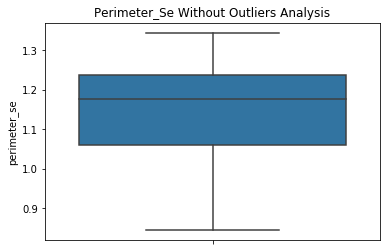
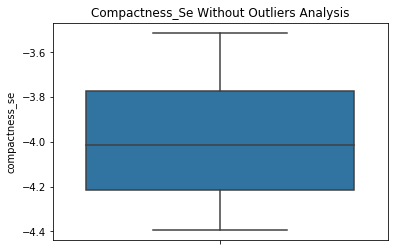
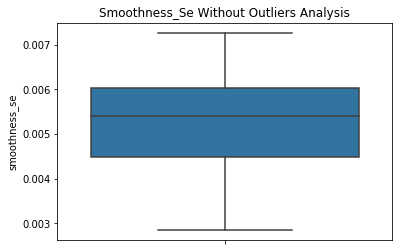
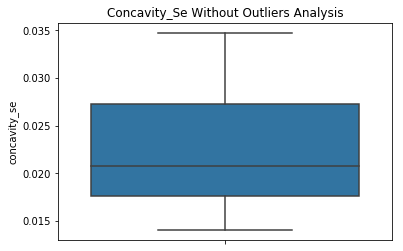
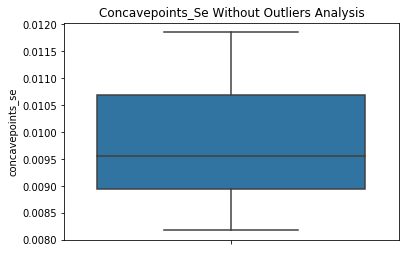
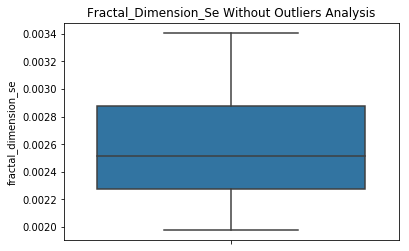
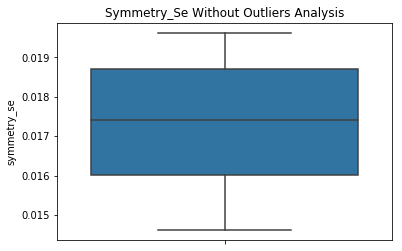
The value less than 25% and greater than 75% are known as outliers. In order to remove the outliers, we’ll use the code below:

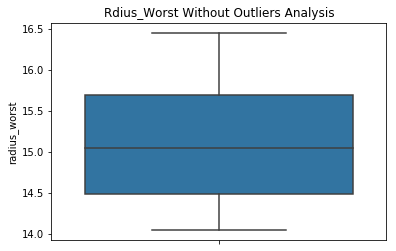
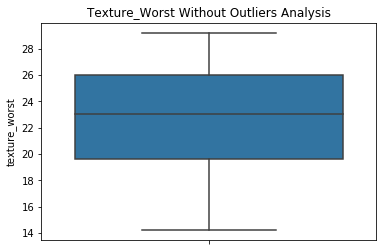
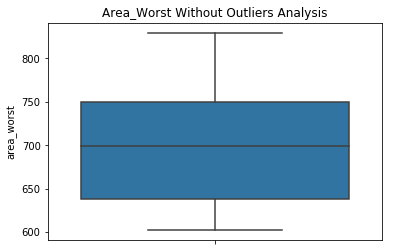
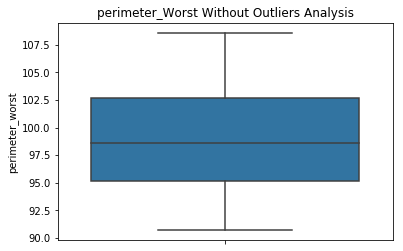
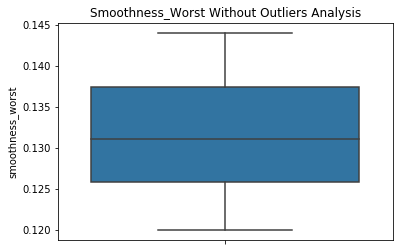
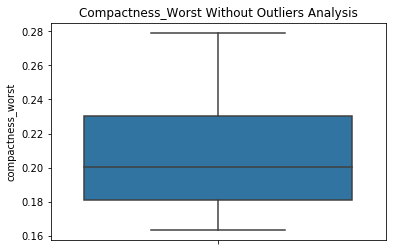
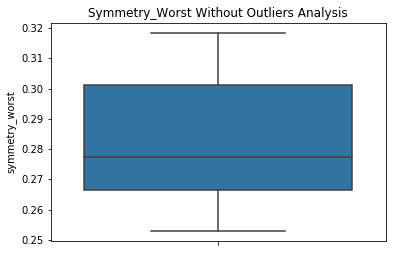
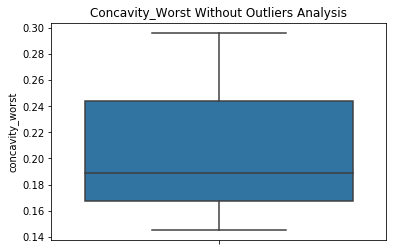
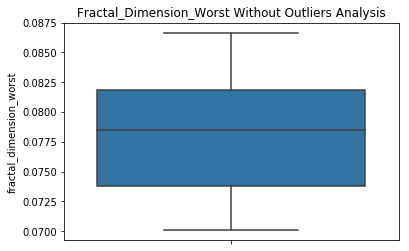
df\_out=df [(df. Variable Name<lb) & (df. Variable Name >ub)]

After removing outliers we again do the boxplots to see if any outliers are present or not .To remove the outliers we calculate the percentage the outliers statistically for each column .If the percentage is above 10% then we decide to remove the outliers and do remaining job .The boxplots without outliers are shown below :









**1.2) Skewness Analysis:**

It is the degree of distortion which can be determined from the symmetrical bell curve or from the normal distribution. It represents the lack of symmetry in data distribution. A symmetrical distribution will have a skewness of nearly zero.

A close up of a map

Description automatically generated

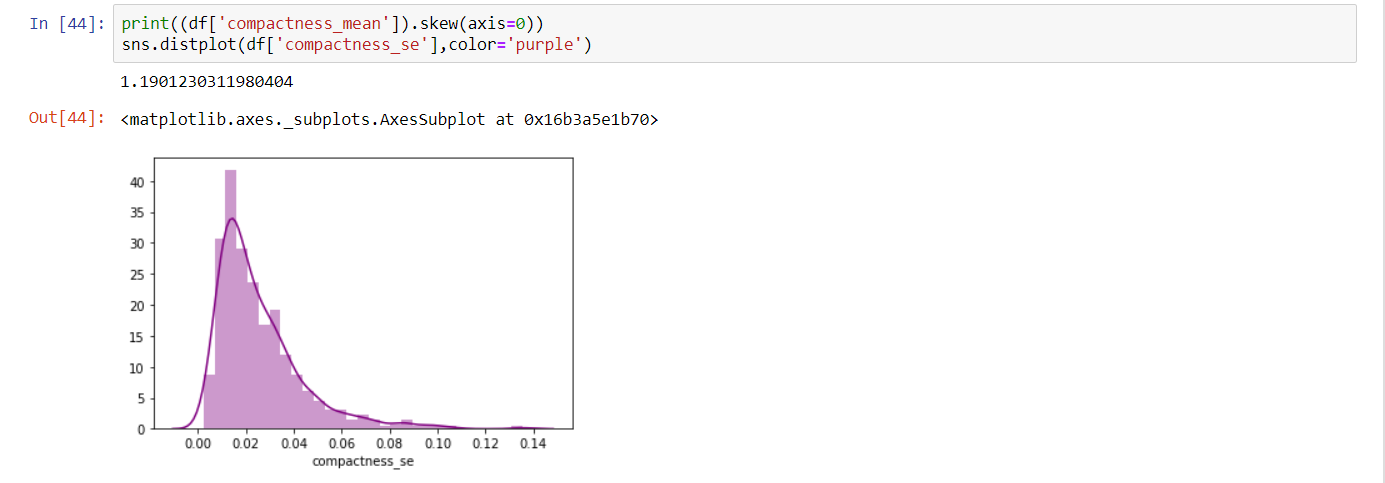
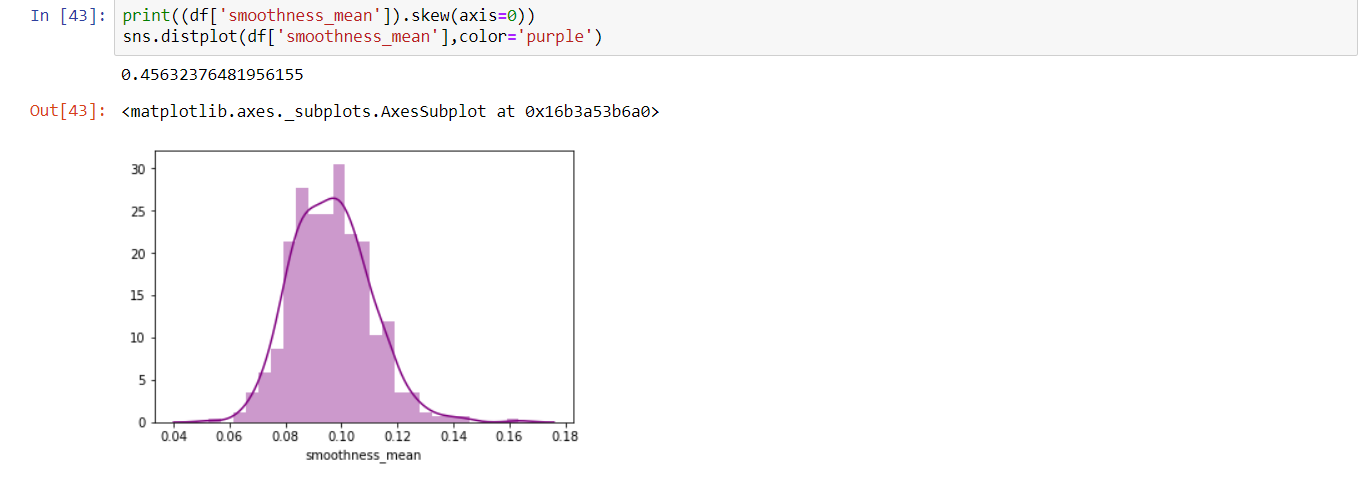
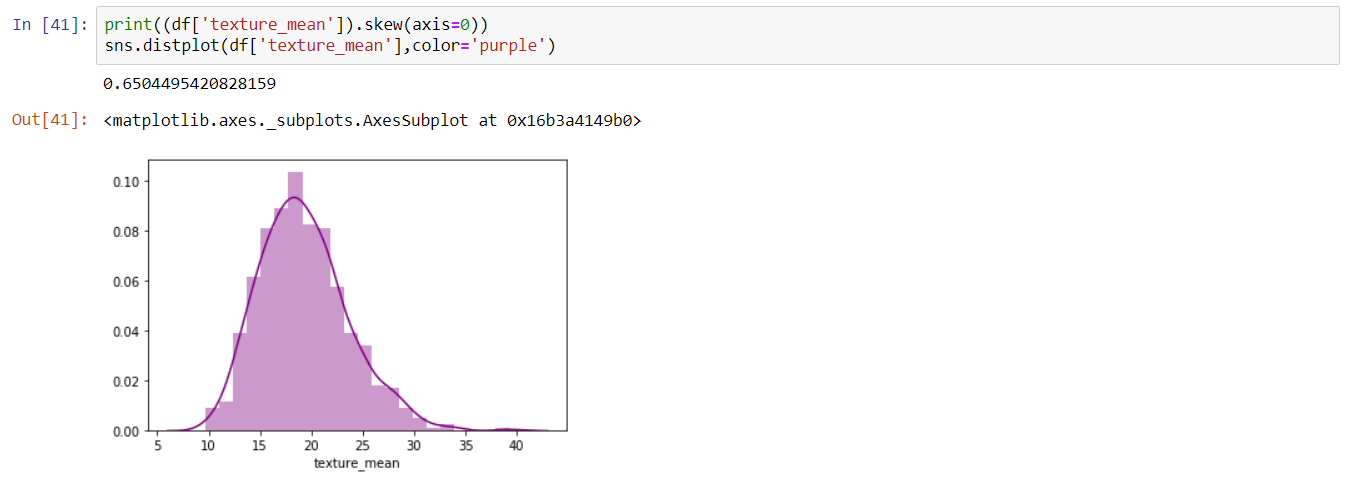
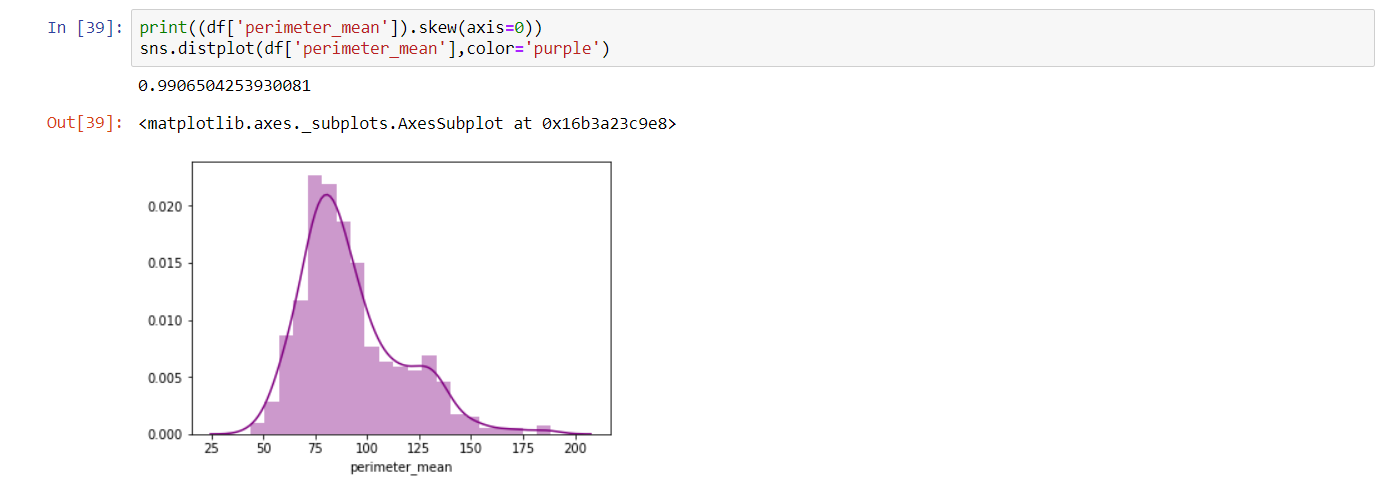
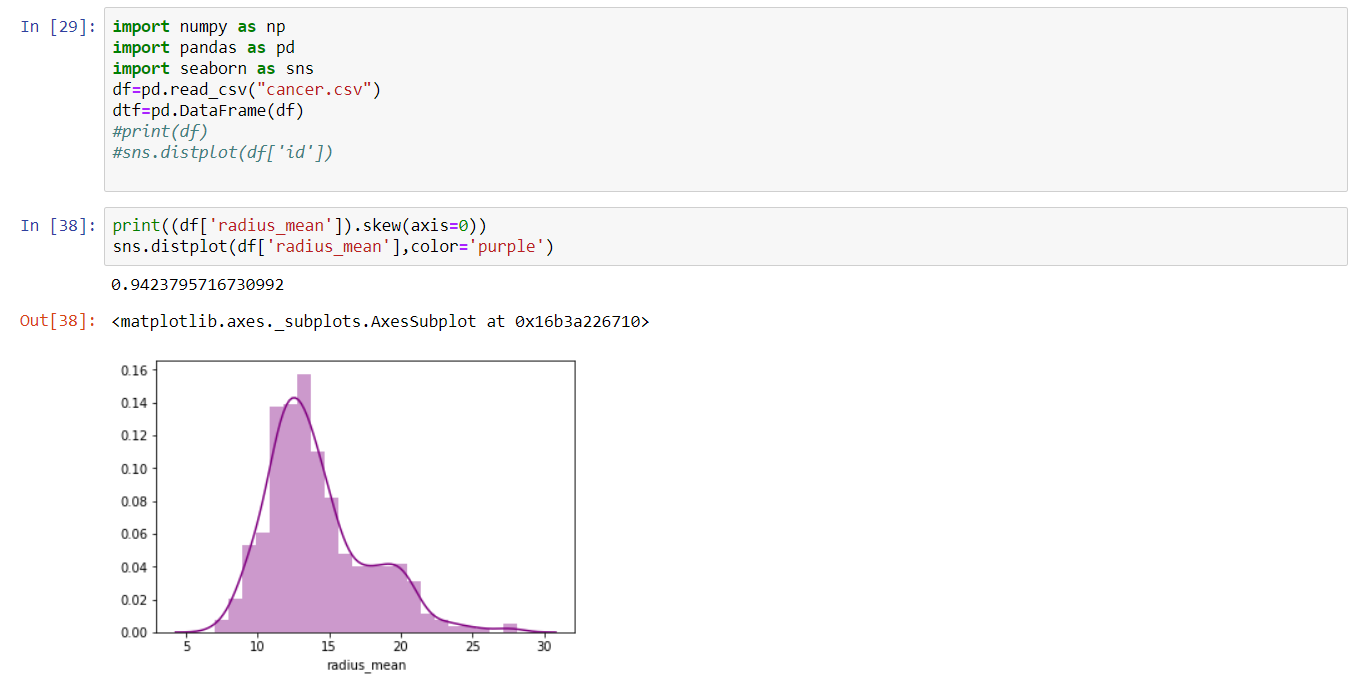
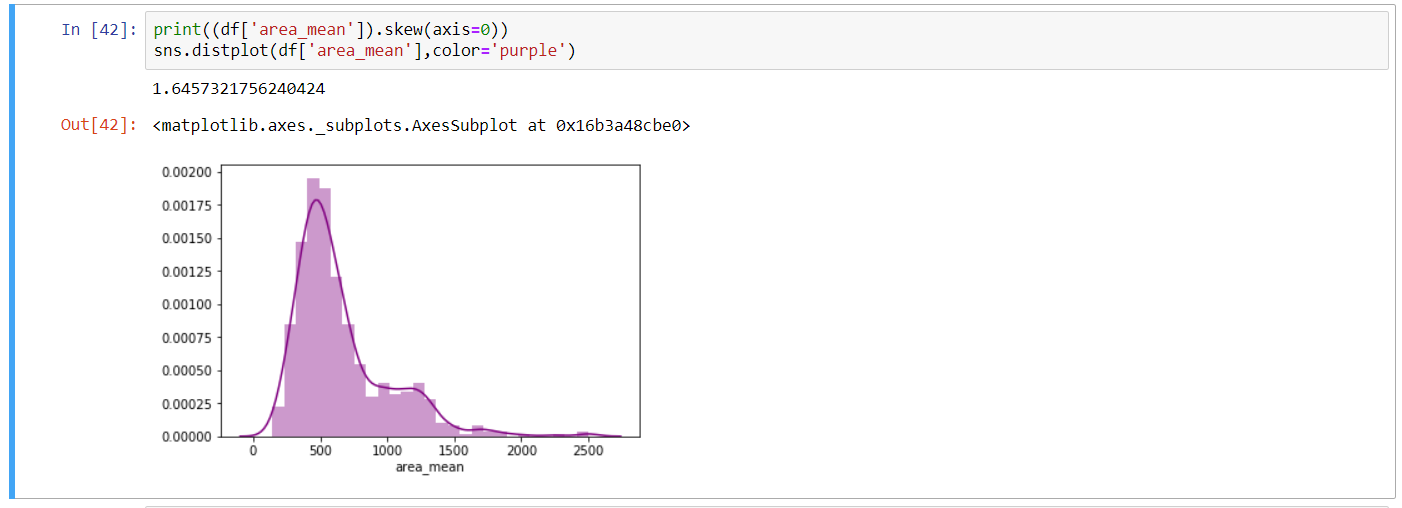
To determine the skewness (distribution of the data) of the variables we plot distribution plot (distplot).  
At first here we import ‘seaborn’ module, ‘matplotlib. pyplot’

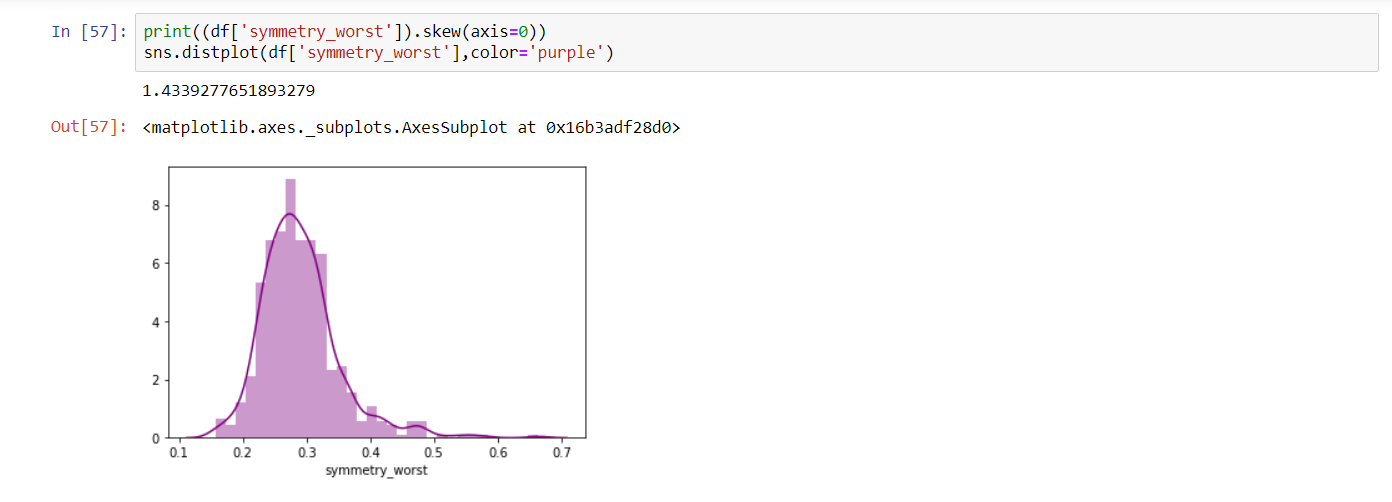
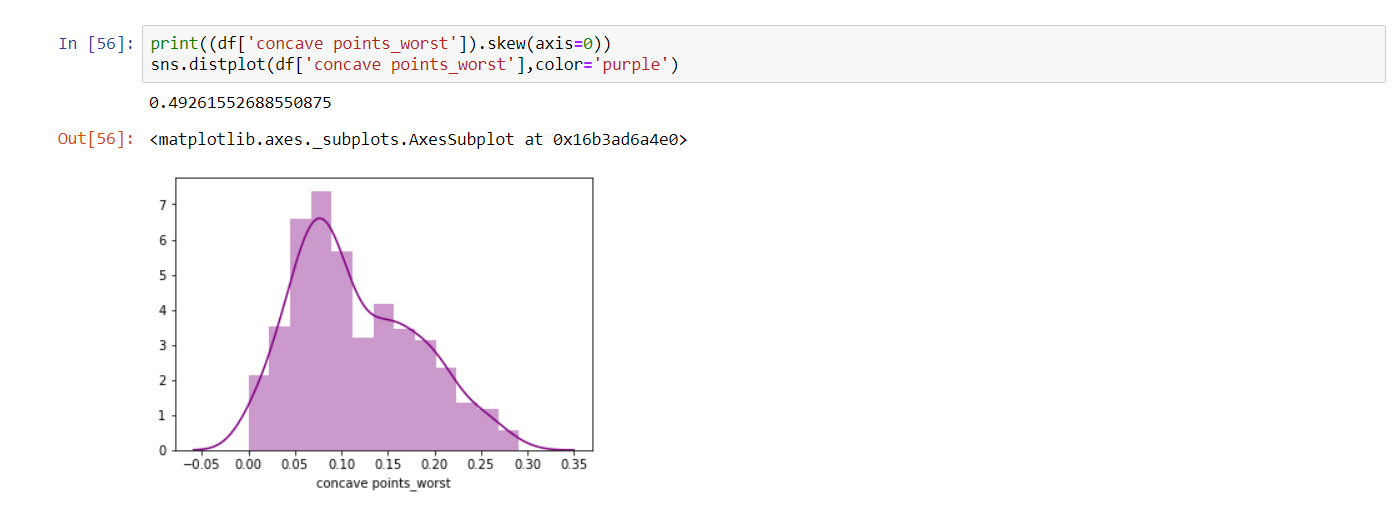
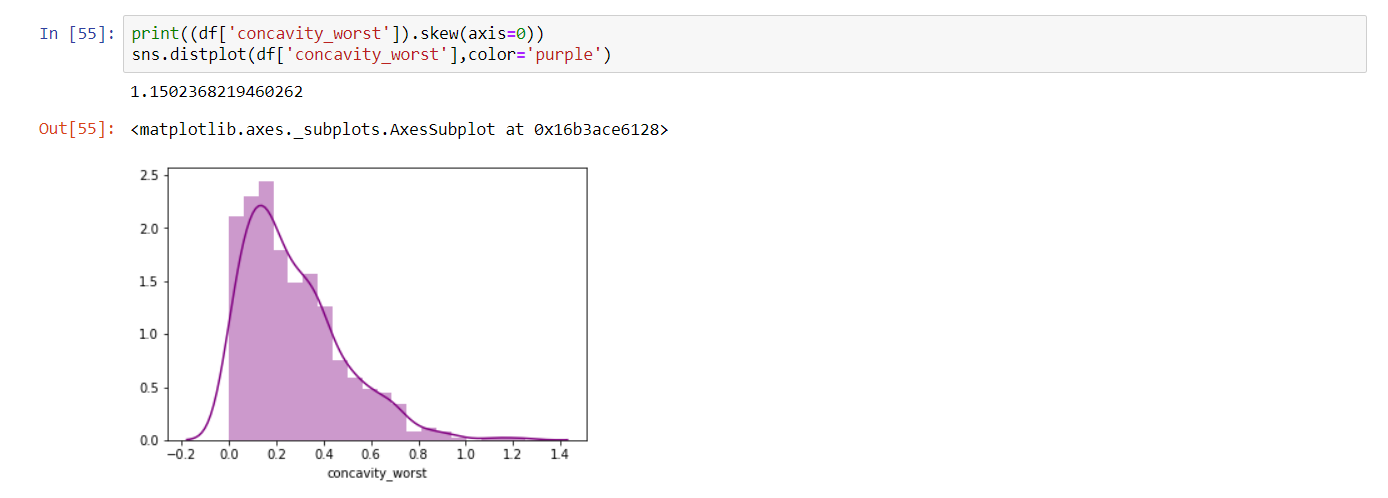
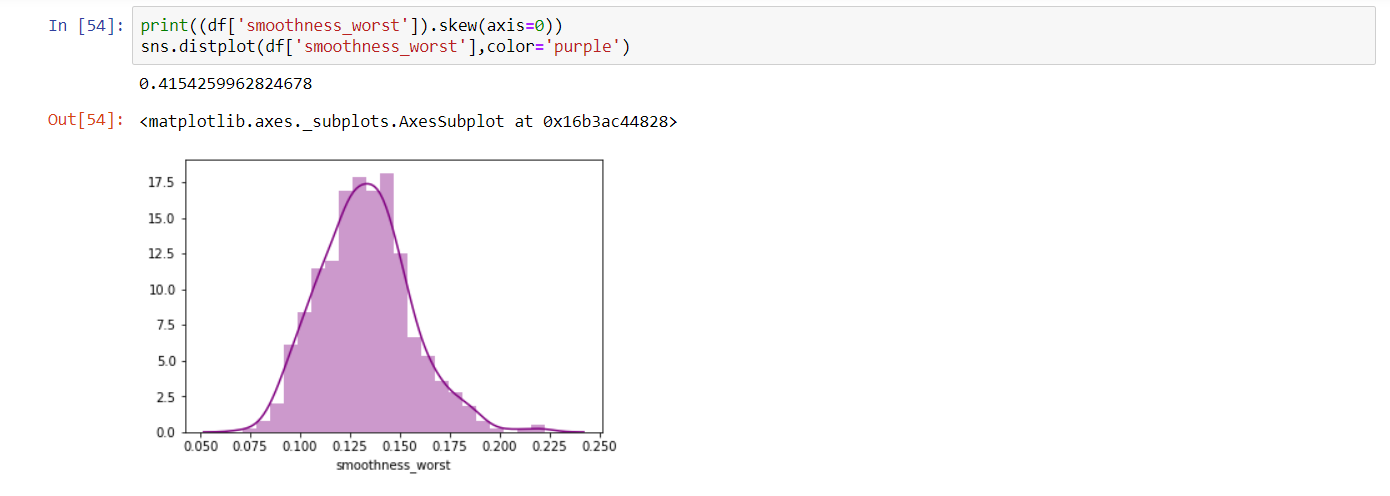
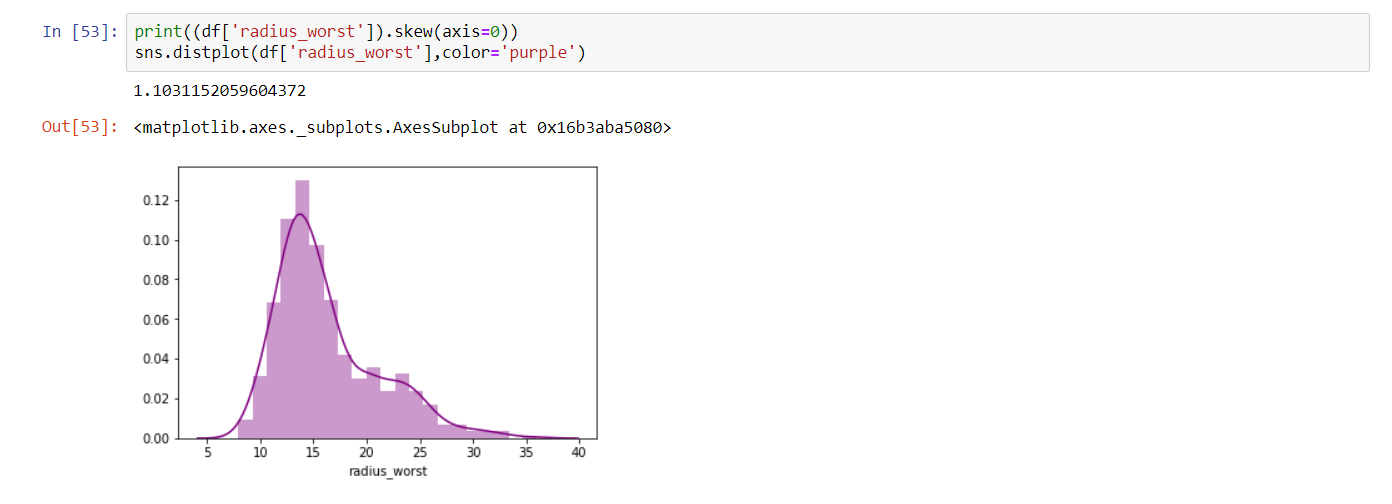
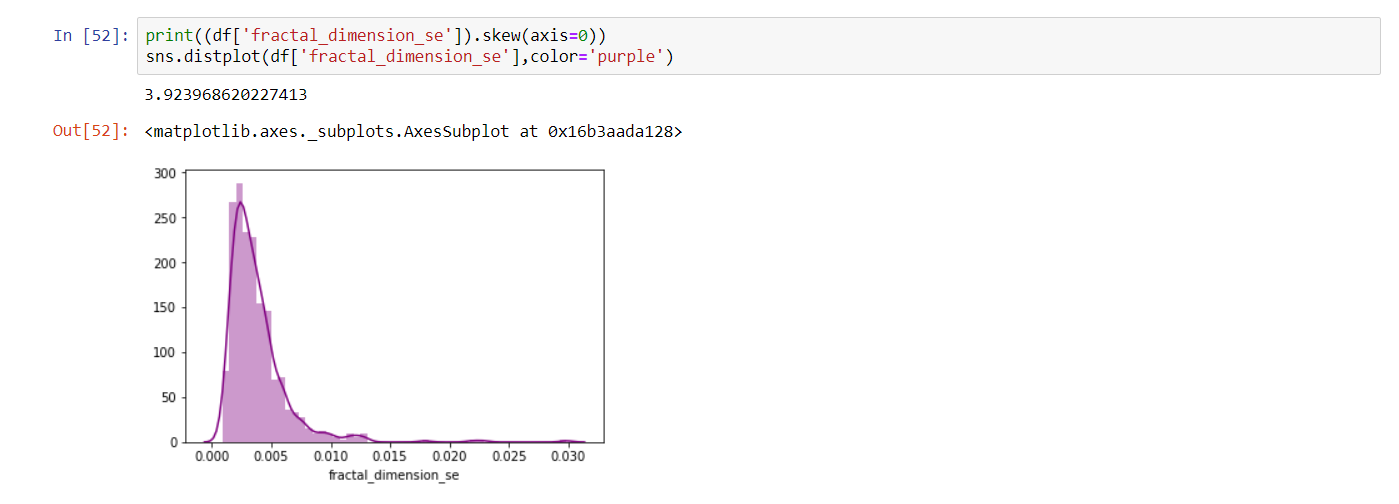
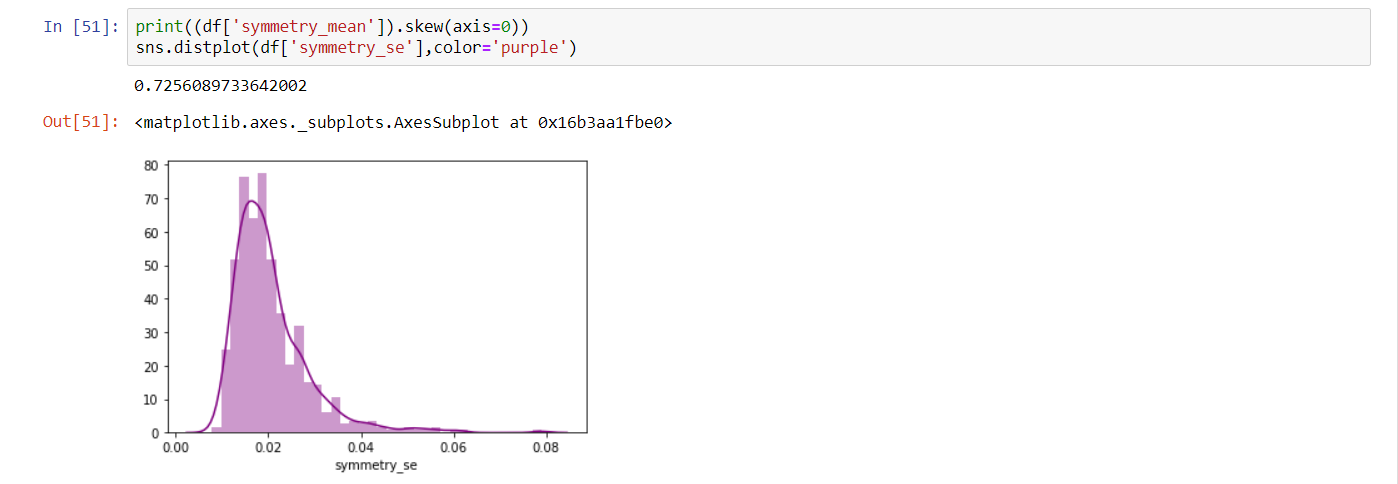
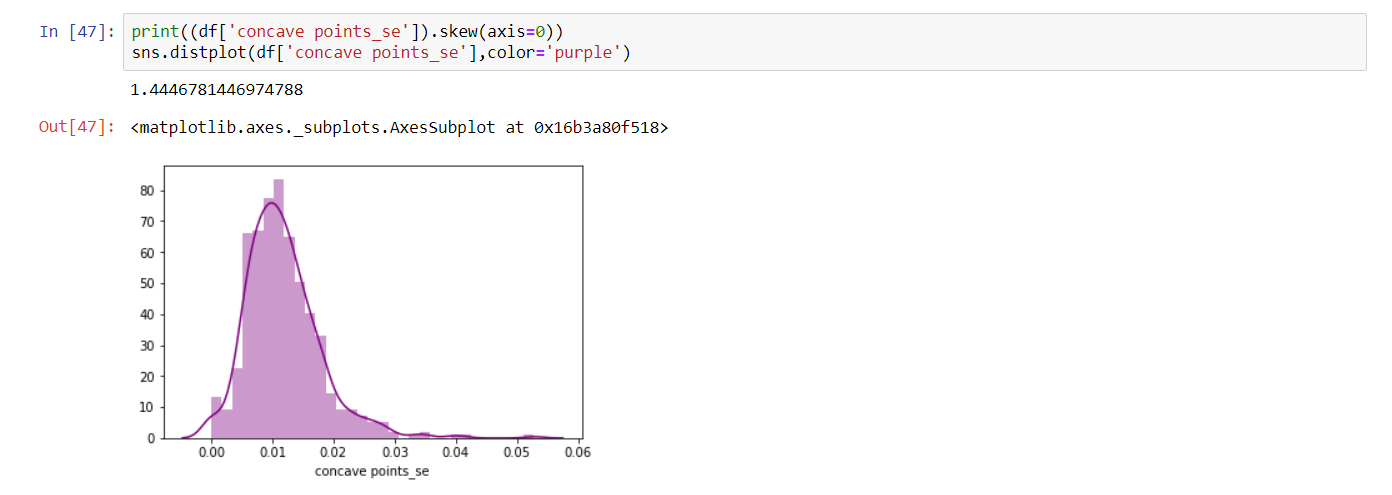
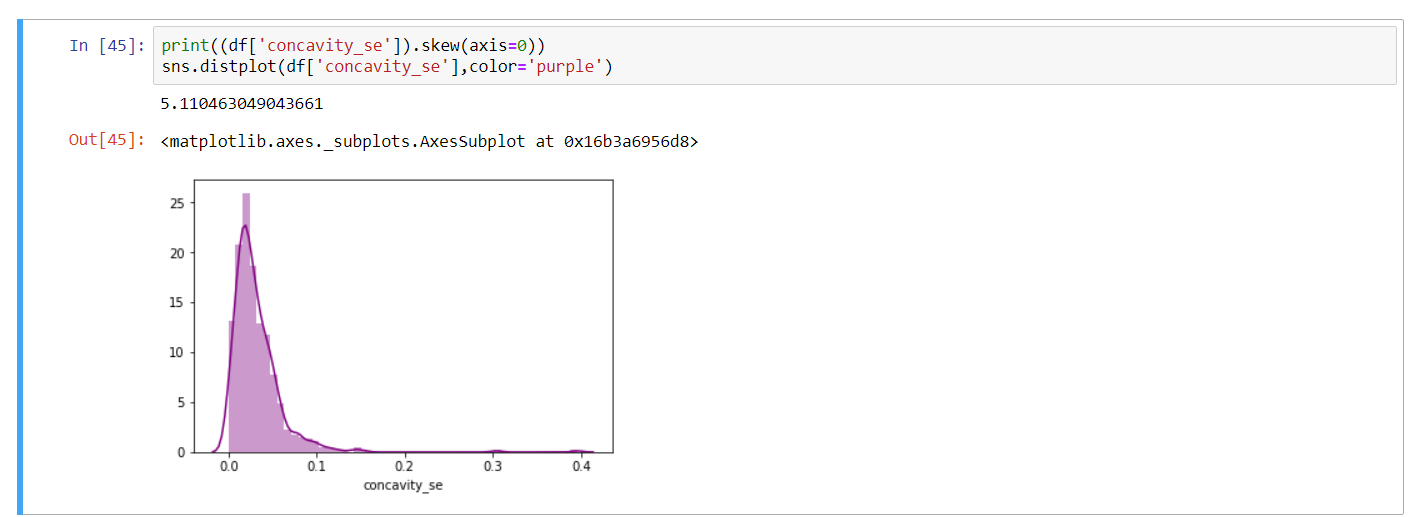
sns. distplot (df ['Variable Name'], color='maroon')

Any **threshold** value for skewness is arbitrary, but If the **skewness** is greater than 1.0 (or less than -1.0), the **skewness** is substantial, and the distribution is far from symmetrical.

To know the value of skewness for each variable we’ve to use the following code:

print((df[‘Variable Name’]).skew(axis=0))





**1.3) Removing Skewness:**

There are two techniques to remove the skewness of the data.

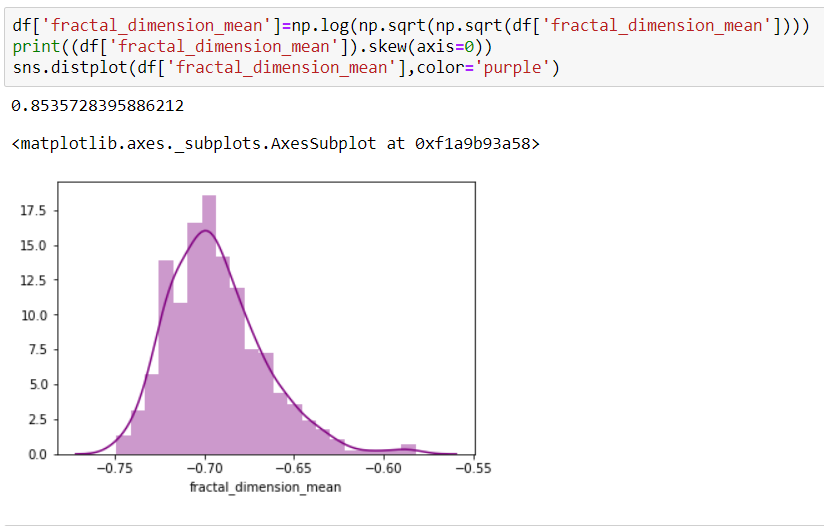
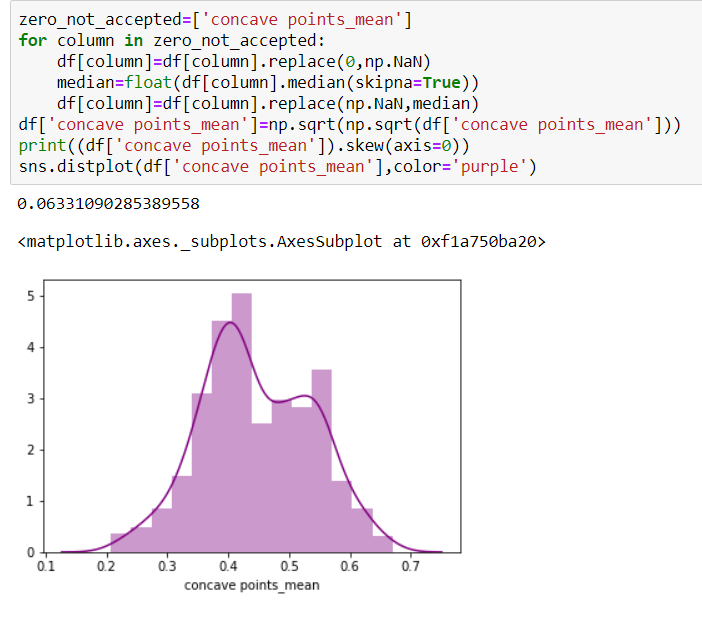
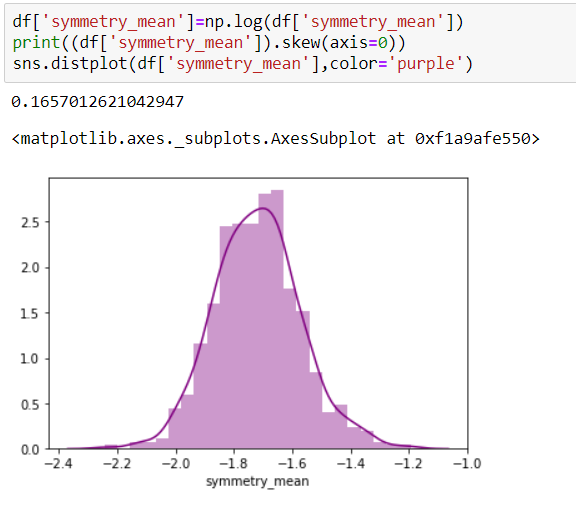
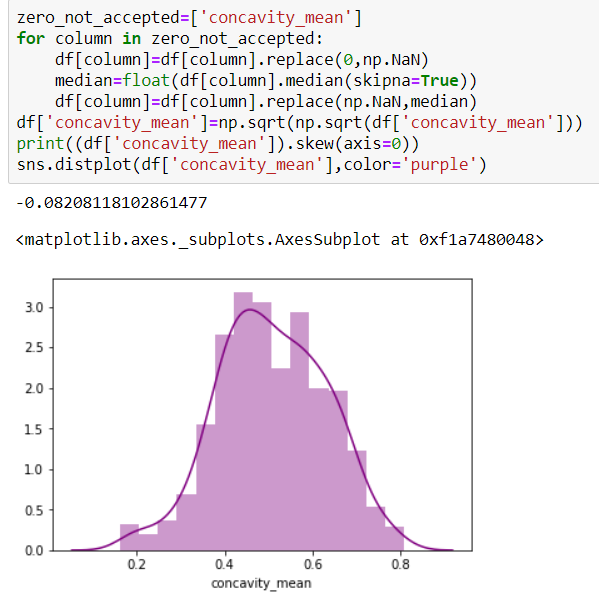
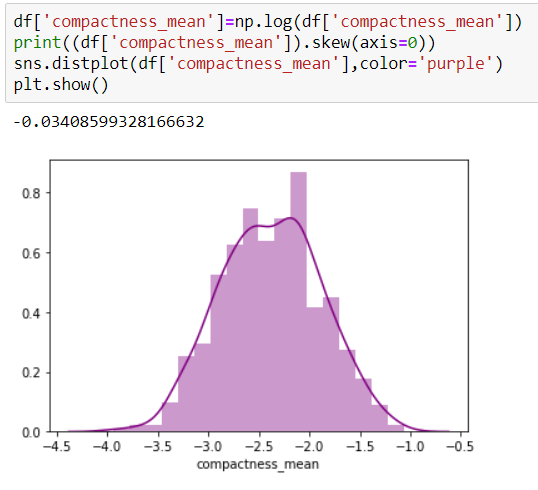
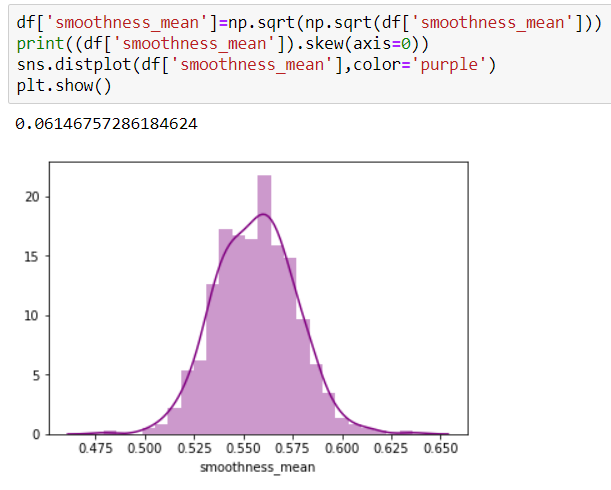
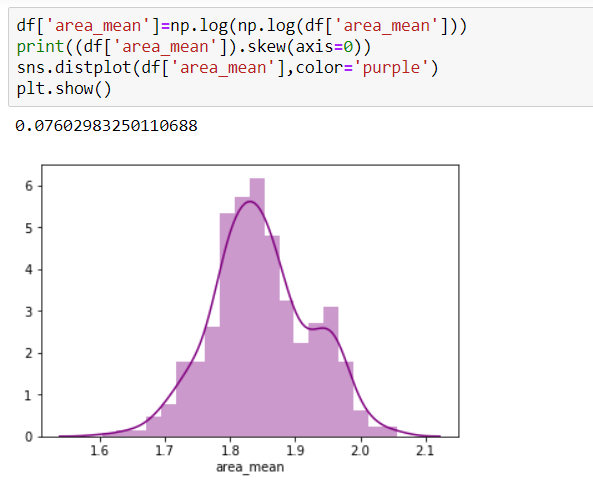
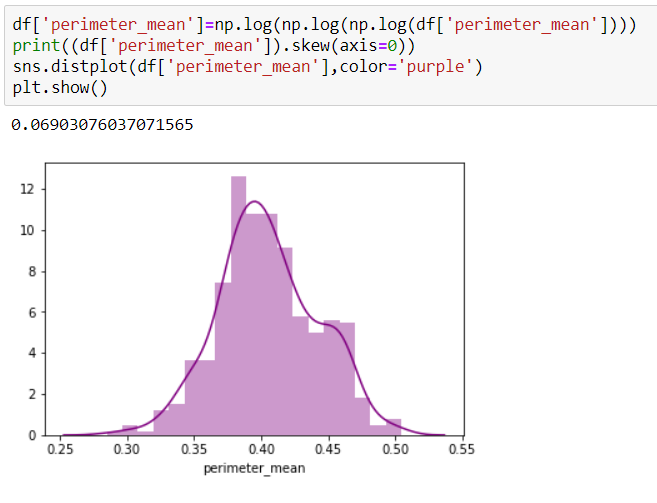
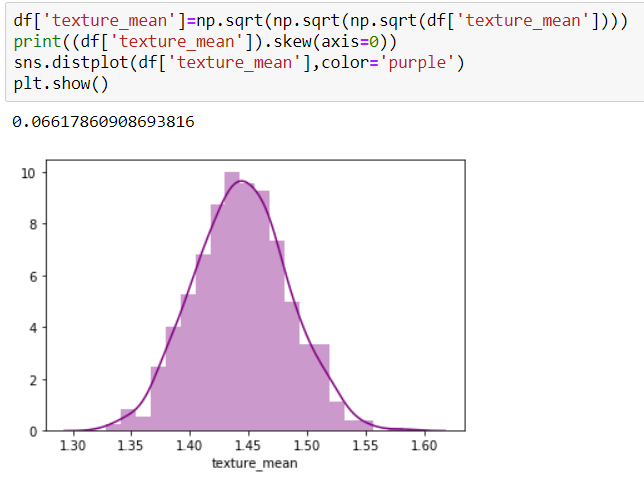
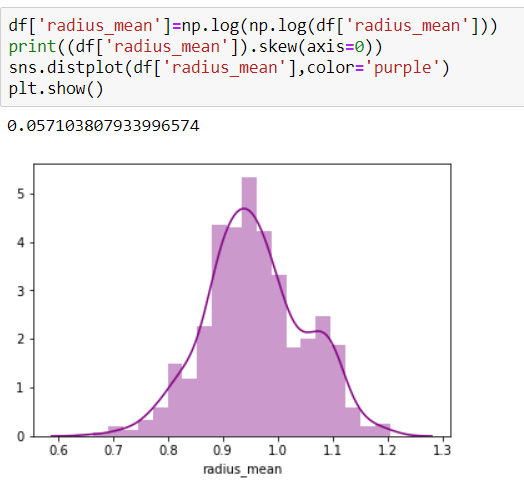
1. If the variables are left skewed we’ve to take logarithm or square root values of the entire columns.
2. Otherwise we’ve to take square and cube values of that variable

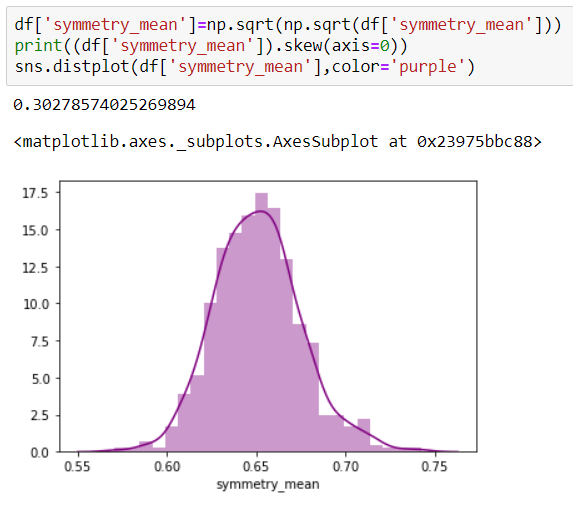
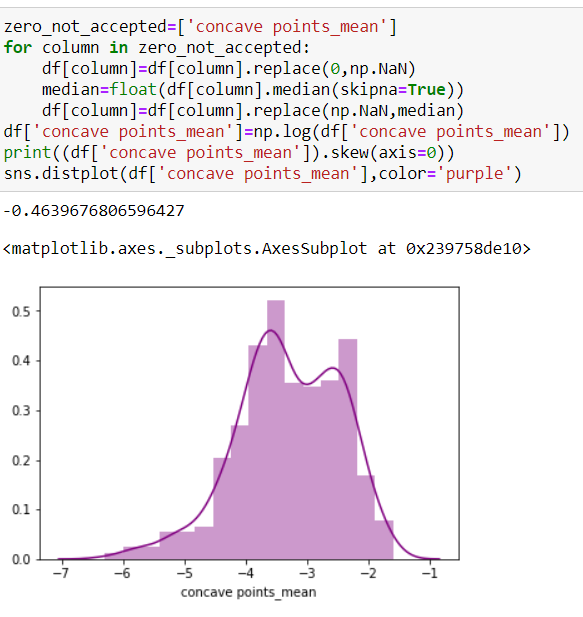
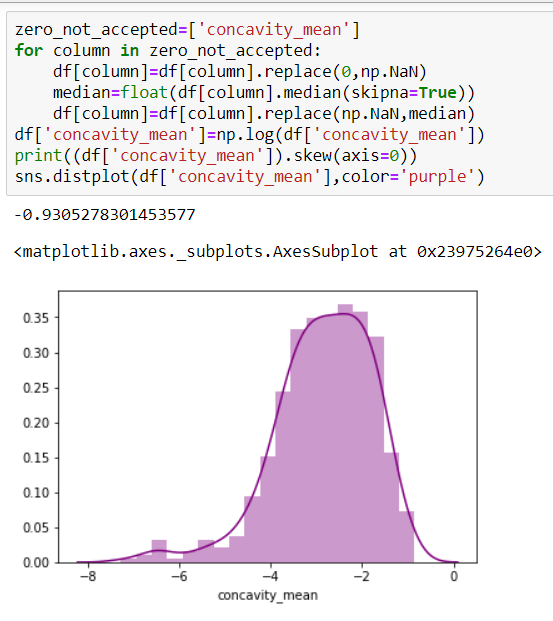
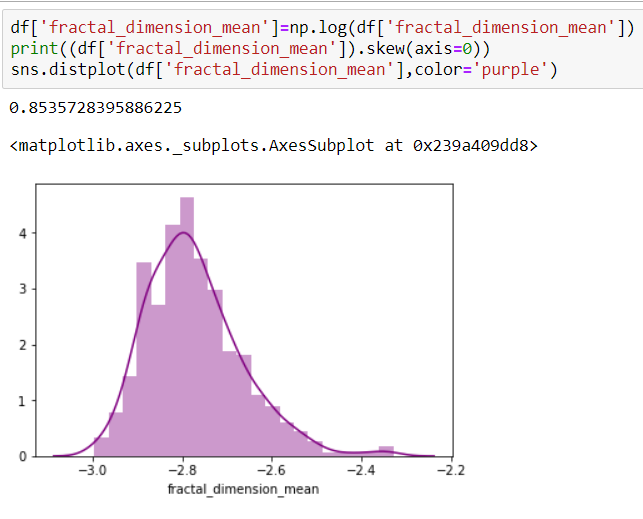
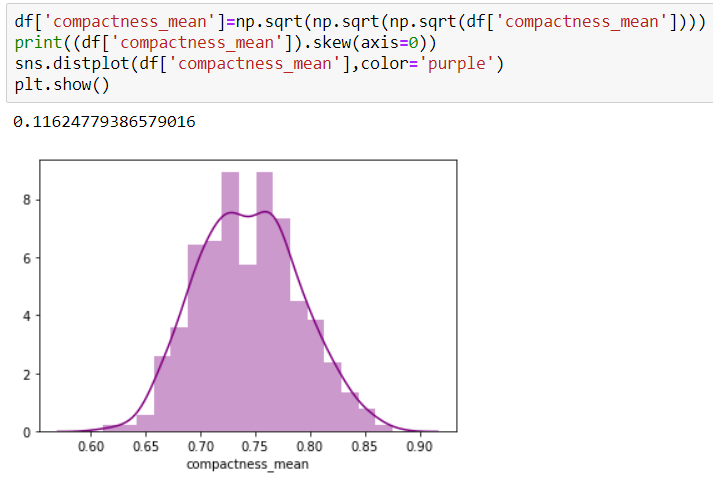
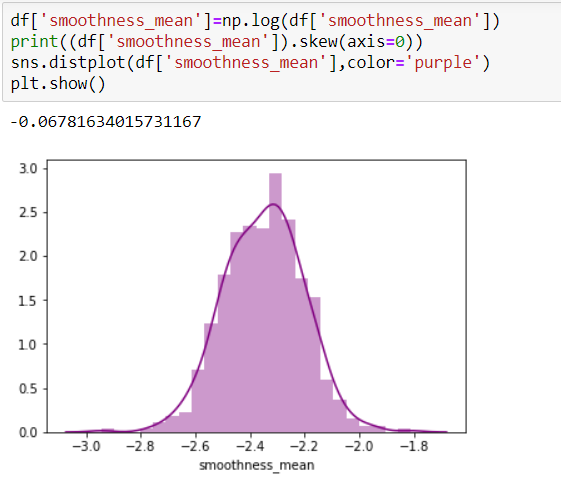
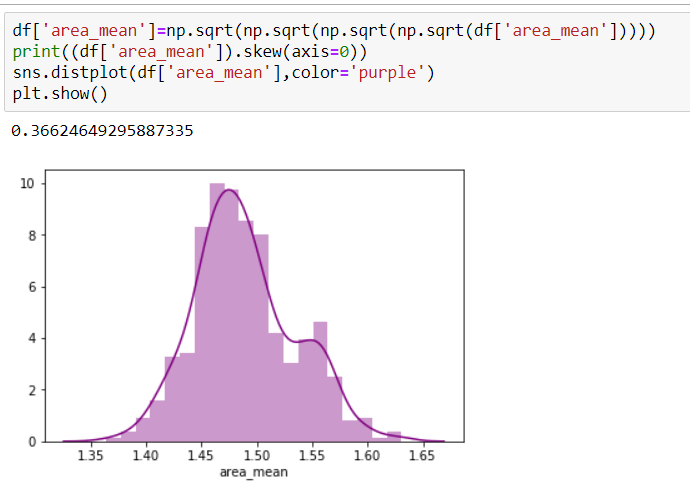
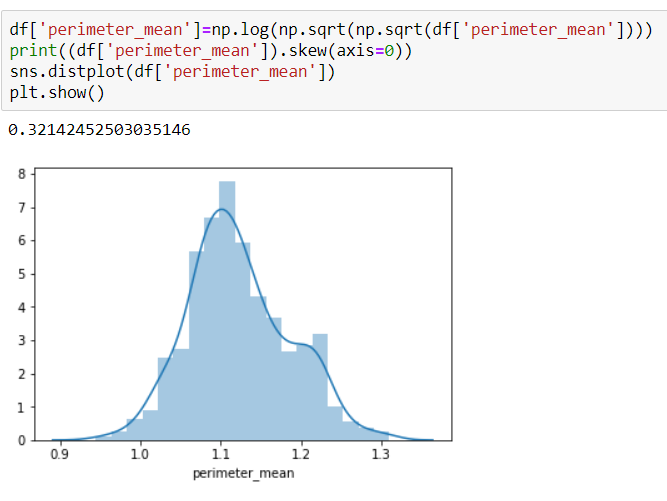
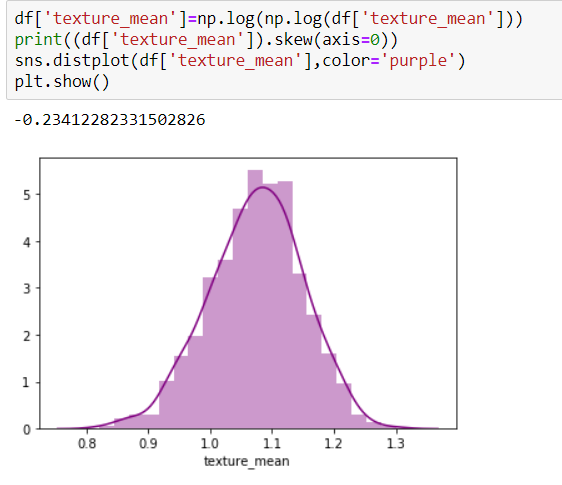
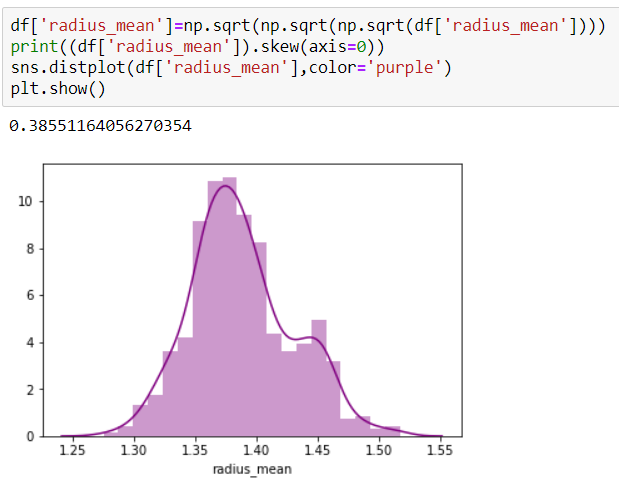
We can see from the distplot, each variable we’ve is left skewed.

So, here we use square root or logarithm. Multiple use of square or logarithm can reduce skewness efficiently, it depends on the variable.  
Here we use both (square and logarithm) for nearly zero value.

df ['Variable Name’] =np.log (df ['Variable Name'])

df ['Variable Name’] =np. sqrt (df ['Variable Name'])

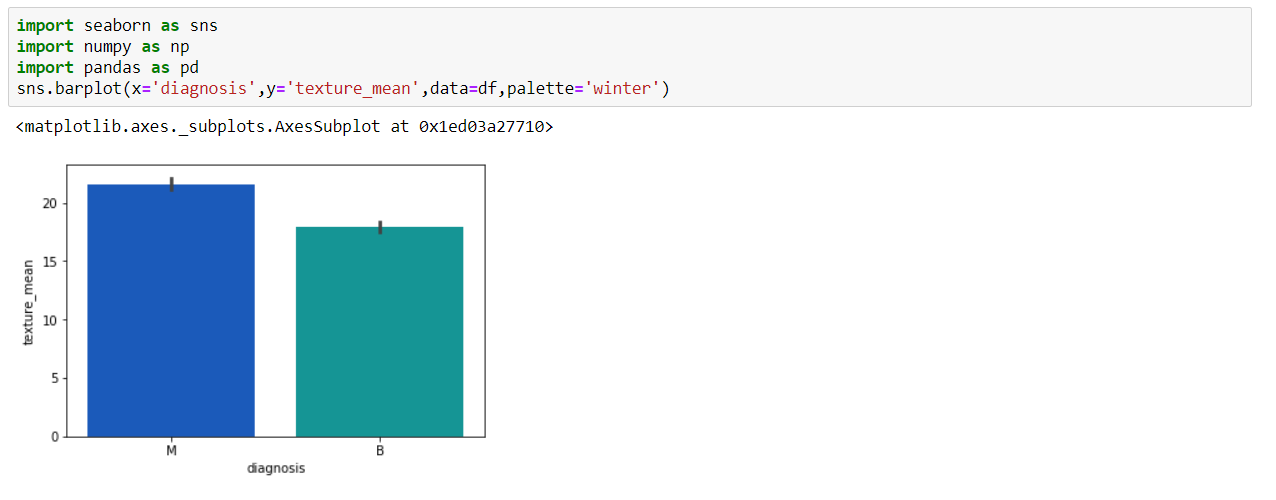
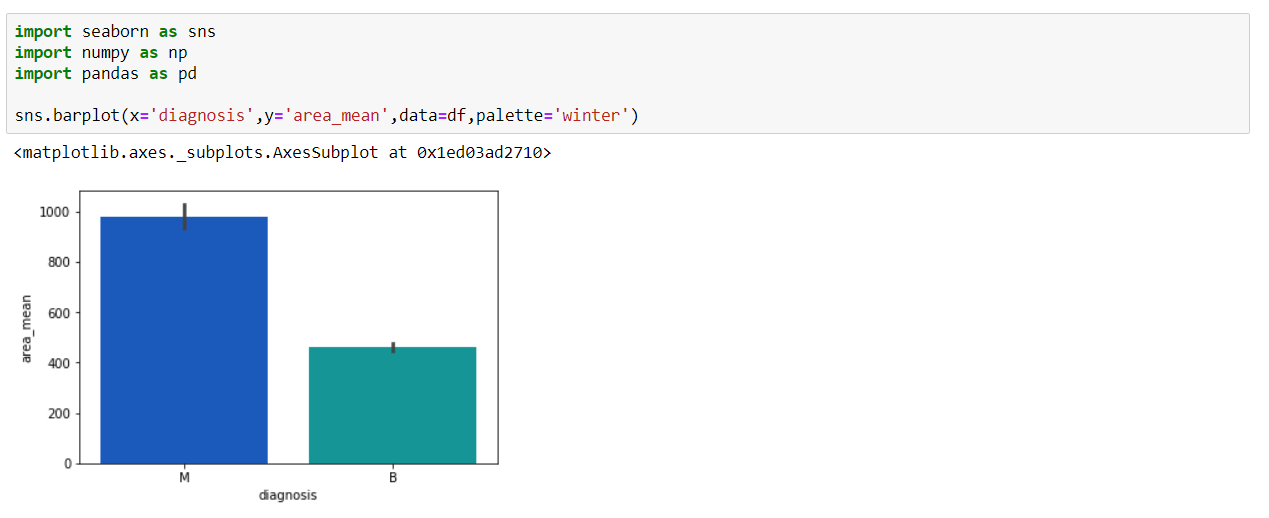
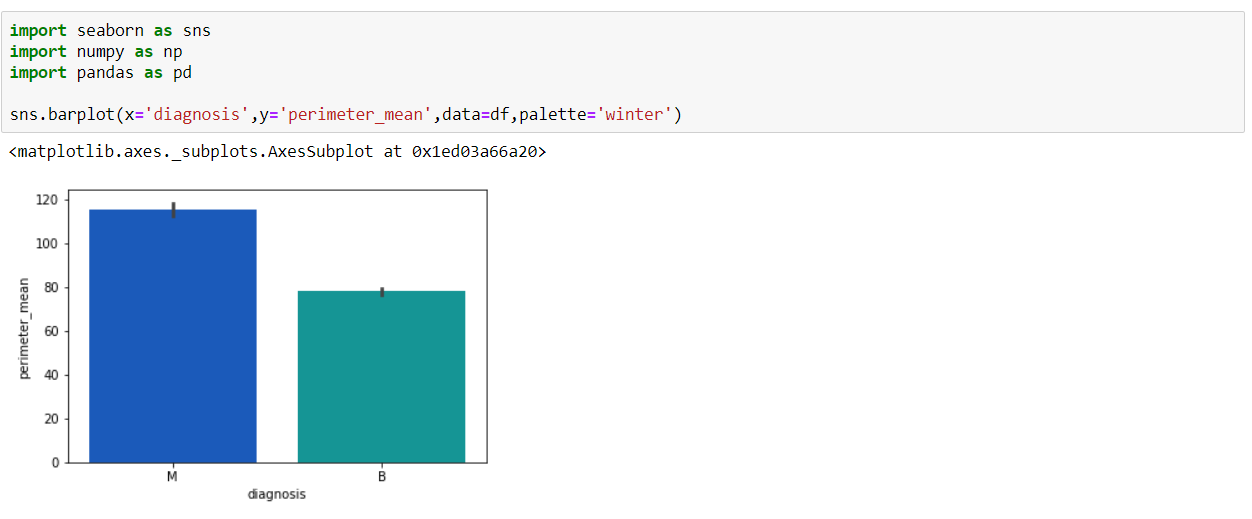
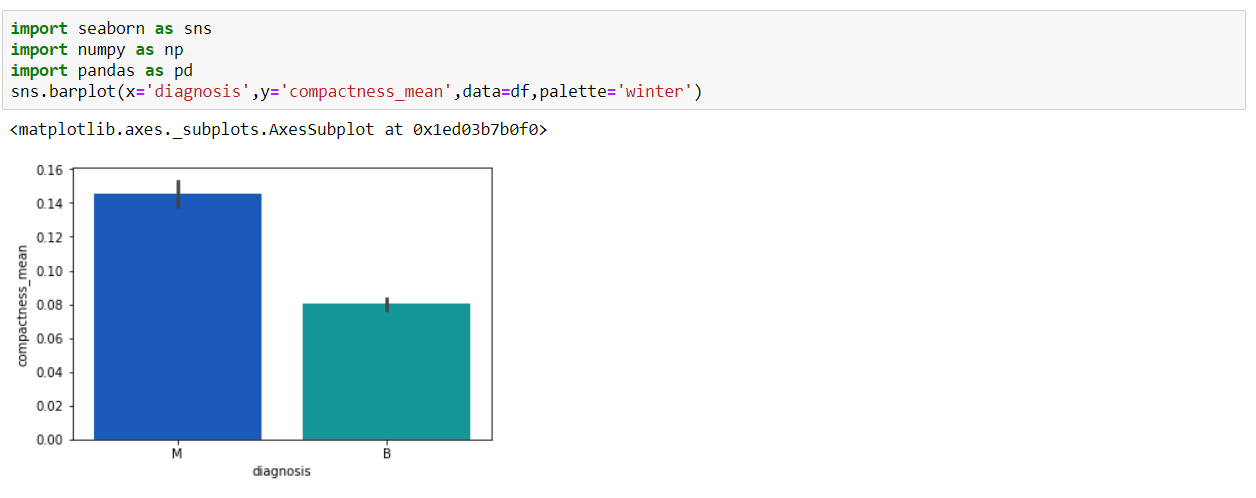
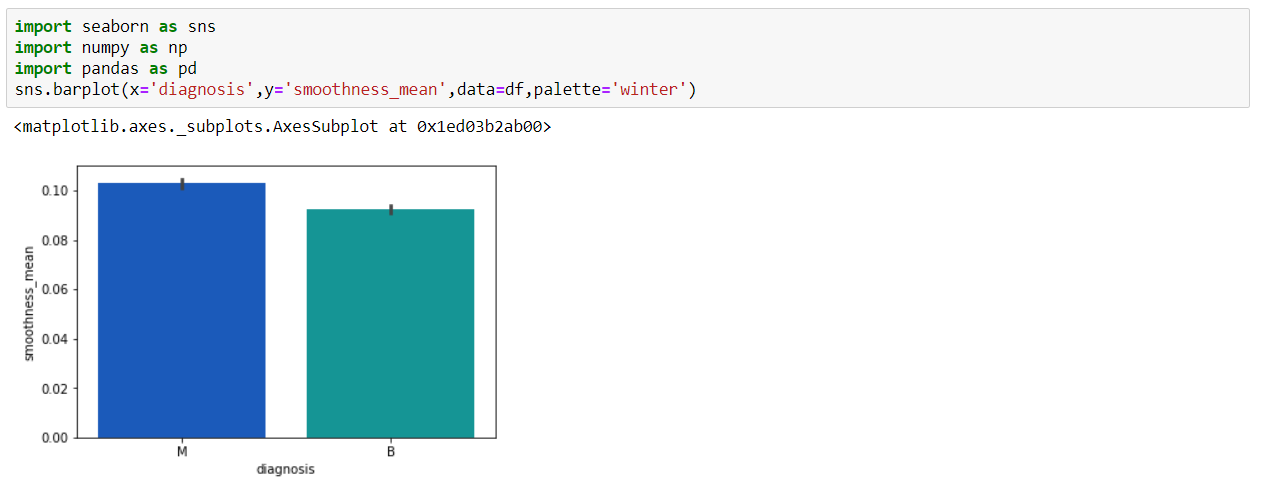
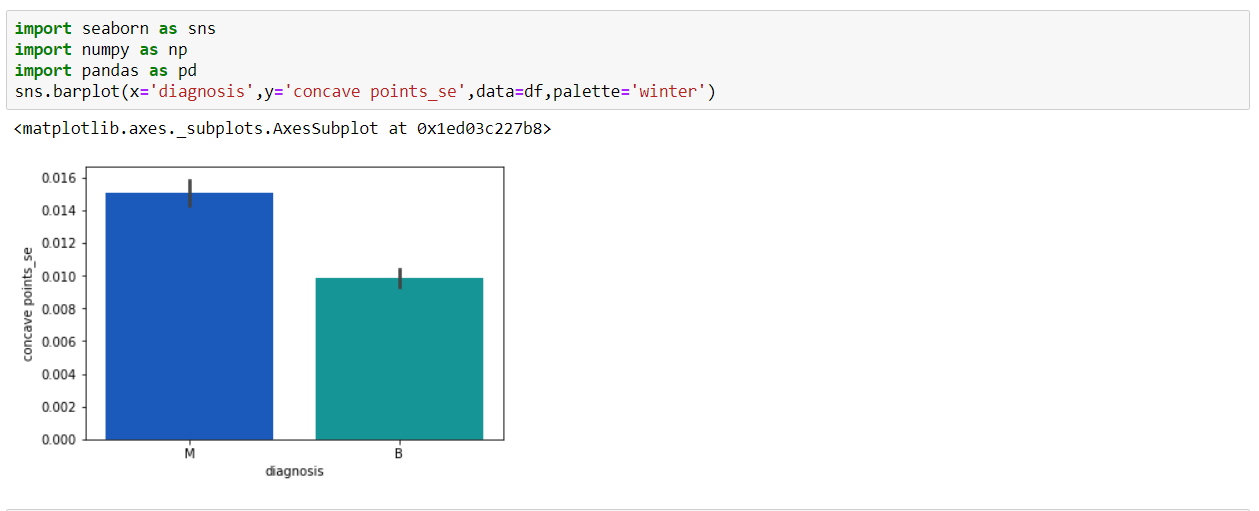
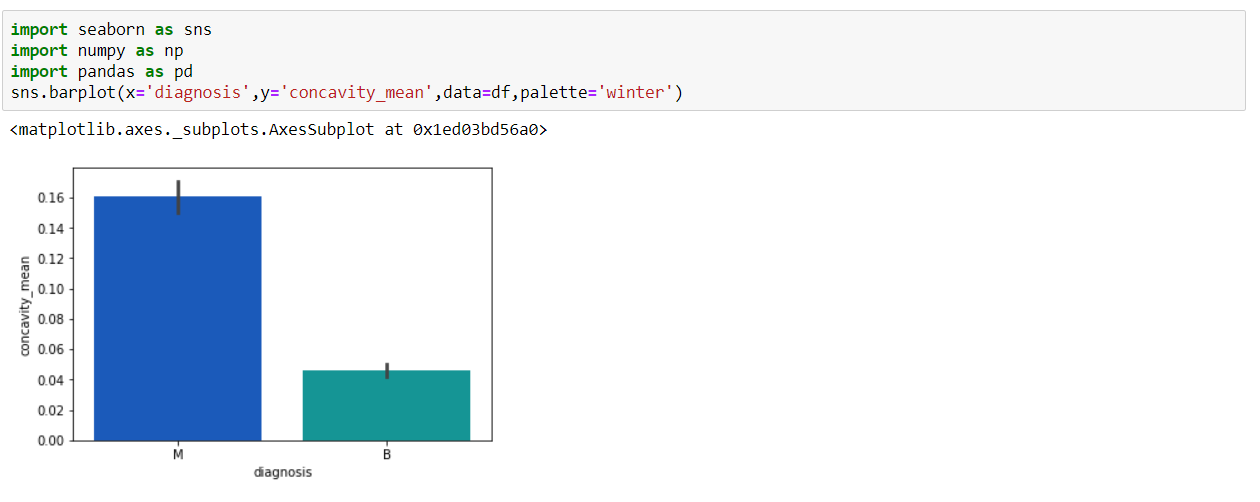
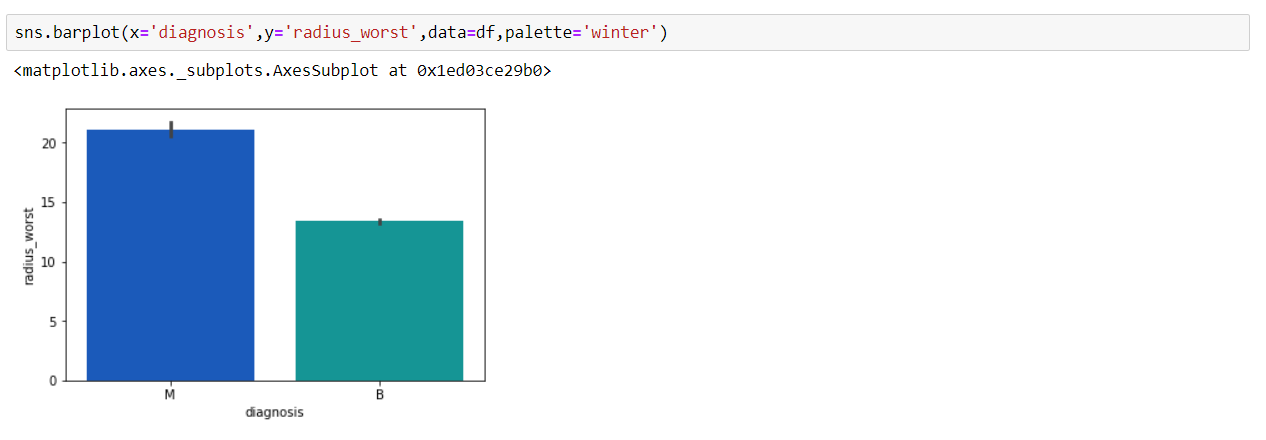
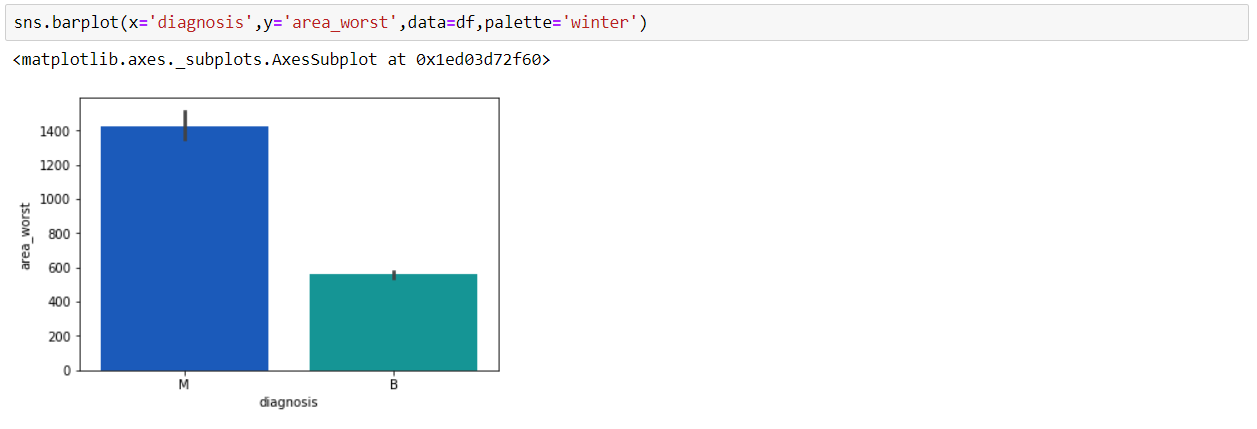
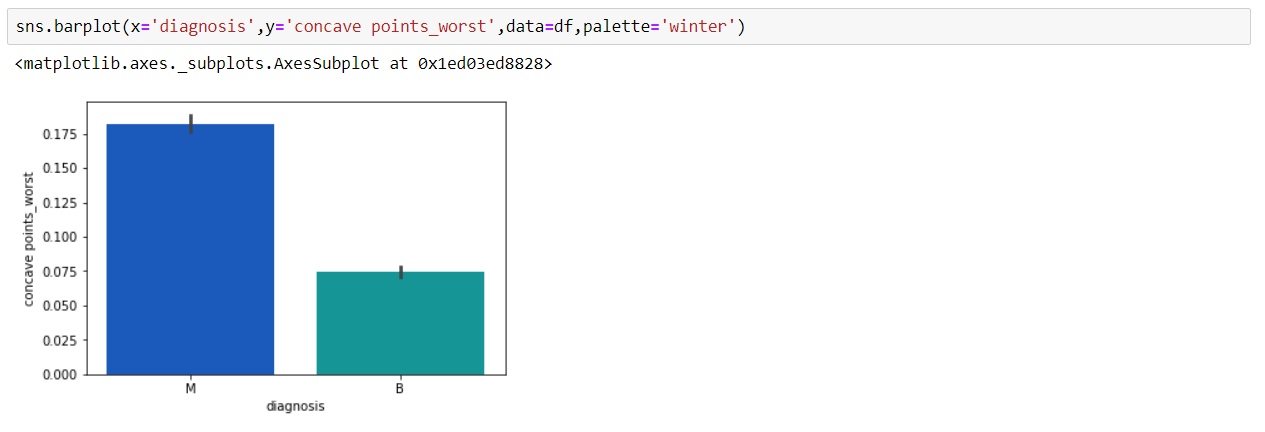




**2) Bivariate Analysis:**

In bivariate analysis we check the relationship between the output feature and the input features through “Bar plot”. At first, we add “seaborn” module as sns.

import seaborn as sns  
sns.barplot(x=’Output Feature’, y=’Input Features’, data=Features)

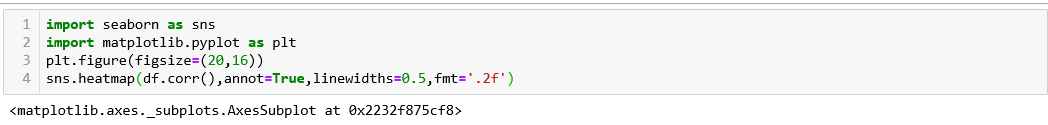
       

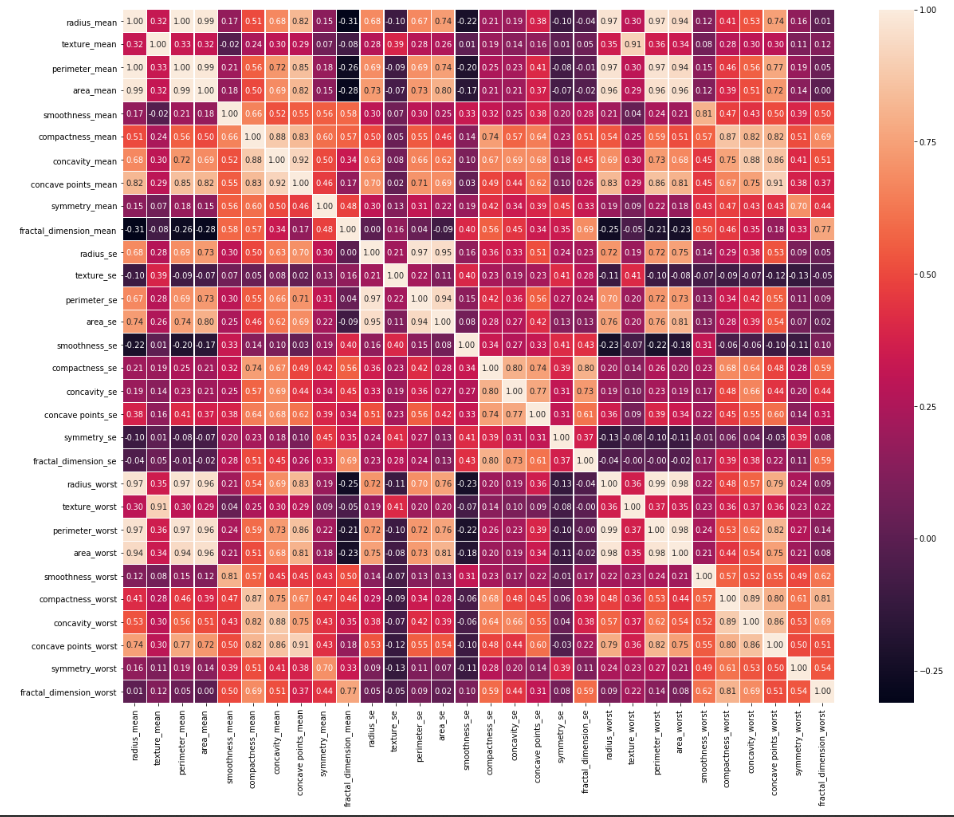


**Building Model:**

**Feature Selection:**

Before building the model we’ve to check **correlation** between two features. If the correlation is high, then we can drop any one of them. Because both will have the same contribution to the output feature. For finding correlation we need to plot ‘Heatmap’ first. The code for plotting heatmap is given below.





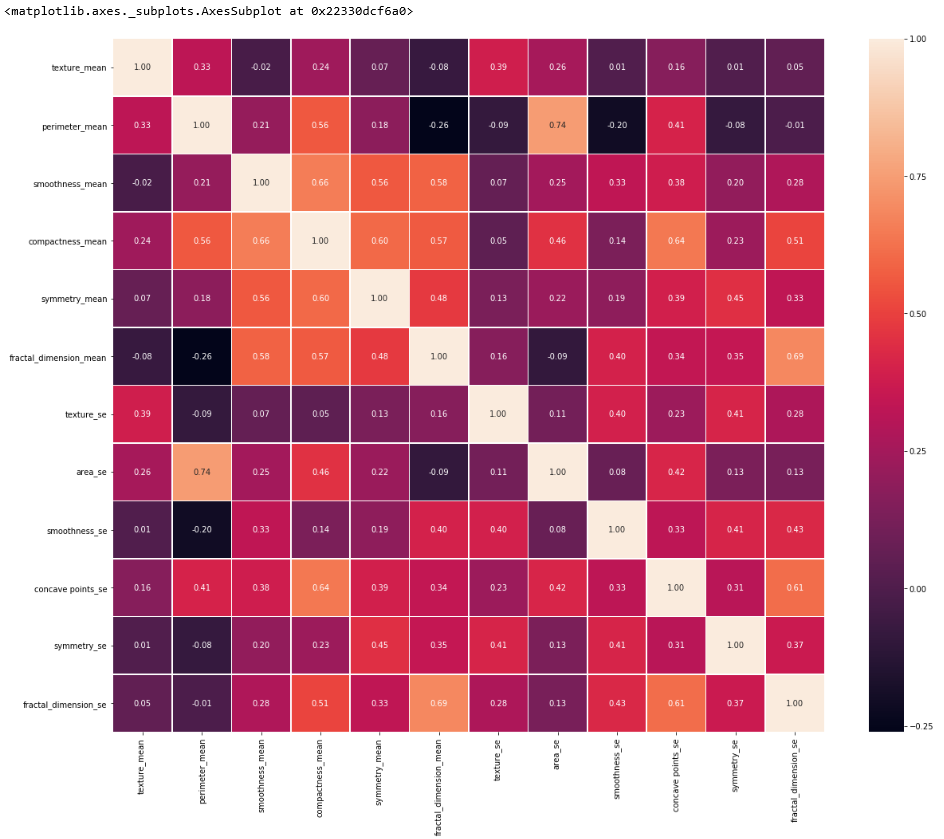
From the heatmap we can determine which columns are to drop. We dropped

From the heatmap we can determine which columns are to drop. We dropped radius\_worst, radius\_se, radius\_mean, texture\_worst, perimeter\_worst, area\_worst, smoothness\_worst, compactness\_worst, concavity\_worst, concave points\_worst, symmetry\_worst, fractal\_dimension\_worst, concavity\_se, perimeter\_se ,concavity\_mean, area\_mean ,concavity\_se, compactness\_se, concave points\_mean as they are strongly correlated with

Diagnosis, texture\_mean, perimeter\_mean, smoothness\_mean, compactness\_mean, symmetry\_mean, fractal\_dimension, \_mean texture\_se, area\_se, smoothness\_se, concave points\_se, symmetry\_se, fractal\_dimension\_se.

A close up of a logo

Description automatically generated  
After dropping columns heatmap will be like this:

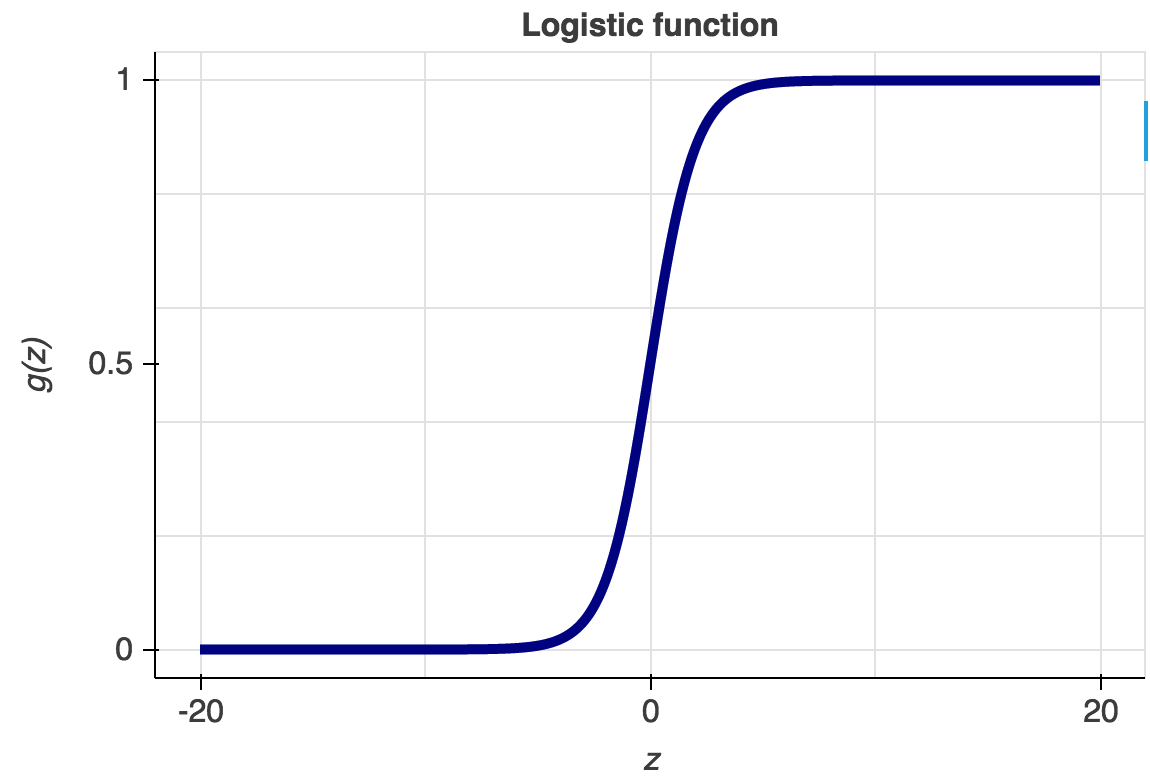


After exploratory data analysis (EDA) now we’ll build our predicting model based on inductive learning. There are two types of task mainly, one is classification and another is regression. Classification type of problem is used for the data with discreet or categorical values and Regression type of problem is used for continuous type of data. As our dataset is filled with continuous values. So, our problem is based on regression(logistic) type.

There are four different models to implement the proposed model.

1. Logistic Regression
2. Decision Tree
3. KNN
4. Naive Bayes’

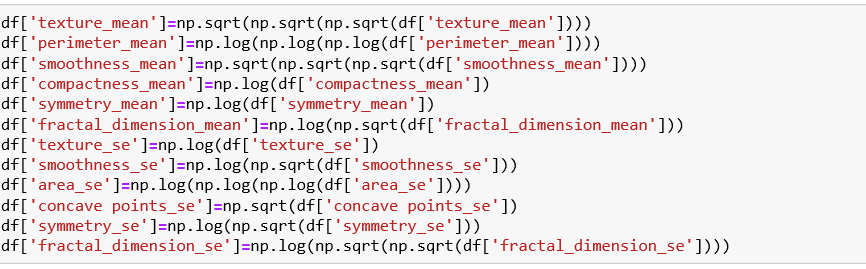
**Logistic Regression:**Logistic regression is used to find the probability of True and False. We should use logistic regression when the dependent variable is binary (0/ 1, True/ False, Yes/ No) in nature. Here the value of Y ranges from 0 to 1 and it can be represented by following graph (‘s’ curve).

****

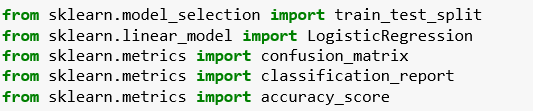
Here we implement the logistic regression.

A close up of a piece of paper

Description automatically generated



(Removing skewness from the data)



(Importing modules)

First, we will train and split data using 9 input features. Then we’ll increase the number of features one by one.

A screenshot of a social media post

Description automatically generated

A close up of text on a white surface

Description automatically generated

A screenshot of a cell phone

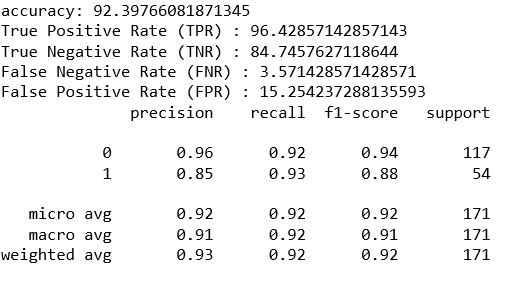
Description automatically generated

A screenshot of text

Description automatically generated

A screenshot of a cell phone

Description automatically generated



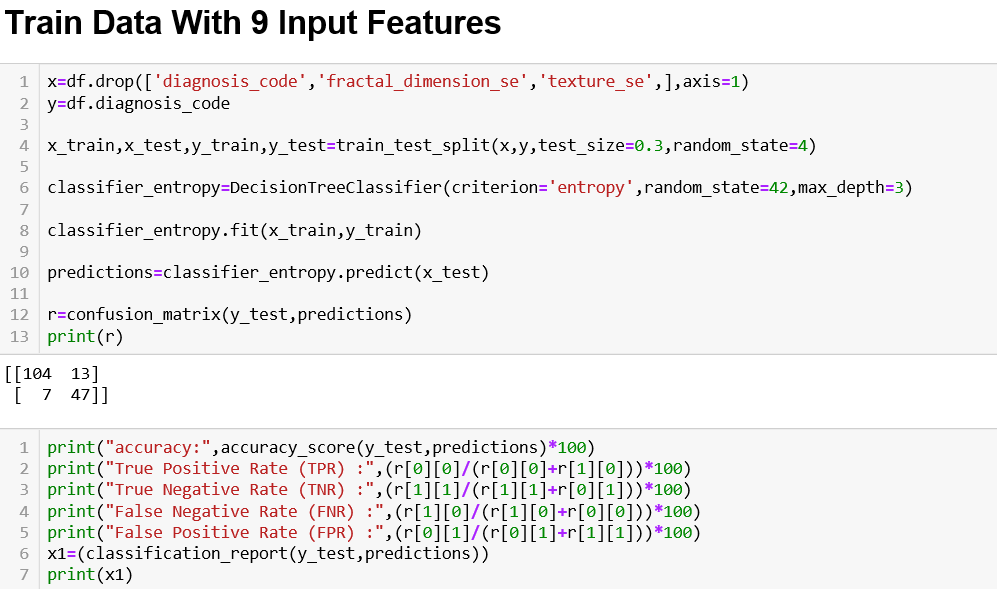
A screenshot of a cell phone

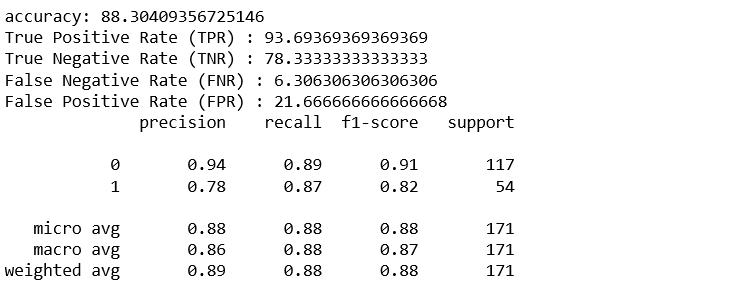
Description automatically generated

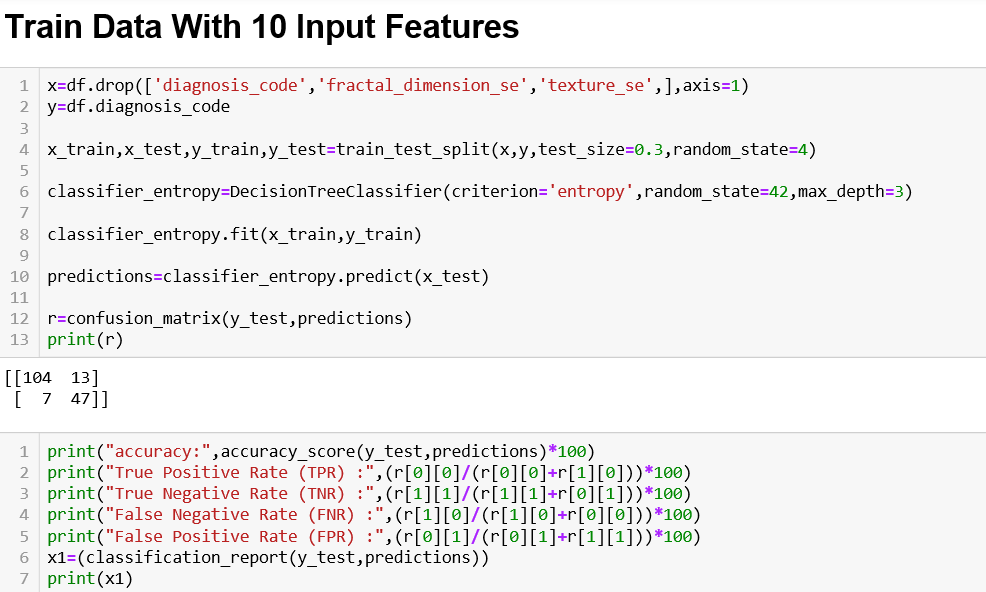
A screenshot of text

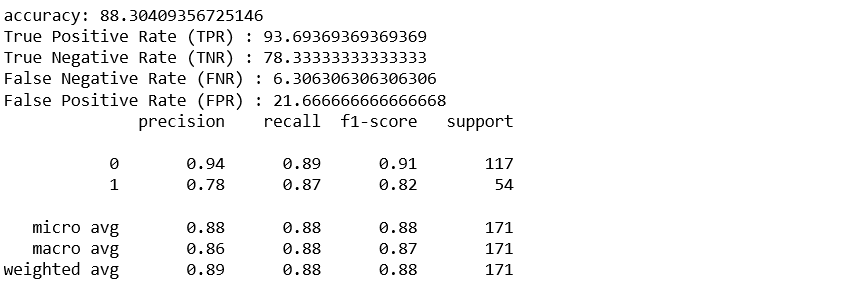
Description automatically generated

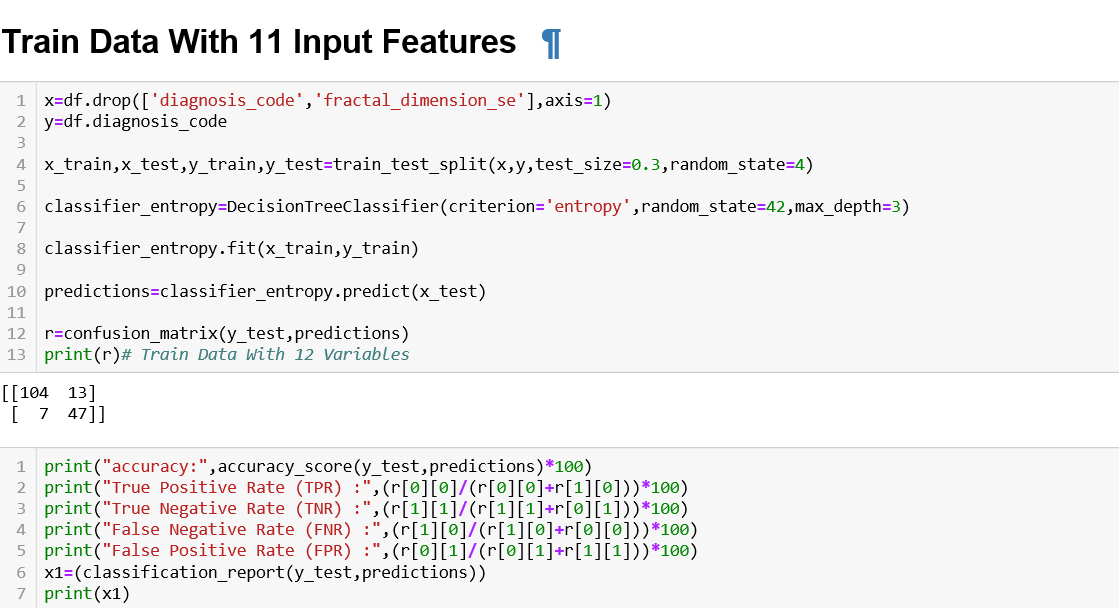
From the above analysis we can see that with 11 input features our predicting model gives highest accuracy which is 92.3976.

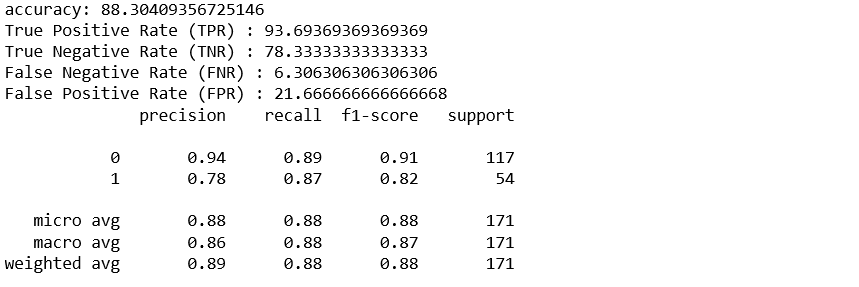
**Decision Tree:**In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making.  
The codes of decision tree for our predicting model is shown below.  
Here also we’ll check the model with 9 input features and then we’ll increase the number of input features.  


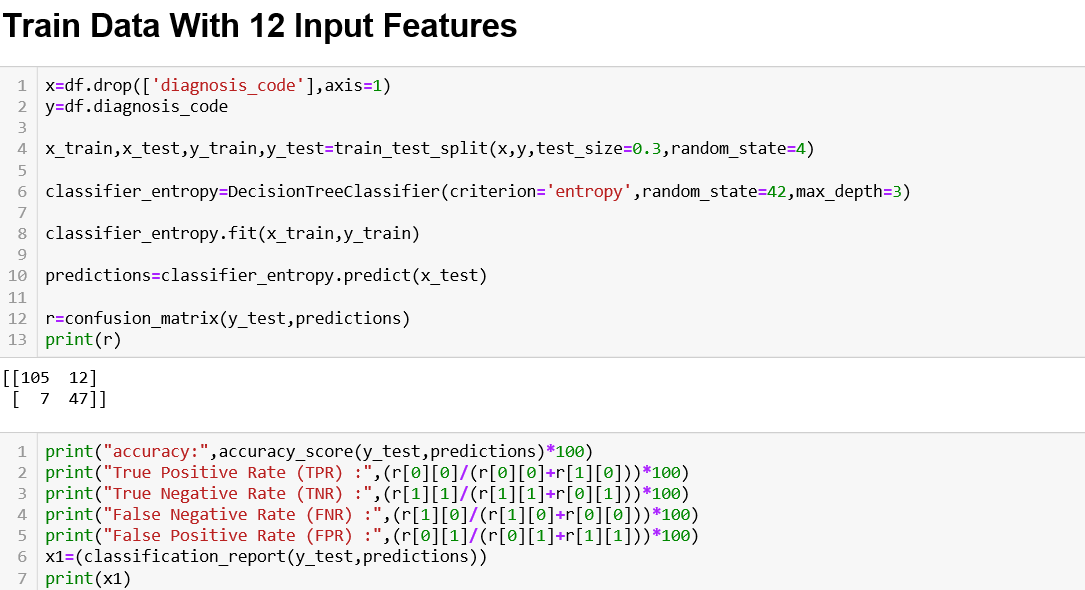


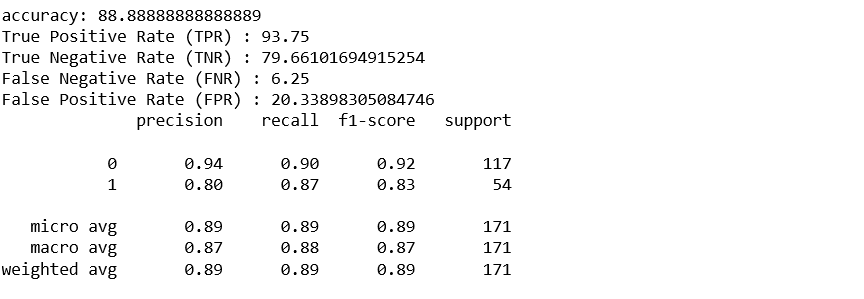






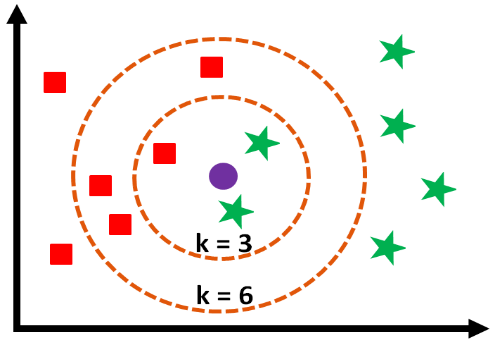






From the above analysis we can visualize that with 11 input features our predicting model gives highest accuracy.

**KNN (k-nearest neighbours) Model:**KNN can be used for both classification and regression predictive problems. However, it is widely used in classification type of problems.



Our predicting model based on KNN is shown below.

A screenshot of a social media post

Description automatically generatedA screenshot of a cell phone

Description automatically generated

**A screenshot of a social media post

Description automatically generatedA screenshot of a cell phone

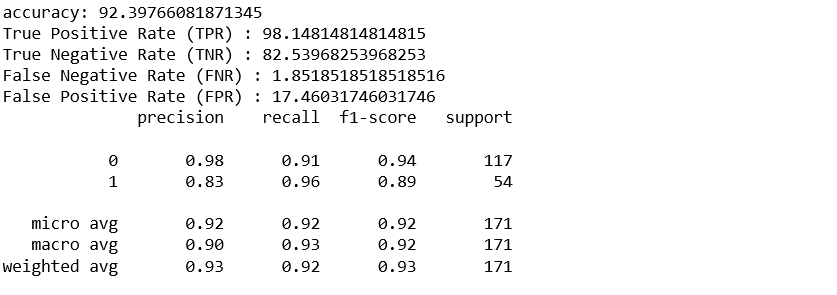
Description automatically generated**

**A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated**

**A screenshot of a cell phone

Description automatically generated**

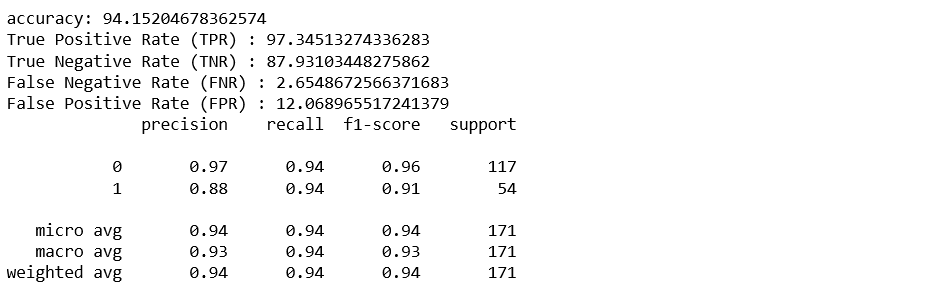
With 11 input features our model gives higher accuracy than others.  
Now we’ll put value of k as 5,7,9 on the model with 11 input features to see for which value of k model is more accurate.

A screenshot of a cell phone

Description automatically generated

A close up of text on a white background

Description automatically generatedA screenshot of a social media post

Description automatically generated

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated

For the value of k=9(n\_neighbours) our model predicts with more accuracy.

**Naive Bayes’:**The Naïve Bayesian classifier is based on Bayes’ theorem.  
In order to do that we need to import GaussianNB from from sklearn.naive\_bayes module.

from sklearn. naive\_bayes import GaussianNB

**A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated**

**A screenshot of a social media post

Description automatically generated**

**A screenshot of a cell phone

Description automatically generatedA screenshot of a social media post

Description automatically generated**

**A screenshot of a cell phone

Description automatically generated**

**A screenshot of a cell phone

Description automatically generated**

**A screenshot of a cell phone

Description automatically generated**

From the above analysis we can see the that with the 11 input features Naïve Bayes predicting model gives the higehst accuracy.  
So, we took the model with 11 input features as the final.

**Ensemble Learning:**It is much reliable to use various models rather than just one. It’s a collection of several models (which are known as ‘stand-alone’ model) working together on a single set is called an Ensemble. This method is called Ensemble Learning.

Ensemble Learning is primarily use to improve the(classification,prediction,function approximation) performance of a model or reduce the likelyhood of an unfortunate selection of a poor one.

Commonly used ensemble learning methods:

1. Bagging
2. Boosting
3. Adaboost
4. Voting

As our values are heterogeneous so we’ve to use voting technique here. In voting technique we need to take all of our predicting model which is mentioned previously.

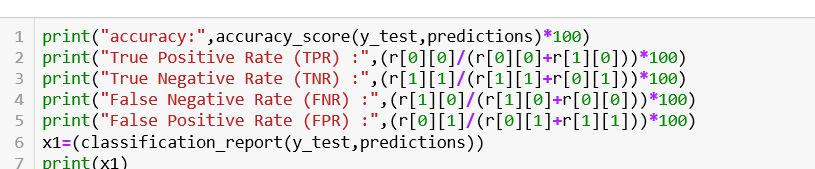
**Voting Technique:**

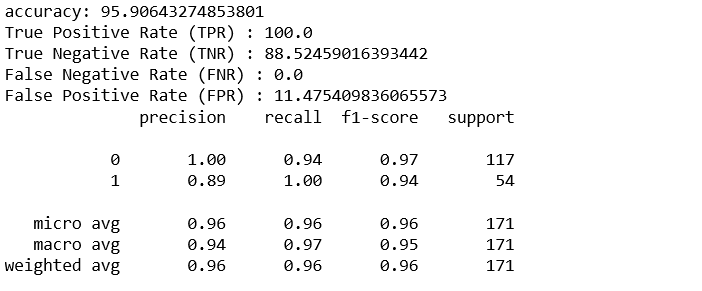
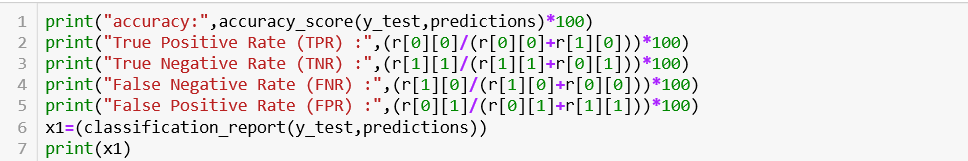
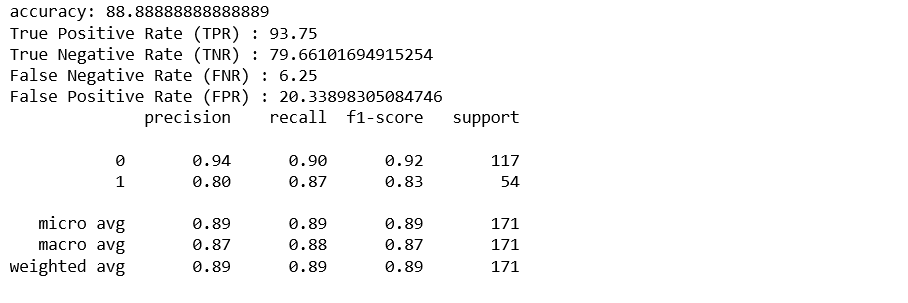
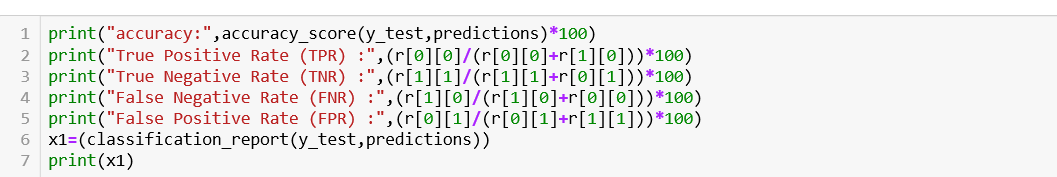
Voting is one of the simplest way of combining the predictions from multilple machine learning algorithms, it works by first creating some standalone model from trainig dataset.

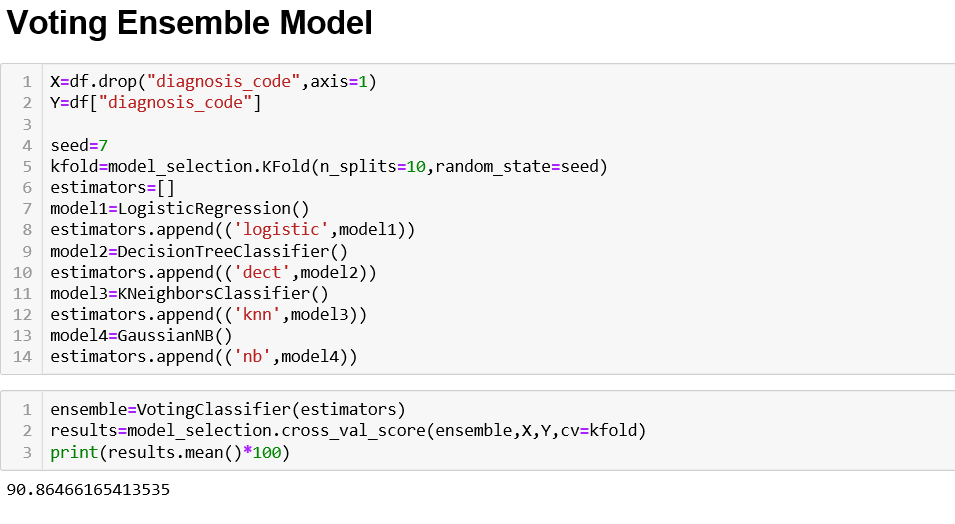
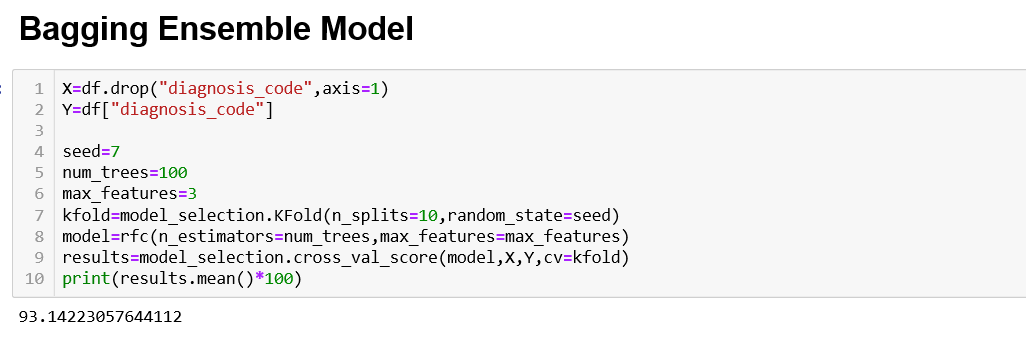
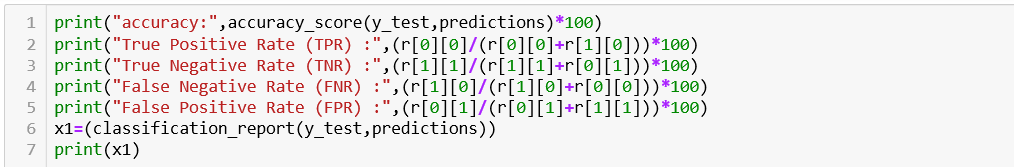
For voting we need to first import **VotingClassifier** from **sklearn.ensemble** and import **classification\_report** from **sklearn.metrics**

At first we implement the ensemble model only by importing standalone models. 



Now we implement the voting by using the codes for all four standalone models.  
A screenshot of text

Description automatically generatedA screenshot of a social media post

Description automatically generated  
From the above two voting technique we can see that if we import codes for all the satndalone models then the accuracy is lower(90.8646616). But if we only call the models then the accuracy is higher than the previous(91.255388)  
So, here we consider the first one as our final voting model.