Project Increment -1

Title: Breast Cancer Prediction Using Machine Learning Algorithms

Team Members:

- Dinesh Bhogadi
- Veena Ravuri
- Kota Sai Teja Jana

Please note that we have submitted Introduction, Goals and Objectives, Motivation, significance, Objectives, and features are submitted in previous submission. In order to remove the plagiarism, we are not adding the same here.

1. Related work (Background):

Machine learning techniques have seen tremendous progress, which has sparked a lot of interest in using them to solve medical imaging issues. Here, we develop a machine learning algorithm that can correctly identify breast cancer on screening mammograms data set using an "end-to-end" training technique that effectively utilizes training datasets with either full clinical annotation or only the cancer status of the data available.

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

In the study, patient classification using machine learning techniques is done using data from microarrays of breast cancer. In the first example, the dataset is subjected to the application of eight different machine learning algorithms, and the classification outcomes are recorded. Then, in the second instance, the microarray breast cancer dataset was subjected to two distinct feature selection methods, such as Recursive Feature Elimination (RFE) and Randomized Logistic Regression (RLR), and 50 features were selected as the stop criterion.

2. Dataset

In this project, we are using Breast Cancer Wisconsin (Diagnostic) Data Set. This is an open-source dataset can be downloaded from Kaggle. Coming to the dataset, the data present in the csv file is from a digital image of a fine needle aspirate (FNA) of a breast mass, features are calculated. They characterize the traits of the visible cell nuclei in the picture. a classification technique that builds several machine learning algorithms using linear programming. An exhaustive search in the domain of 1-4 features and 1-3 separation planes was used to select relevant features. Information on 569 women across 32 different qualities can be found in the dataset. The first group of variables is Mean (3–13), followed by Stranded Error (1–23) and Worst (23–32), each of

which has ten parameters. Mean, Standard Error, and Worst all refer to the average of all the cells, respectively. Every instance contains a parameter of cancerous and non-cancerous cells, and we can forecast cancer simply by inputting features. The feature values are presented in a numeric format. The term "Target" refers to the patient who is suffering from either "Benign" or "Malignant" cancer. Malignant denotes the presence of cancer, while benign denotes the absence of cancer. Features in the dataset have a wide range of units and magnitudes. Therefore, it is necessary to equalize the magnitude of all aspects.

3. Detail design of Features

In this data set we have 569 rows and 32 columns. Below are some of the features that will be used for the working model. There are total of benign 357 and malignant 212 values.

There are some real value features that can be calculated from each cell nuclei

- Radius which is the mean of distances from center to points on the perimeter
- Texture which is the standard deviation of gray-scale values.
- Perimeter which is the perimeter of the cell
- Area is the total area of the contour
- Smoothness which is the local variation in radius lengths
- Compactness can be calculated using ((perimeter)^2/area -1)
- Concavity is the severity of concave portions of the contour
- Concave points are the number of concave portions of the contour
- Symmetry is defined as the symmetry as the contour
- Fractal dimension: A fractal dimension is a ratio that compares how a pattern's level of detail alters depending on the size at which it is measured. This can be calculated by coastline approximation – 1.

4. Analysis

A) Data Collection:

(I) Importing the dataset

We have imported the dataset with the help of pandas. The code is shown below for importing the dataset using pandas.

```
import numpy as np
import pandas as pd
cancer_data = pd.read_csv("BreastCancer.csv")
```

After successfully importing the dataset we will check for the shape of the dataset which can be done using the Cancer_data.shape. To print the first 5 and last five cells of the data frame we are using the commands cancer_data.head() and cancer_data.tail().

Displaying the first five rows of data frame.

```
cancer_data.head() #printing the first five cells
```

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	...	texture_worst	perimeter_worst	area_worst	smoothness_worst	compactnes
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	...	17.33	184.60	2019.0	0.1622	
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	...	23.41	158.80	1956.0	0.1238	
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	...	25.53	152.50	1709.0	0.1444	
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	...	26.50	98.87	567.7	0.2098	
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	...	16.67	152.20	1575.0	0.1374	

5 rows × 33 columns

Displaying the last five rows of dataset

```
cancer_data.tail() #printing the last 5 cells
```

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	...	texture_worst	perimeter_worst	area_worst	smoothness_worst	compactnes
564	926424	M	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	...	26.40	166.10	2027.0	0.14100	
565	926682	M	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	...	38.25	155.00	1731.0	0.11660	
566	926954	M	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	...	34.12	126.70	1124.0	0.11390	
567	927241	M	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	...	39.42	184.60	1821.0	0.16500	
568	92751	B	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	...	30.37	59.16	268.6	0.08996	

5 rows × 33 columns

Once the dataset is loaded into the data frame we can proceed with the EDA of the dataset.

B) EXPLORATORY DATA ANALYSIS:

I) Basic information about the data:

For any dataset we need to know about the basic structure of the data before working on the dataset. The cancer_data.info() will give us the required information about the dataset.

```
cancer_data.info() #getting the info of the dataset
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 33 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
 32  Unnamed: 32              0 non-null      float64
dtypes: float64(31), int64(1), object(1)
memory usage: 146.8+ KB
```

The cancer_data.describe() will give us the statistic view of the dataset with all the information like mean, count, std.

```
cancer_data.describe()
```

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	...	radius_worst	texture_worst	perimeter_worst	area_worst	smoo
count	5.690000e+02	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	...	569.000000	569.000000	569.000000	569.000000	
mean	3.037183e+07	0.372583	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799	0.048919	...	16.269190	25.677223	107.261213	880.583128	
std	1.250206e+08	0.483918	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720	0.038803	...	4.833242	6.146258	33.602542	569.356993	
min	8.670000e+03	0.000000	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	0.000000	...	7.930000	12.020000	50.410000	185.200000	
25%	8.692180e+05	0.000000	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560	0.020310	...	13.010000	21.080000	84.110000	515.300000	
50%	9.060240e+05	0.000000	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540	0.033500	...	14.970000	25.410000	97.660000	686.500000	
75%	8.813129e+06	1.000000	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700	0.074000	...	18.790000	29.720000	125.400000	1084.000000	
max	9.113205e+08	1.000000	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800	0.201200	...	36.040000	49.540000	251.200000	4254.000000	

8 rows × 32 columns

II) Finding the Null values:

This is an important step during the EDA of data. Ensuring the quality of data is important. Lets see how to find the null values.

```
cancer_data.isna().sum()  #checking the sum of null values
```

```
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
Unnamed: 32                569
dtype: int64
```

From the above image we can see that a column named unnamed : 32 has 569 null values so we need to drop the entire column. This can be done using the data.dropna.

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

III) Checking for duplicated values:

The df.duplicate can be used. The sum of any duplicate values is calculated using the ancer_data.diagnosis.unique() function. If duplicate values are present in the data, it will display how many of them are there.

```
diagnosis_unique_data = cancer_data.diagnosis.unique() #checking for unique values
```

IV) Knowing the data types:

It is essential to know about the data types of each and every column of a dataset. This can be done using cancer_data.dtypes

```
cancer_data.dtypes #checking data type of each column
```

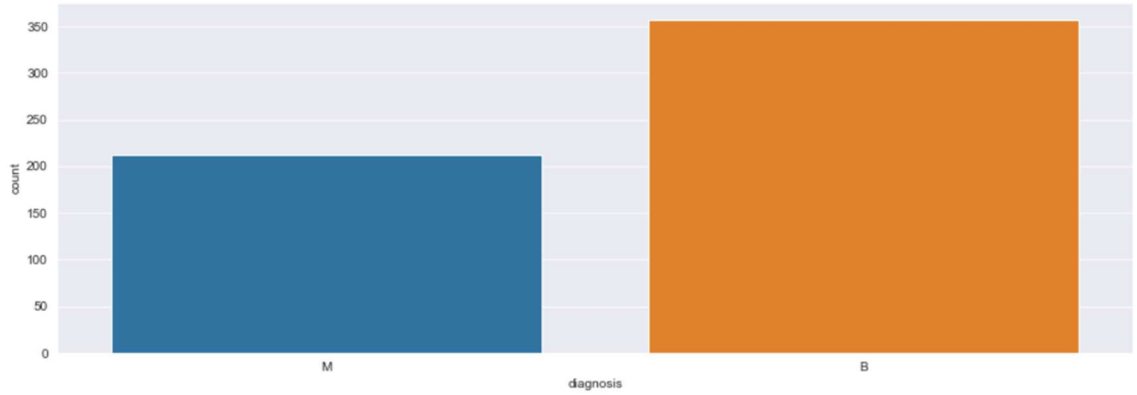
```
id                          int64
diagnosis                  object
radius_mean               float64
texture_mean              float64
perimeter_mean            float64
area_mean                 float64
smoothness_mean           float64
compactness_mean          float64
concavity_mean            float64
concave points_mean       float64
symmetry_mean             float64
fractal_dimension_mean    float64
radius_se                 float64
texture_se                float64
perimeter_se              float64
area_se                   float64
smoothness_se             float64
compactness_se            float64
concavity_se              float64
concave points_se         float64
symmetry_se               float64
fractal_dimension_se      float64
radius_worst              float64
texture_worst             float64
perimeter_worst           float64
area_worst                float64
smoothness_worst          float64
compactness_worst         float64
concavity_worst           float64
concave points_worst      float64
symmetry_worst            float64
fractal_dimension_worst   float64
dtype: object
```

C) Data visualization:

I) Bar graph:

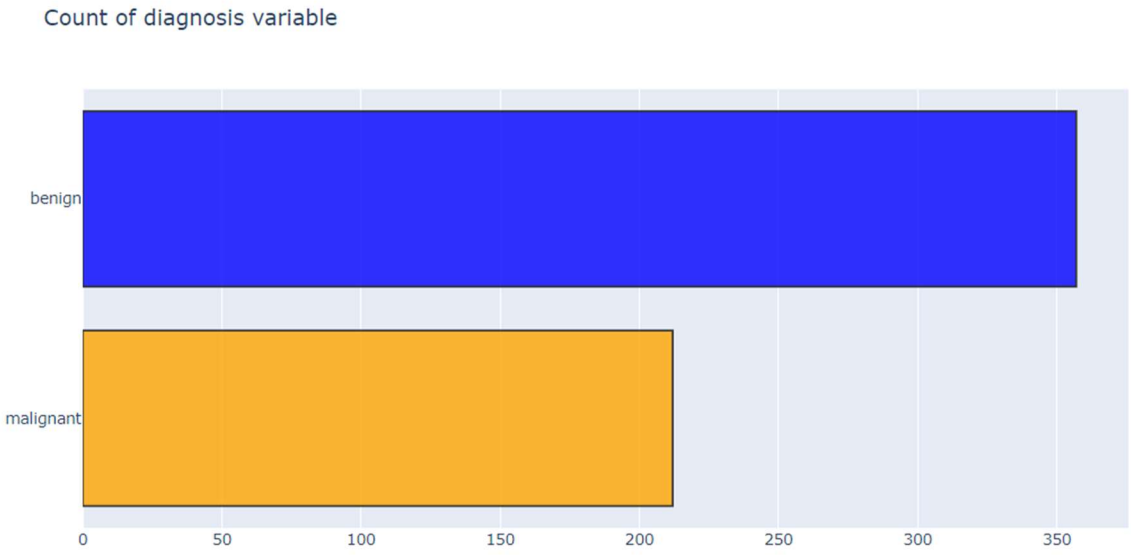
The bar plots are rectangular, vertical/horizontal graphs that compare data and allow you to see how things have changed over time as shown by another axis. Each bar can hold the value of a single value, or several values divided into ratios. A bar's worth increases with increasing length.

```
plt.figure(figsize=(15, 5)) #size of the plot
sns.countplot('diagnosis', data=cancer_data);
```



Histogram

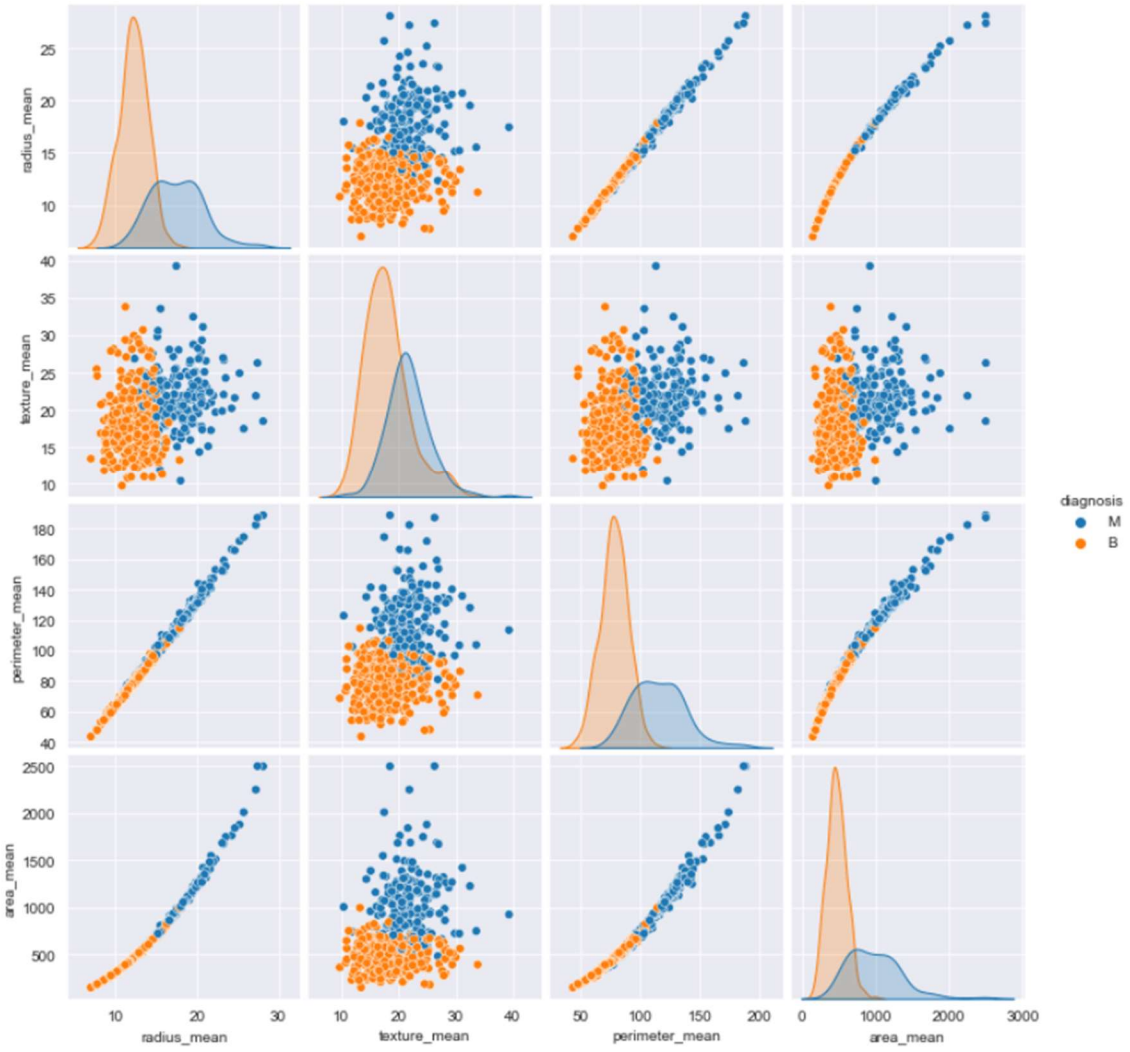
II) Bar graph Using plotly



II) Pair Plot:

In order to discover the link between the given data—where the variables may be continuous or categorical—a pair plot visualizes the data. Pairwise relationships in a data collection should be plotted. A high-level interface for creating eye-catching and instructive statistical visuals is provided by the Seaborn library module Pair plot.

```
columns = ["diagnosis", "radius_mean", "texture_mean", "perimeter_mean", "area_mean"]
sns.pairplot(cancer_data[columns], hue="diagnosis") #plotting pairplot
plt.show()
```

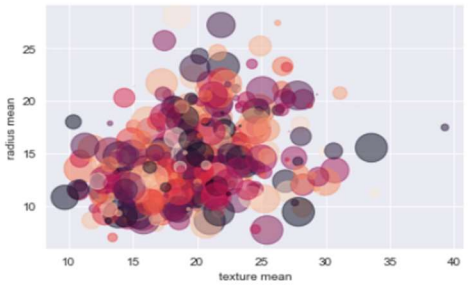


III) Scatter plot

When analyzing multiple data variables to ascertain the relationship between dependent and independent variables, we can use plots. The data is represented as a set of points that have been logically clustered together. Here, a single variable (x) determines how each number relates to the others (Y).

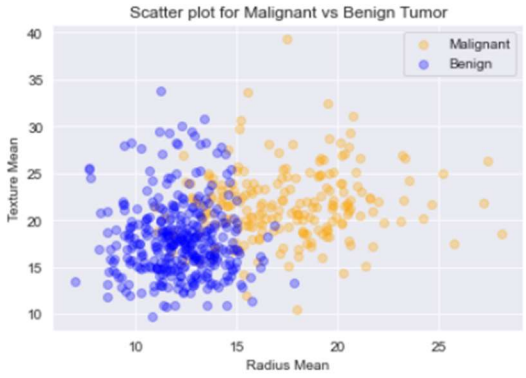
Scatter Plot

```
radius = len(cancer_data['texture_mean'])

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

plt.xlabel("texture mean")
plt.ylabel("radius mean")
plt.scatter(cancer_data['texture_mean'], cancer_data['radius_mean'], s=area, c=colors, alpha=0.5); #plotting a scatter plot
```



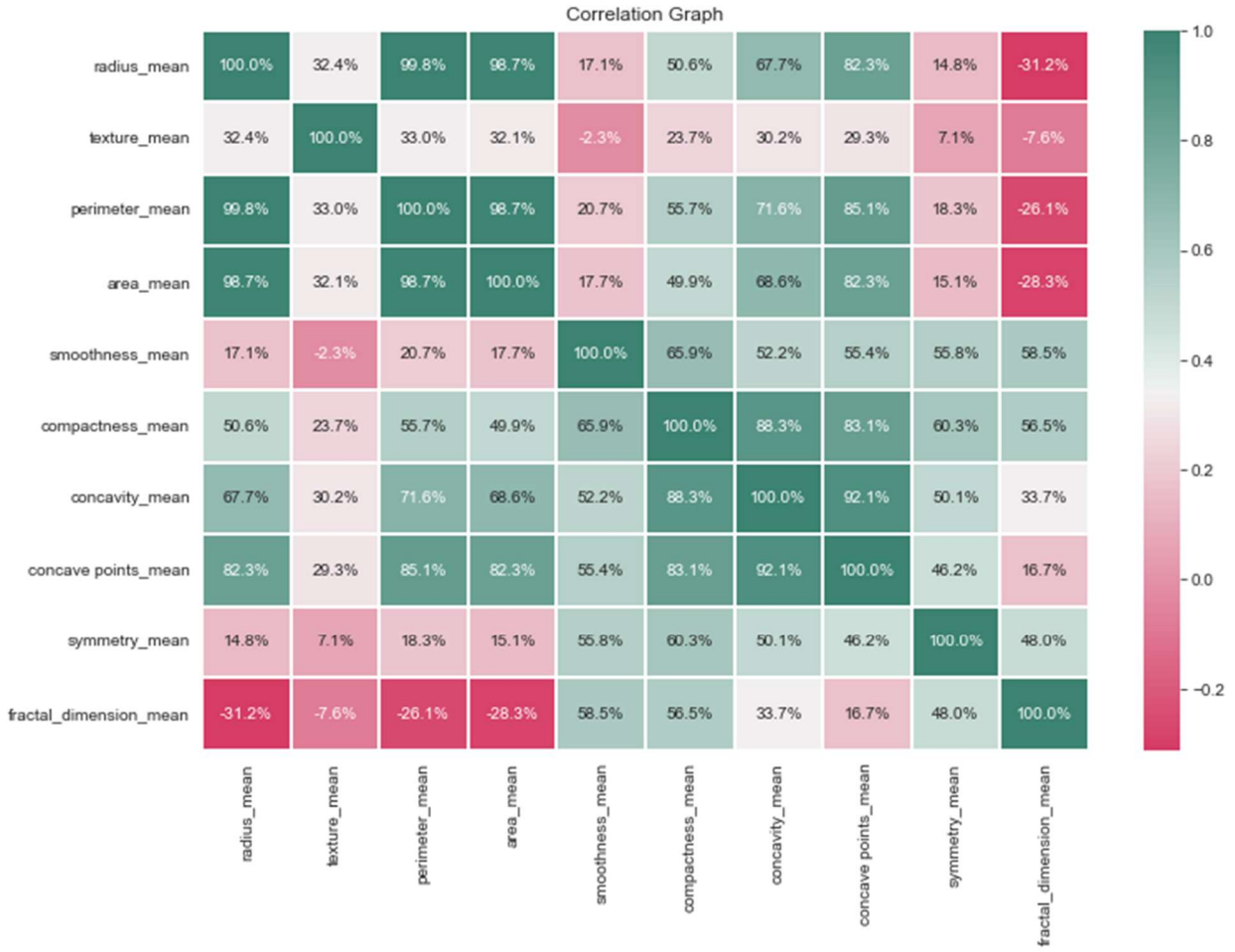
ScatterPlot

```
Malignant = cancer_data[cancer_data.diagnosis == "M"]
Benign_Tumor = cancer_data[cancer_data.diagnosis == "B"]

plt.title("Scatter plot for Malignant vs Benign Tumor")
plt.xlabel("Radius Mean")
plt.ylabel("Texture Mean")
plt.scatter(Malignant.radius_mean, Malignant.texture_mean, color = "Orange", label = "Malignant", alpha = 0.3)
plt.scatter(Benign_Tumor.radius_mean, Benign_Tumor.texture_mean, color = "blue", label = "Benign", alpha = 0.3)
plt.legend()
plt.show()
```

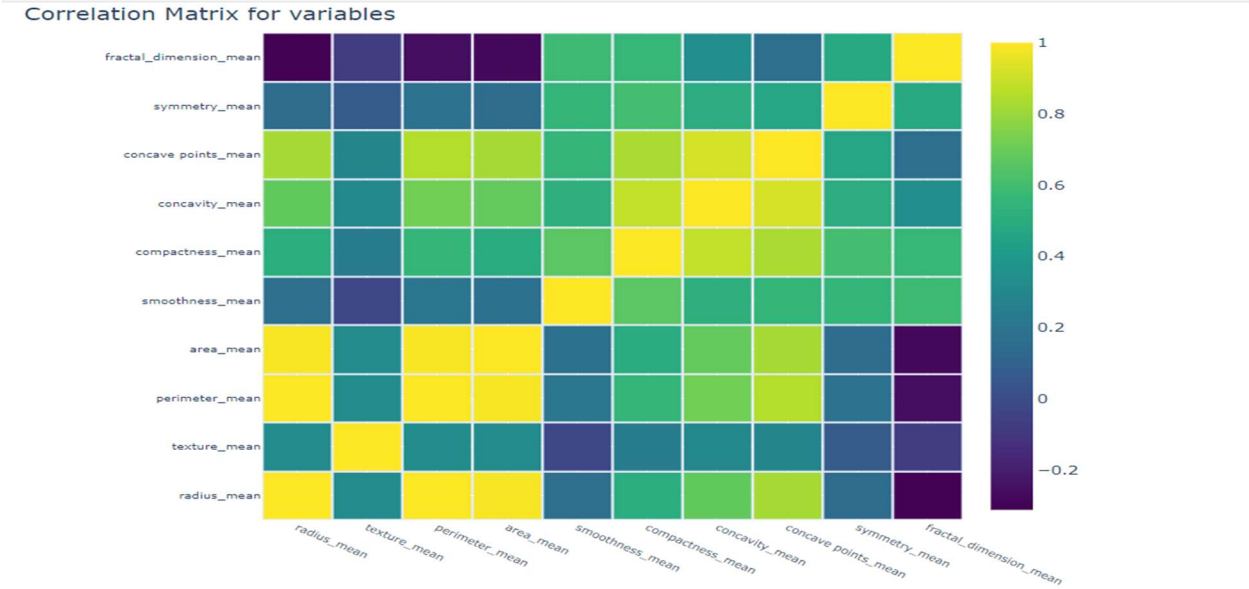


IV) Heatmap using matplot:

A table of the correlation coefficients between different variables is called a correlation matrix. The correlation between any two variables is displayed in each table cell. Data are summarized, input into more complex studies, and diagnostics for complex analyses are all done using correlation matrices.



V) Heat map using plotly

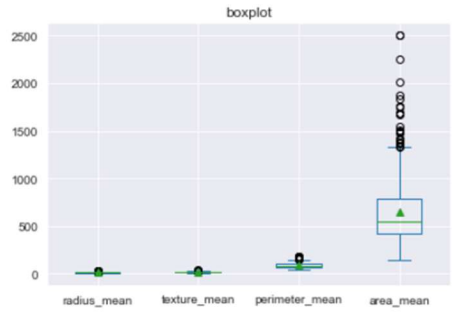


VI) Box Plot

Box plots are basically used to summarize the data values with characteristics like minimum, first quartile, median, third quartile, and maximum. The first quartile to the third quartile are taken as the boundaries of the box plot.

BoxPlot

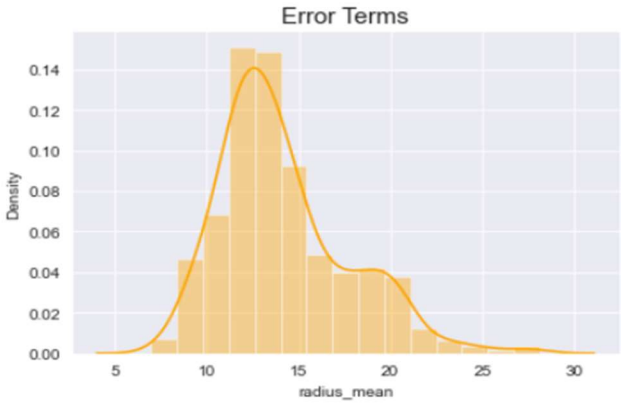
```
col1 = cancer_data['diagnosis']
col2 = cancer_data['radius_mean']
col3 = cancer_data['texture_mean']
col4 = cancer_data['perimeter_mean']
col5 = cancer_data['area_mean']

DF = pd.DataFrame({'diagnosis': col1, 'radius_mean': col2,'texture_mean': col3, 'perimeter_mean': col4, 'area_mean': col5})
#plotting Box plot
ax = DF[['diagnosis', 'radius_mean','texture_mean', 'perimeter_mean','area_mean']].plot(kind='box', title='boxplot', showmeans=True)
plt.show()
```

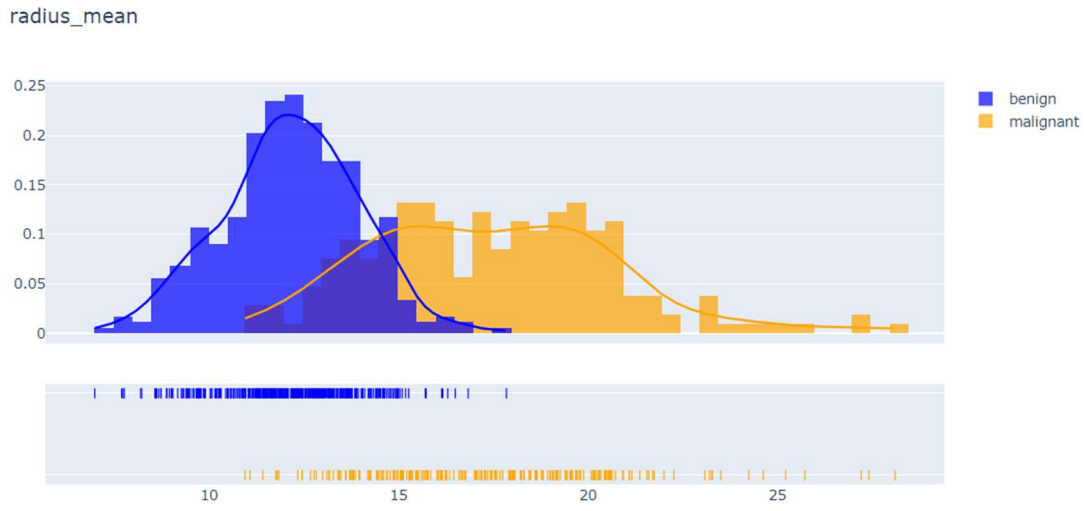


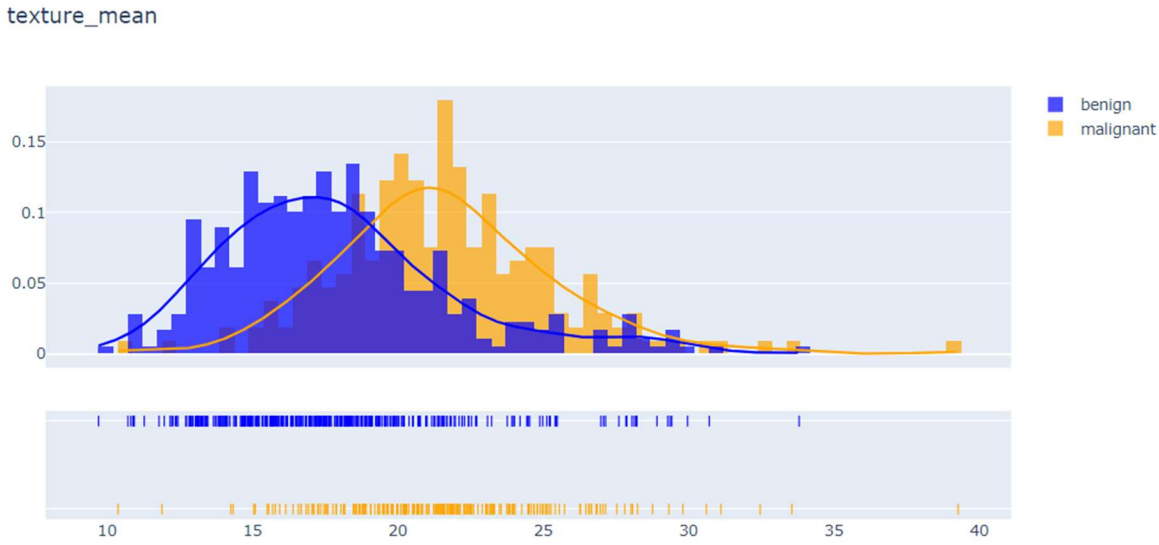
VII) Distplot

A distribution plot, often known as a distplot, shows how the data distribution varies. The total distribution of continuous data variables is depicted by a Seaborn Distplot. The distplot with several modifications is shown using the Seaborn and Matplotlib modules.

```
fig = plt.figure()
sns.distplot(cancer_data['radius_mean'], bins = 15, color='Orange') #plotting displot
plt.title('Error Terms', fontsize = 15)
plt.show()
```



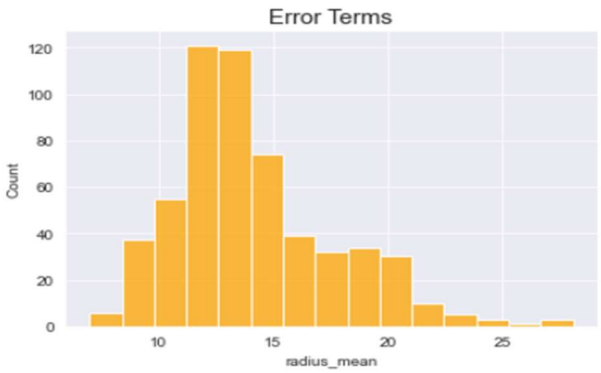




VIII) Histogram:

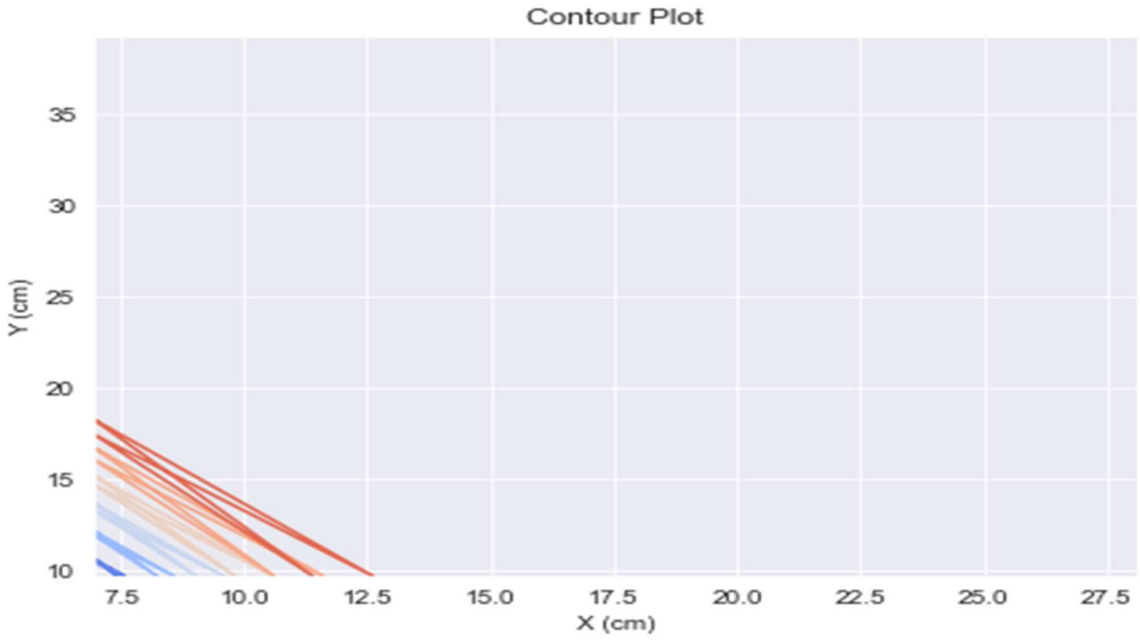
While a bar graph can be used to compare two entities, a histogram plot can be used when the data is still scattered. Despite having similar appearances, histograms and bar plots are employed in various contexts. This is represented by Matplotlib's hist() function.

```
fig = plt.figure()
sns.histplot(cancer_data['radius_mean'], bins = 15, color='Orange') #plotting histogram
plt.title('Error Terms', fontsize = 15)
plt.show()
```



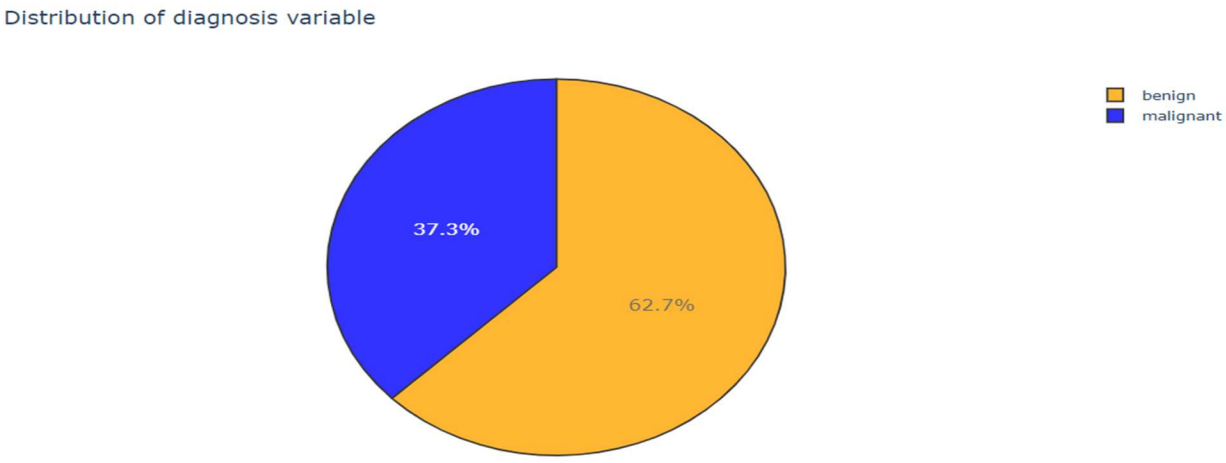
IX) Contour plot

Using constant z slices, often known as contours, a contour plot is a graphicl method for expressing a 3d surface in a two-dimensionl format. To put it another way, lines are drawn linking the (x,y) coordinates where a certain value of z appears.



X) Pie chart:

A circular graph known as a "pie plot" shows data as slices, components, or portions of a pie. Each pie slice represents a different item or category of data, and data analysts utilize them to illustrate the percentage or proportionate data. Pie () is used in Matplotlib to visualize it.



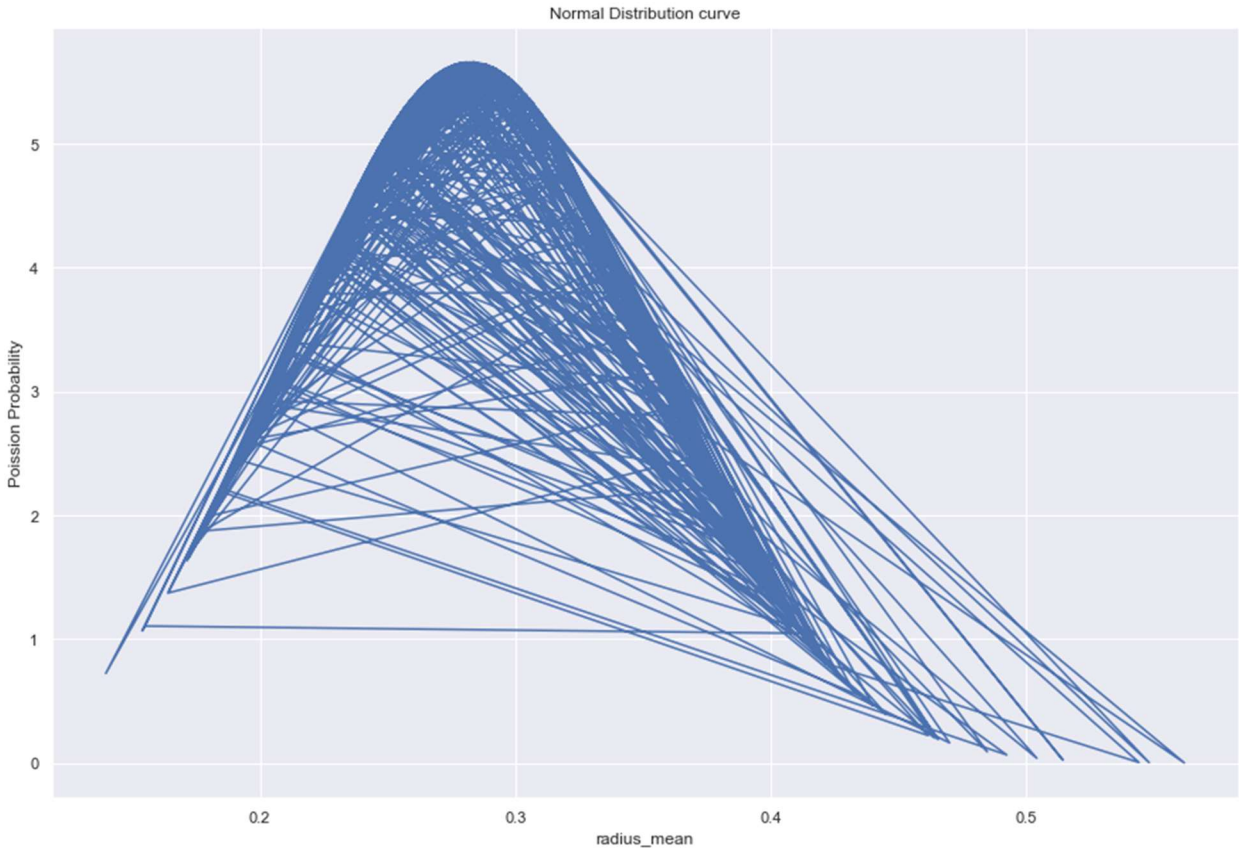
D) Probability Models:

I) Poisson Distribution:

A Poisson point process, also known as a Poisson process, is a group of points dispersed at random in mathematical space.

The Poisson process is frequently defined on a real line because of its many characteristics, where it can be viewed as a random (stochastic) process in one dimension. This further enables the creation of mathematical systems and the investigation of specific random events.

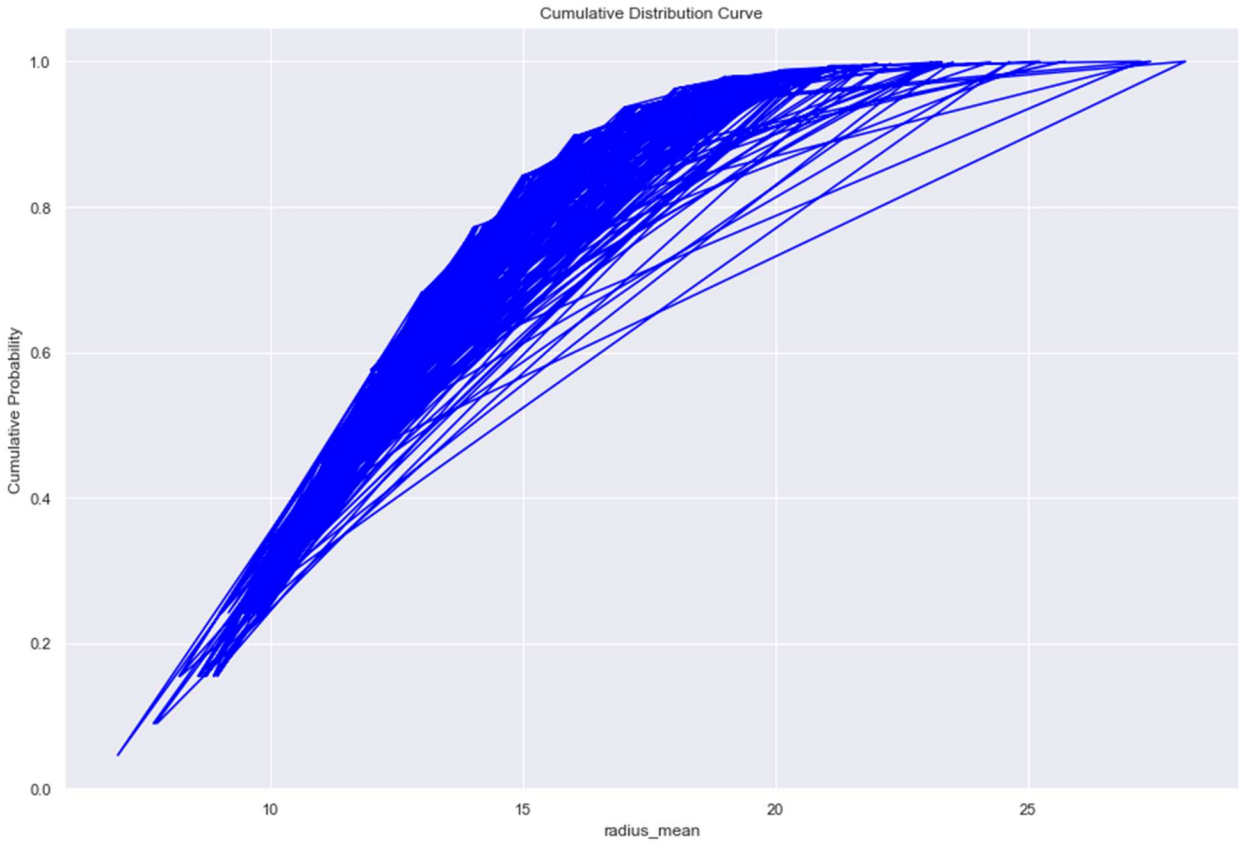
Every point in the process is stochastically independent from every other point, which is one of its key characteristics.



II) Poisson CDF (cumulative distribution function)

The Poisson cumulative distribution function allows you to determine the likelihood that an event will occur less frequently than or equally frequently than x times during a certain time or space interval if the event occurs on average times within that interval.

$$p = F(x|\lambda) = e^{-\lambda} \sum_{i=0}^{floor(x)} \frac{\lambda^i}{i!}.$$

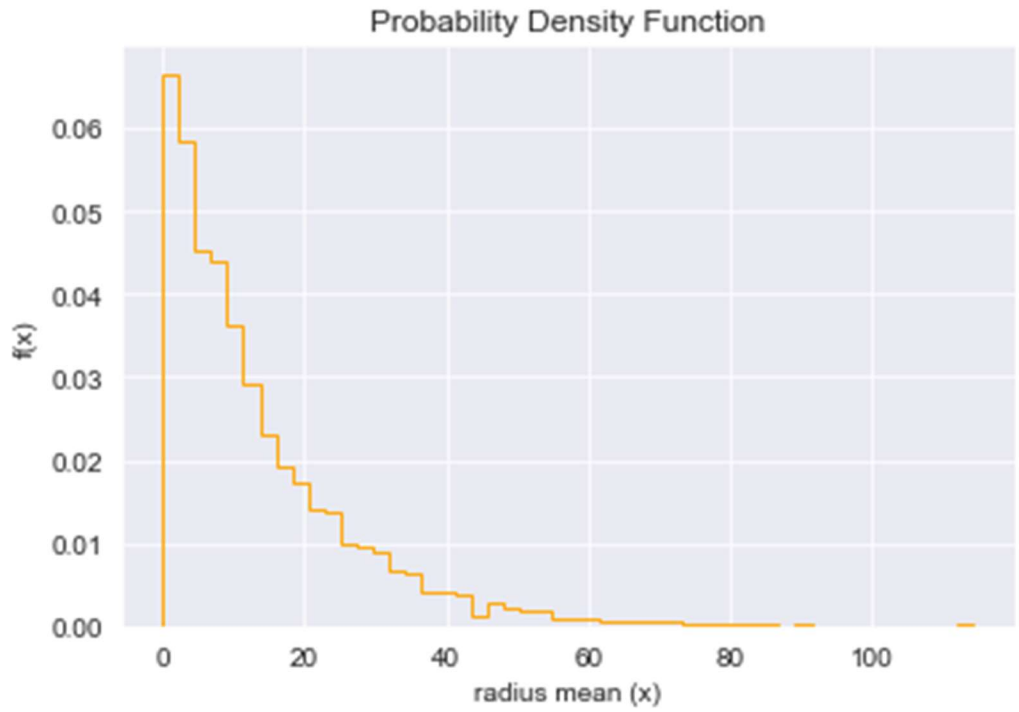


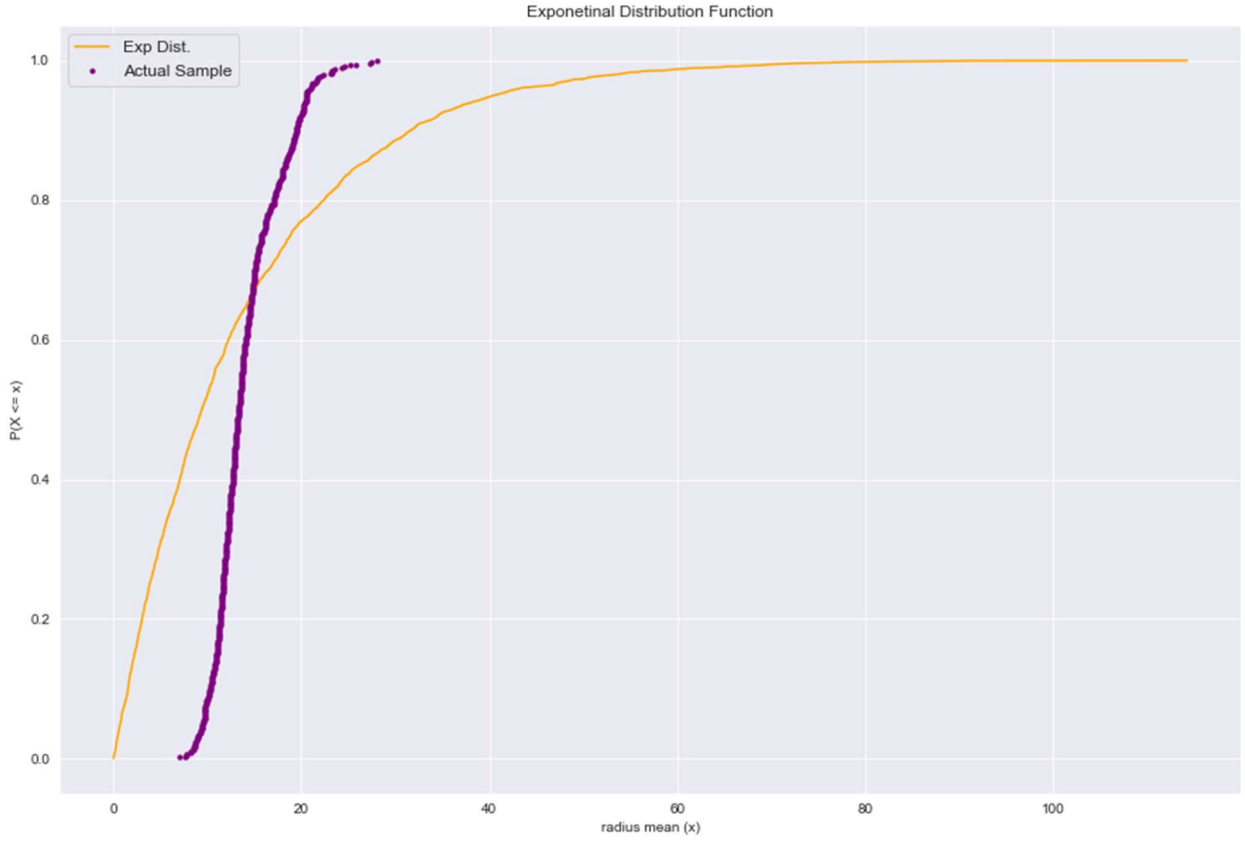
III) Exponential Distribution

The length of time we must wait until an event occurs is modeled using the exponential distribution, a probability distribution.

The cdf of a random variable X can be expressed as follows if it has an exponential distribution

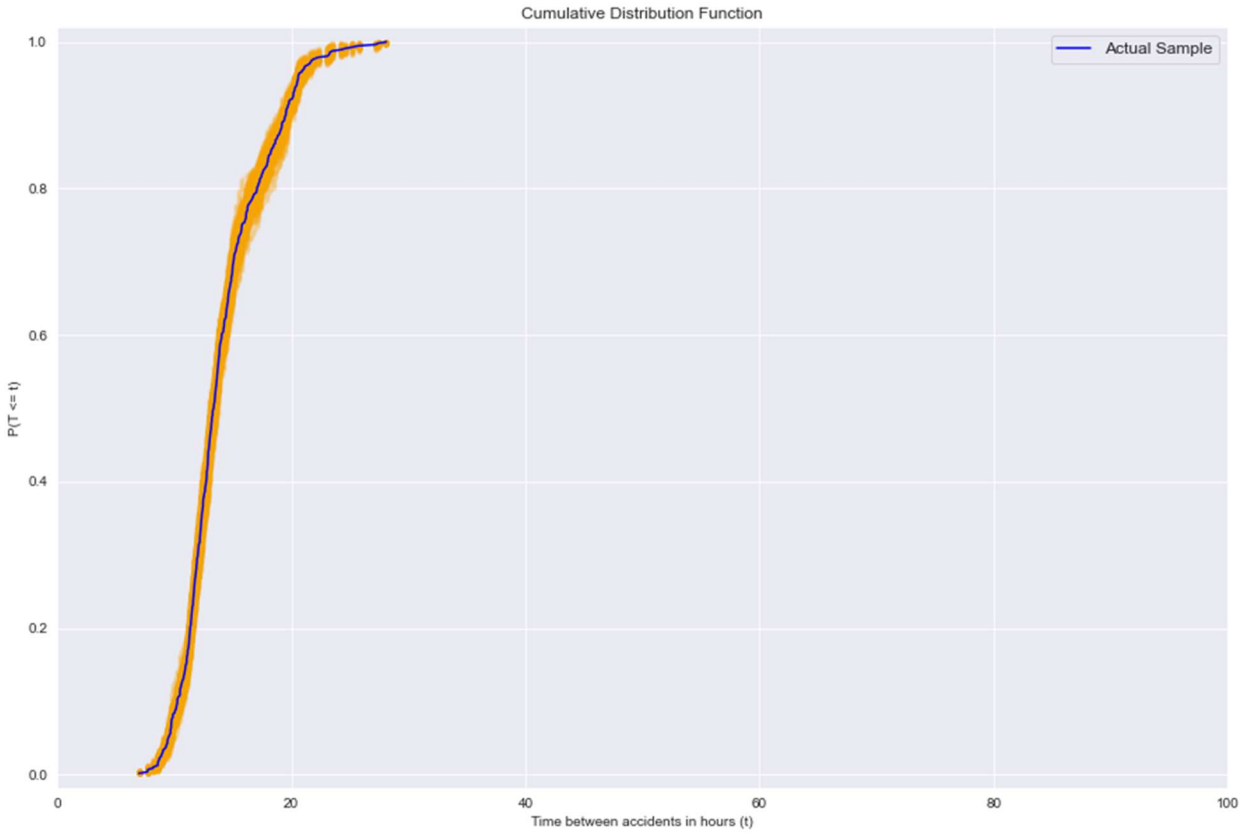
$$F(x; \lambda) = 1 - e^{-\lambda x}$$


Probability Density Function
f(x)
0.06
0.05
0.04
0.03
0.02
0.01
0.00
0
20
40
60
80
100
radius mean (x)

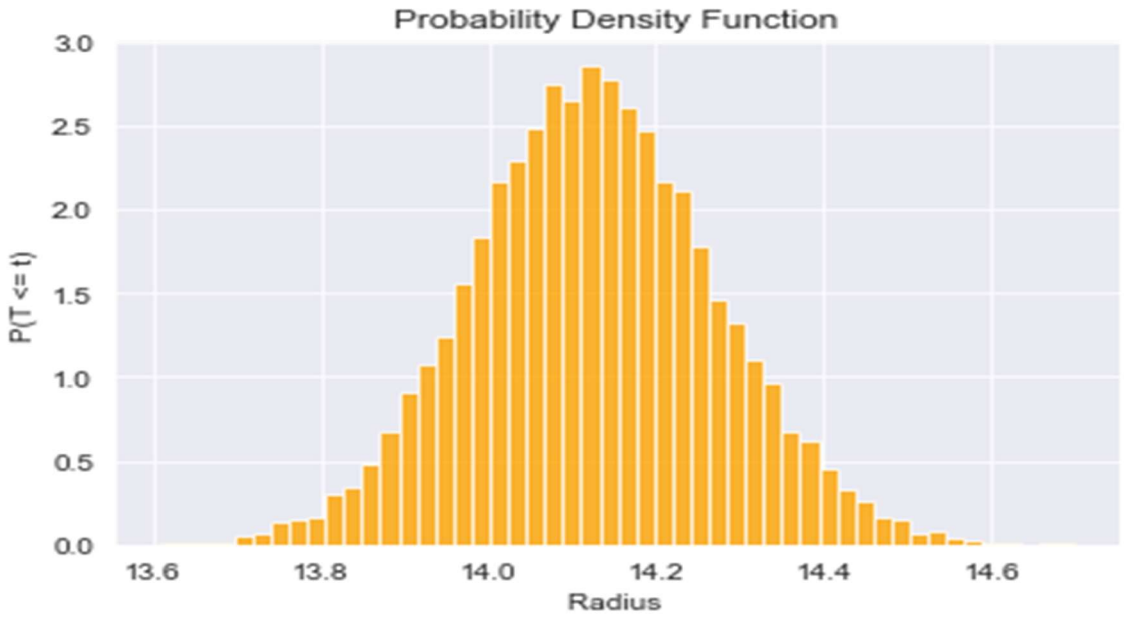

Exponetinal Distribution Function
Exp Dist.
Actual Sample
P(X <= x)
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100
radius mean (x)

IV) Confidence interval

A confidence interval is a range of values that are bound by the mean of the statistic and are most likely to contain an unidentified population parameter. Confidence level is the chance, or degree of certainty, that the confidence interval would include the true population parameter when a random sample is drawn numerous times.



V) Bootstrap Replicate



E) Statistical Models:

I) One Way anova:

a one-way ANOVA is performed when comparison is needed. Each participant's score on a category X predictor variable is used to determine their participation in a group. ANOVA is an extension of the t test. In an ANOVA, the categorical predictor variable could reflect naturally occurring groups or groups that are created and subsequently subjected to various treatments.

II) Calculating f_oneway ()

One-way anova can be calculated using the f_oneway(data). We need to import the SciPy. Stats package for this f_oneway. From this we can retrieve p-value.

F_oneway

```
from scipy.stats import f_oneway

f_oneway(cancer_data['radius_mean'],cancer_data['texture_mean'])
```

```
F_onewayResult(statistic=490.4551601701196, pvalue=1.2675924841686346e-90)
```

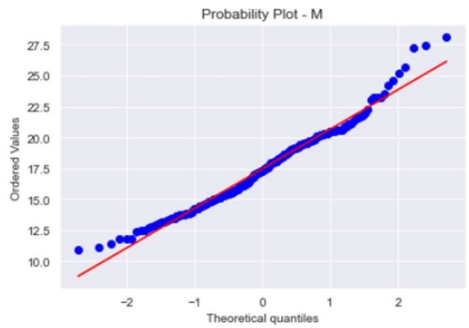
III) Printing Anova Table

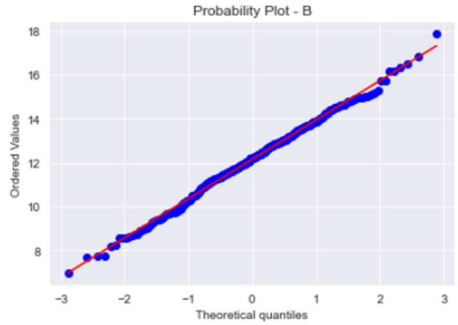
The sum of squares (SS), mean squares, and between-group and within-group sources of variance are all displayed in the ANOVA table (MS). The sum of the within- and between-group variances is the overall variation. The F value is a comparison of the mean squares between and within groups (MS). The degree of freedoms and F value are used to estimate the p value.

Source of Variation	SS	df	MS	F	P-value	F crit
Between Groups	109410.76756	1	109410.76756	18829.543546	0.0	5.050677
Within Groups	3294.604835	567	5.810591			
Total	112705.372395	568	198.424951			

IV) Normality assumption check

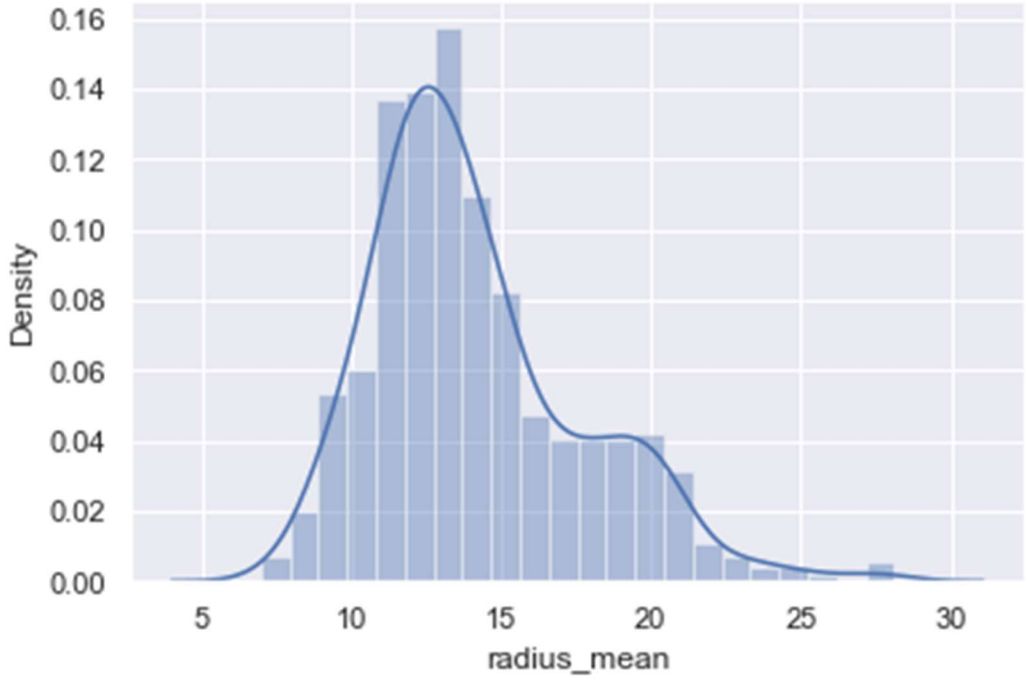
When the study is being planned, this assumption is examined. All groups are thus mutually exclusive, i.e., a person can only be a member of one group. Additionally, this implies that the data are not from repeated measures. This prerequisite is fulfilled in this instance. When a model is derived from an ANOVA or regression framework, the assumption of normality is checked on the model's residuals. The Shapiro-Wilk test is one strategy for examining the validity of the assumption of normality.





V) Type 1 errors:

A Type I error in statistical hypothesis testing is simply the rejection of the actual null hypothesis. Type I errors are frequently referred to as false positive errors. In other words, it makes a misleading assumption about the existence of a nonexistent phenomenon. We should always know that a type I error does not indicate that we accept the alternative hypothesis of an experiment in error.



VI) Summary of both sample 1 and sample2:

The Summary of the dataset can be easily calculated in python. This can be done by calling stats. Describe (data sample). Below is the image for code in python.

Sample 1 Summary

```
k = 100
sample1 = np.random.choice(pop,100,replace=True)

print ("Sample 1 Summary")
stats.describe(sample1)
```

```
Sample 1 Summary
DescribeResult(nobs=100, minmax=(6.981, 27.22), mean=14.262279999999999, variance=13.17138598141414, skewness=0.9751117048030952, kurtosis=0.8729699629107515)
```

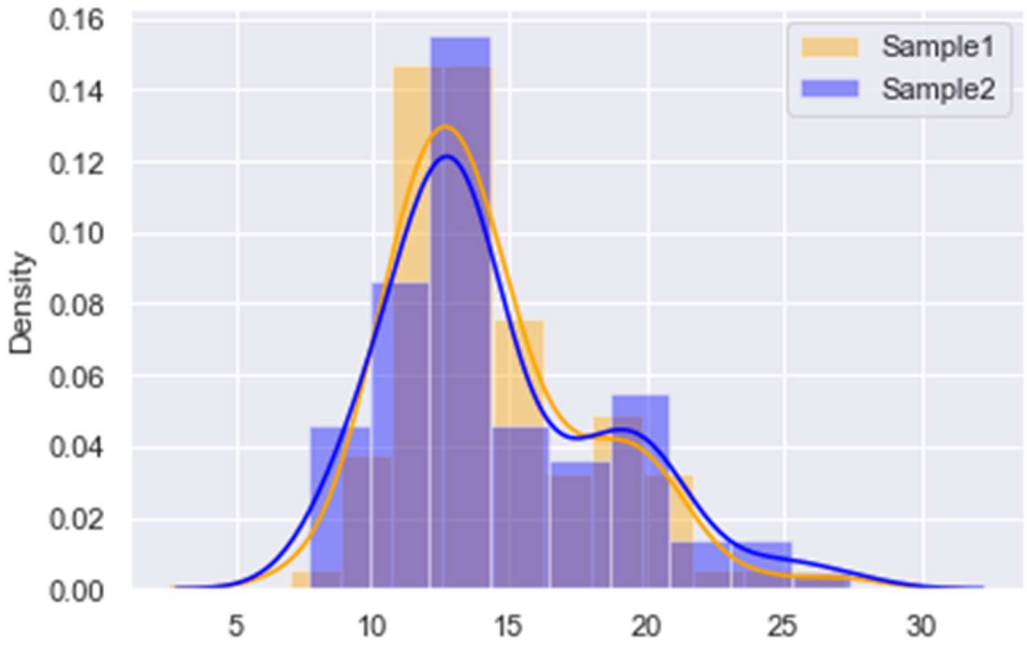
Sample 2 Summary

```
sample2 = np.random.choice(pop,100,replace=True)
print ("Sample 2 Summary")
stats.describe(sample2)
# test the sample means
```

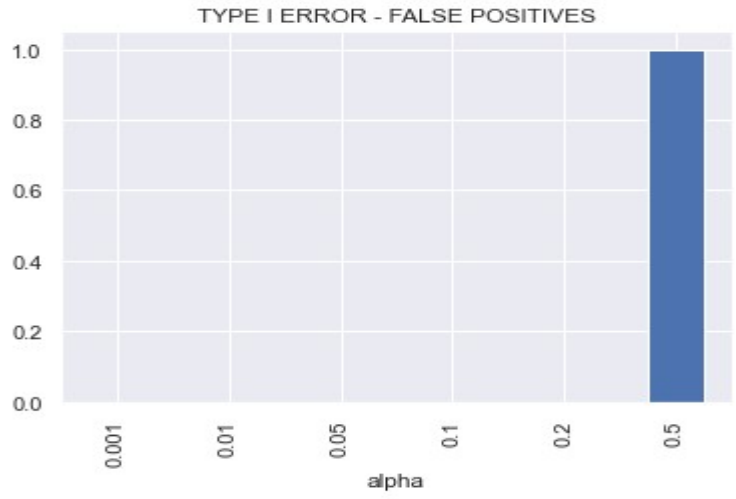
```
Sample 2 Summary
DescribeResult(nobs=100, minmax=(7.691, 27.42), mean=14.47933, variance=16.469118991010102, skewness=0.936837425830828, kurtosis=0.43948561492321403)
```

VII) Distplot:

The below graph shows the density of sample 1 and sample 2.



VIII) T-test on samples

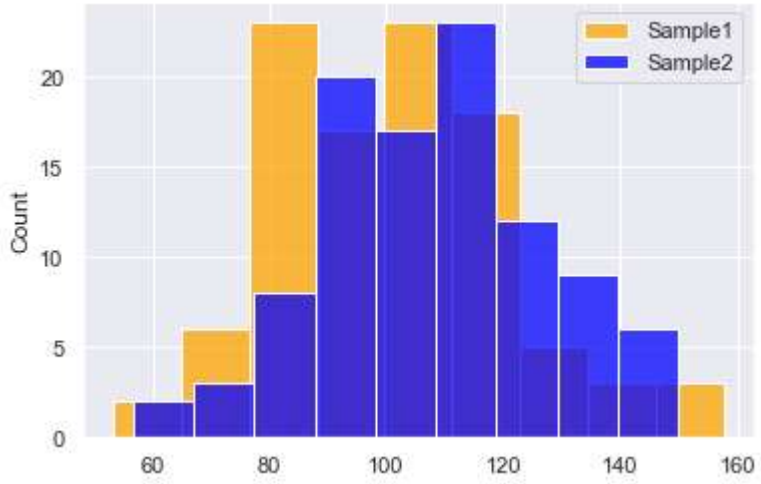


IX) Type II errors :

A type II error, often known as a false negative finding or conclusion, happens when a faulty null hypothesis is not rejected. A type II error has happened in statistical hypothesis testing if a faulty

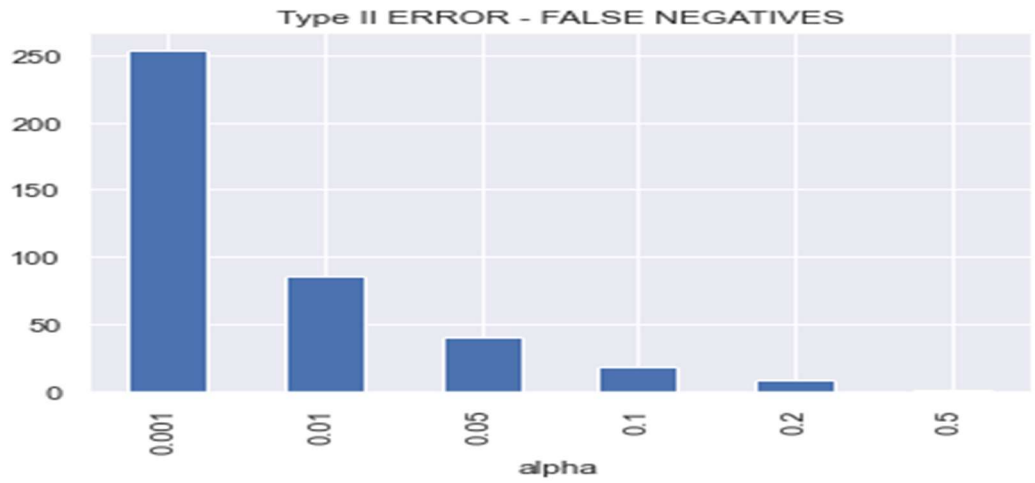
null hypothesis is not rejected when it ought to have been. A genuine null hypothesis is rejected while a false null hypothesis is accepted in this kind of error.

X) Displot for Type II errors:



XI) Type II ERROR - FALSE NEGATIVES

We can clearly see that value of alpha is decreases from .001 to 0.5

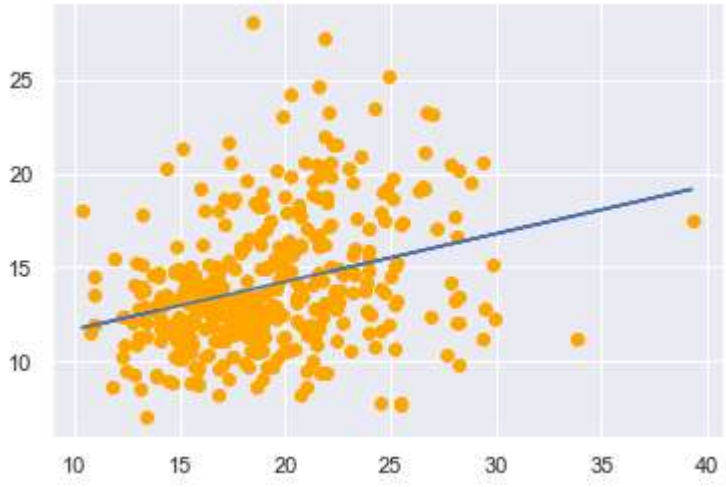


F) Implementing linear regression

Modeling the relationship between two continuous variables using simple linear regression. Predicting the value of an output variable (or response) from the value of an input variable (or predictor variable) is frequently the goal.

I) Best fit regression line

The phrase "line of best fit" describes a line that best depicts the relationship between the data points in a scatter plot. The geometric equation for the line is often determined by statisticians using the least squares approach, also known as ordinary least squares (OLS), either manually or using software.



II) Residual analysis

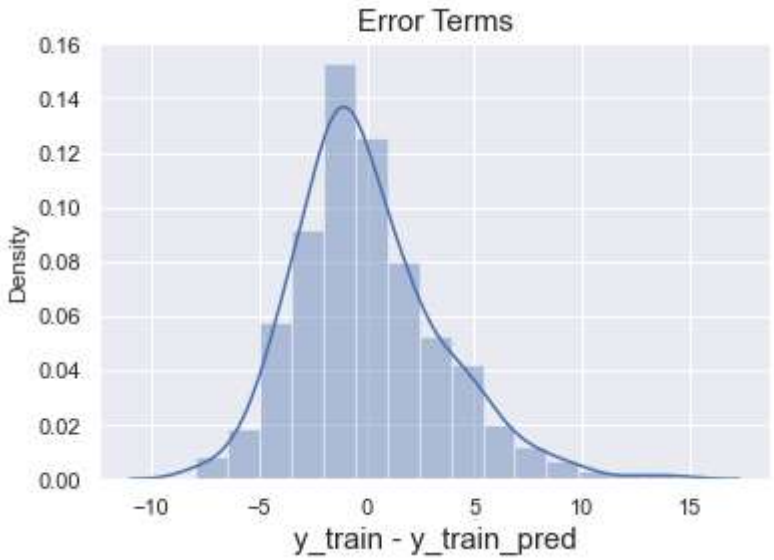
The term "residuals" refers to the discrepancies between the model's one-step predicted output and the measurd output from the validation data set. As a result, residuals are the portion of the validation data that the model is unable to account for.

Two tests make up residual analysis: the independence test and the whiteness test.

Residual analysis

```
# Predicting y_value using traingn data of X
y_train_pred = lr.predict(X_train_sm)

# Creating residuals from the y_train data and predicted y_data
res = (y_train - y_train_pred)
```

```
# Plotting the histogram using the residual values
fig = plt.figure()
sns.distplot(res, bins = 15)
plt.title('Error Terms', fontsize = 15)
plt.xlabel('y_train - y_train_pred', fontsize = 15)
plt.show()
```



III) Calculating r2- score:

Code in python for calculating r2-score

```
# Importing r2_square
from sklearn.metrics import r2_score

# Checking the R-squared value
r_squared = r2_score(y_test, y_test_pred)
r_squared
```

```
0.11955670316468836
```

We can see that value is 0.11

IV) Visualizing the line on test data :

Below graph shows the fit of test values of x_test and y_test as scatter and predicted values are shown in the form of a line.

4) Model Implementation

A) Label Encoder:

Using label encoder to decode the diagnosis parameters. The labels of 'M' and 'B' will be replaced as 0 and 1. From this we can do the desired operations of our dataset.

```
from sklearn.preprocessing import LabelEncoder
labelencoder_Y = LabelEncoder()
cancer_data.diagnosis = labelencoder_Y.fit_transform(cancer_data.diagnosis)
```

B) Train Test Splitting

It is a quick and simple technique to complete, and the outcomes let you compare how well machine learning methods work when applied to your particular predictive modeling issue. Although the process is straightforward to use and understand, there are some circumstances in which it should not be applied, such as when the dataset is tiny or when further setup is necessary, such as when it is applied for classification and the dataset is unbalanced.

Importing the header file

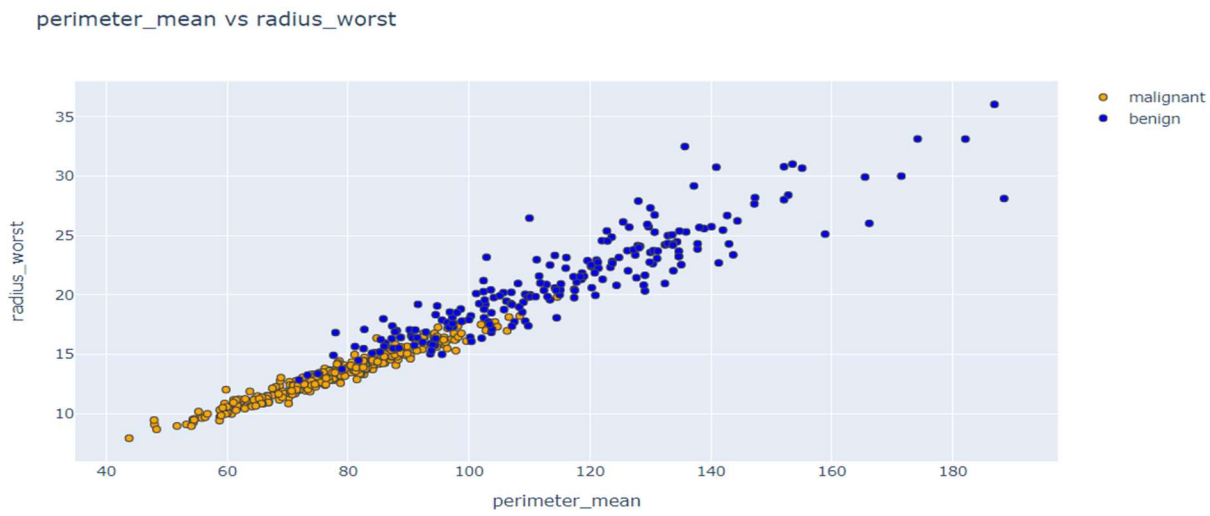
```
from sklearn.model_selection import train_test_split
```

C) Feature Selection

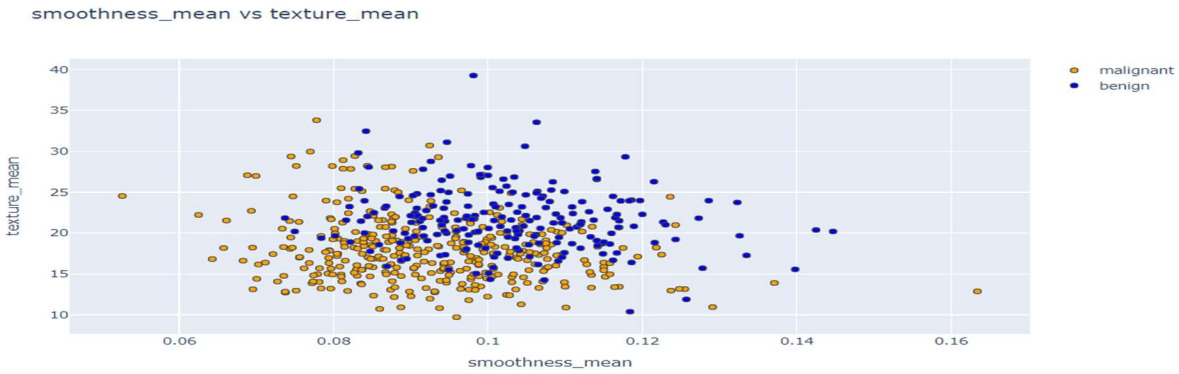
A predictive model's input variables are minimized through the process of feature selection.

This can be classified positive correlated features, uncorrelated features and negative correlated features.

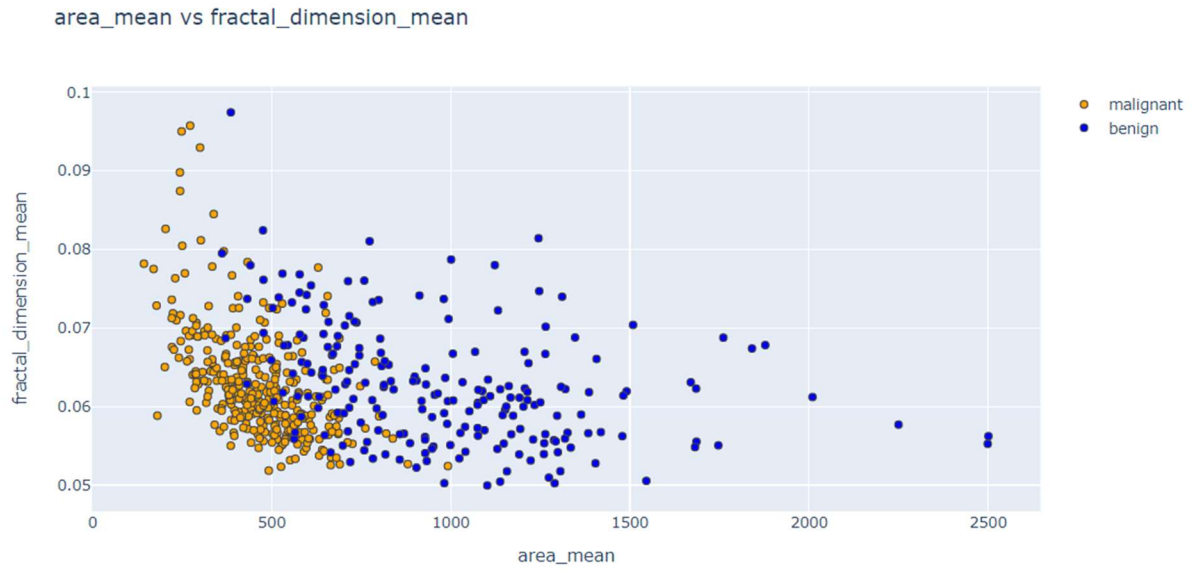
Positive correlated



Un correlated features



Negative corelated features



Feature selection is the process of deciding which current features will be most helpful in training the model. In our implementation we are taking 6 columns which are dependent on the data.

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

targeted_feature = 'diagnosis'

len(prediction_feature)
```

```
6
```

Split the dataset into Training Set and Testing Set by 33% and set the 15 fixed records

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

print(X_train)
```

D) Perform Feature Standard Scaling

A technique for normalizing the variety of independent variables or features in data is called feature scaling. It is typically carried out during the data preprocessing step and is sometimes referred to as data normalization in the context of data processing.

By eliminating the mean and scaling to the unit variance, standardize the features.

A sample x's average score is determined as follows:

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

Code for implementation:

```
sc = StandardScaler()

X_train = sc.fit_transform(X_train)
X_test = sc.fit_transform(X_test)
```

5) ML Model Selecting and Model Prediction

A) Model Building

In this we have created a function model building which have five arguments. From this function we get prediction, accuracy score.

Arguments that need to be passed through the function.

1. Model which is Machine Learning Model object
2. Feature Training data
3. Feature Testing data
4. Targeted Training data
5. Targeted Testing data

```
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)
```

B) Model Implementation and Preliminary results

IT depends on the application or the situation that we consider whether the accuracy is important, or the specificity is important in that or any other metric. For this case since we are trying to analyze and predict if a person has breast cancer or not, the model can have a chance of classifying a

person without breast cancer as having one. But cannot tolerate to predict a person with breast cancer that he/she is benign.

This can be measured by the False Negative Rate this has to be low It can be obtained by the sensitivity form the Classification Report.

The False Negative Rate is the proportion of the actual positives but are predicted as negatives. It is called as sensitivity.

Implementation of Logistic Regression

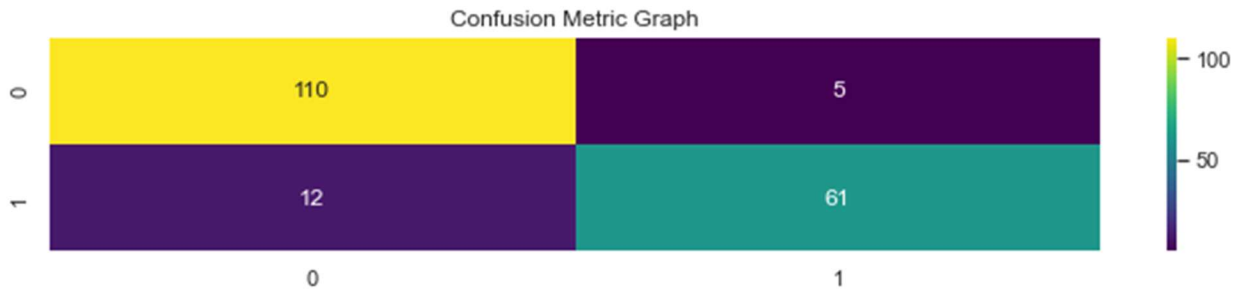
a) Classification report:

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

The above image shows the classification report for the Logistic regression
The Actual Positives are: 115, F1-score for this is 93%
The Predicted negatives are: 73.
The prediction for this algorithm is nearly 91%

b) Confusion Matrix



From the above confusion matrix, we can classify the TP, TN, FP and FN.

The true positive in this case is 110 which means people who have been identified as malignant nodes, this means they truly has breast cancer.

The True Negative is 5 which means these people are healthy and they don't have cancer, but our model predicted them to have a breast cancer.

The False positive are misclassified with the cancer, but they are healthy the total is 12. This is also known as the Type I – error.

The False Negative is misclassified as healthy even though they have been diagnosed this value is 61.

C) Applying K-Fold to the data

By applying the K-fold to our algorithm we can test our data with various types of data. We can see that our model has an overall accuracy of 90%.

```
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
```

D) Hyper Tunning the ML model

From this hyper tunning we can get the best score, best fit and best estimator for our algorithm.

```
# Pick the model
model = LogisticRegression()
param_grid = [
              {'fit_intercept': [1],
               'penalty' : ['12'],
             'C' : [0.001 , 0.1,1]
               }
]
gsc = GridSearchCV(model,param_grid) # 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
0.9132262474367737

 Best Estinator is
LogisticRegression(C=1, fit_intercept=1)

 Best Parametes are
{'C': 1, 'fit_intercept': 1, 'penalty': '12'}
```

Implementation of RandomForestClassifier

A) Classification Report:

The above image shows the classification report for the RandomForestClassifier

The Actual Positives are: 115, F1-score for this is 94%

The Predicted negatives are: 73.

The prediction for this algorithm is nearly 93%

```
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

B) Confusion Matrix:

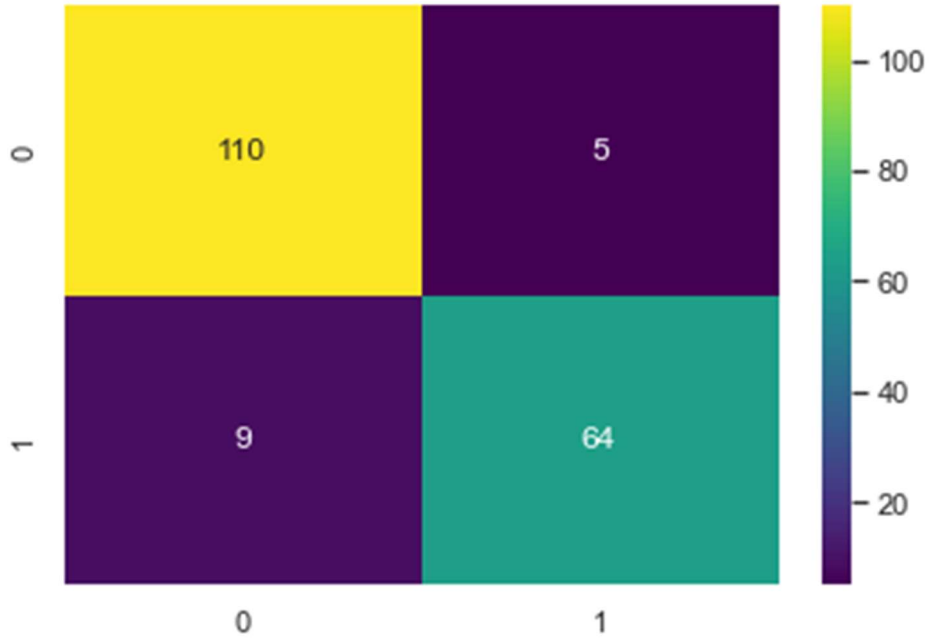
From the above confusion matrix we can classify the TP, TN, FP and FN.

The true positive in this case is 110 which means people who have been identified as malignant nodes, this means they truly has breast cancer.

The True Negative is 5 which means these people are healthy and they don't have cancer, but our model predicted them to have a breast cancer.

The False positive are misclassified with the cancer, but they are healthy the total is 9. This is also known as the Type I – error.

The False Negative is misclassified as healthy even though they have been diagnosed this value is 64.



C) Applying K-Fold to the data

By applying the K-fold to our algorithm we can test our data with various types of data. We can see that our model has an overall accuracy of 100%.

```
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
```

D) Hyper Tunning the ML model

From this hyper tunning we can get the best score , best fit and best estimator for our algorithm. All the best parameters are shown in the below output. And best estimator with number of estimators are also shown.

```
# Pick the model
model = RandomForestClassifier()


# Tunning Params
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=50, n_estimators=200)

 Best Parametes are
{'bootstrap': True, 'max_depth': 50, 'max_features': 'auto', 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 200}
```

Implementation of DecisionTreeClassifier

A) Classification Report:

The above image shows the classification report for the DecisionTreeClassifier

The Actual Positives are: 115, F1-score for this is 93%

The Predicted negatives are: 73.

The prediction for this algorithm is nearly 91%

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

B) Confusion Matrix:

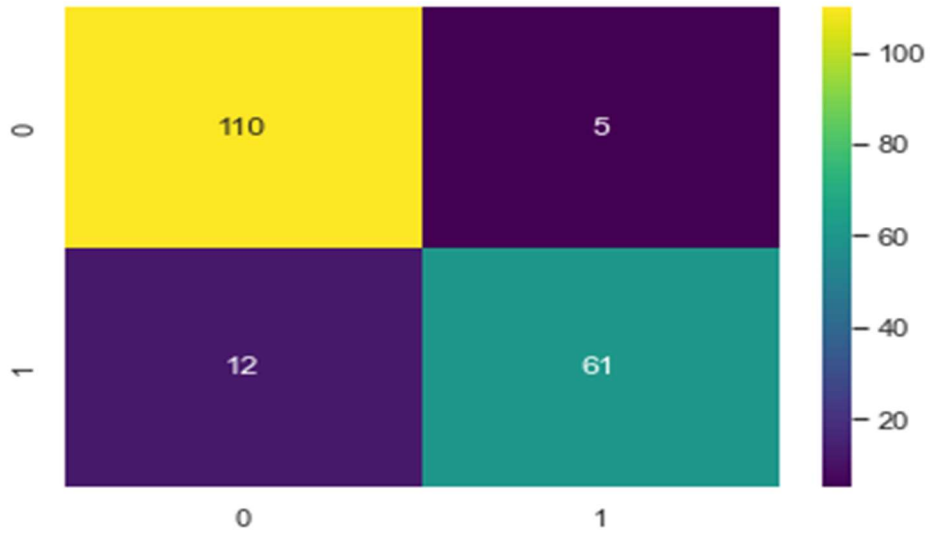
From the above confusion matrix we can classify the TP, TN, FP and FN.

The true positive in this case is 110 which means people who have been identified as malignant nodes, this means they truly has breast cancer.

The True Negative is 5 which means these people are healthy and they don't have cancer, but our model predicted them to have a breast cancer.

The False positive are misclassified with the cancer, but they are healthy the total is 12. This is also known as the Type I – error.

The False Negative is misclassified as healthy even though they have been diagnosed this value is 61.



C) Applying K-Fold to the data

By applying the K-fold to our algorithm we can test our data with various types of data. We can see that our model has an overall accuracy of 100%.

```
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
```

D) Hyper Tunning the ML model

From this hyper tunning we can get the best score , best fit and best estimator for our algorithm. All the best parameters are shown in the below output. And best estimator with number of estimators are also shown.

```
# Let's Implement Grid Search Algorithm

# Pick the model
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
0.9211875843454791

 Best Estinator is
DecisionTreeClassifier(max_features='sqrt', min_samples_leaf=5,
                       min_samples_split=3)

 Best Parametes are
{'max_features': 'sqrt', 'min_samples_leaf': 5, 'min_samples_split': 3}
```

Among the implementation of 3 algorithms, we can see that the accuracy for random forest algorithm is high. For the logistic and Decision Tree classifier the accuracy score is nearly same.

```
dataframe_pred.sort_values('accuracy_score', ascending=False)
```

	model_name	score	accuracy_score	accuracy_percentage
1	RandomForestClassifier	0.992126	0.925532	92.55%
0	LogisticRegression	0.916010	0.909574	90.96%
2	DecisionTreeClassifier	1.000000	0.909574	90.96%

7) **Deploy Model**

We have completed our work thus far. The deployment of our model in the production map is the final phase. Therefore, we must export our model and tie it to an API for a web application. With Pickle, we can export our model and store it in a model.pkl file that we can easily retrieve and use the Web App API to compute customized predictions.

Sample code for using pickle is shown below. The result will give the accuracy of algorithm

```
import pickle
model.fit(X_train, y_train)
# save the model to disk
filename = 'finalized_model.sav'
pickle.dump(model, open(filename, 'wb'))

# some time later...

# load the model from disk
loaded_model = pickle.load(open(filename, 'rb'))
result = loaded_model.score(X_test, y_test)
print(result)
```

```
0.9095744680851063
```

```
logistic_model = LogisticRegression()
logistic_model.fit(X_train, y_train)

filename = 'logistic_model.pkl'
pickle.dump(logistic_model, open(filename, 'wb'))
```

8) Project Management

Implementation status report

Work completed:

Description: In this project we have implemented the Data preprocessing, EDA and Implementing probability and statical methods, Implemented Decision Tree classification, Random Forest, and Logistic regression.

Responsibility (Task, Person)

Data- preprocessing and EDA: Kota Sai Teja

Probability and Statical methods: Veena Ravuri

Importing data and machine learning implementation: Dinesh Bhogadi

Contributions (members/percentage)

Veena Ravuri: 100%

Kota Sai Teja Jana: 100%

Dinesh Bhogadi: 100%

Work to be completed:

Description: Need to implement K-NN, Naive Bias and Gaussian NB. We are also trying to implement Neural Networks implementation.

Responsibility (Task, Person)

K-NN, Naive Bias: Veena Ravuri

Resampling, cross validation: Kota Sai Teja Jana

Gaussian NB. And Neural networks: Dinesh Bhogadi

Issues or concerns:

We don’t have any issues or concerns.

9) References:

1. Anshuman; Upendra Kumar , "Machine Learning model for detection of Breast Cancer" IEEE 2021

2. Meriem Amrane; Saliha Oukid;"Breast cancer classification using machine learning" IEEE 2018

3. Udit Pratap; Saurabh Chhabra,"Breast Cancer Prediction using Different Machine Learning Algorithms", IEEE 2021

GITHUB link:

https://github.com/Dinesh101010/Project_CSCE5310