ACKNOWLEDGEMENT:

In accomplishment of this project successfully, it is our genuine pleasure to express our deep felt thanks to all the people who have best owed up their pledged support throughout the project.
We extend gratitude to Keshav Memorial institute of technology for giving us this wonderful opportunity.
We are heart fully thankful to Prof. Neil Gogte Sir, Mrs.Deepa Madam and Mr.Aditya Godes Sir, for providing the facility and giving us the opportunity for such a conducive environment to complete the project.
We would like to express our sincere thanks of gratitude to our supervisors Mr.Balram Murthy Sir,Mr. Ajeet Jain Sir, Ms.M.Swetha Madam and also the supporting faculty for providing their valuable guidance, comments, suggestions and instructions throughout the course of the project which served as the major contributor towards the completion of the project. We came to know about new things and we are really thankful to them.
We would like to thank each other for working in a co-operative manner, sharing valuable ideas, thoughts and enlightening the team spirit, which made the project easy and accurate, so that we could complete it in the stipulated time.
Any omission in this brief acknowledgement doesn't mean lack of gratitude.

Thank you.

ABSTRACT

CONTEXT

Cancer has been characterized as a heterogeneous disease consisting of many different subtypes. The early diagnosis and prognosis of a cancer type have become a necessity in cancer research, as it can facilitate the subsequent clinical management of patients. Data science techniques can be used to discover hidden patterns and relationships. Models developed from these techniques are useful for medical practitioners to make right decisions and can improve our understanding of cancer prediction.

PROBLEM

It is complicated to look at huge piles of data. Using graphical representations and other visual aids, we can show the relationships in a more readable and meaningful way. For this we make use of various libraries available in python. Given the speed limit, we can predict a parameter by making using of classification models, we will be able to predict whether the person has a breast cancer or not.

PROPOSED WORK

We predict if the tumour which is present in the breast is benign or malignant based on some of the parameters like concavity, radius, perimeter, smoothness etc of the tumor and by making use of regression models .Plotting graphs and analyzing them we find the relationships between different parameters. Ultimately these predictions are going to save person's life from cancer and we can reduce the death causes due to cancer.

INTRODUCTION WITH DOMAIN ANALYSIS

Breast cancer (BC) is the most common cancer in women, affecting about 10% of all women at some stages of their life. In recent years, the incidence rate keeps increasing and data show that the survival rate is 88% after five years from diagnosis and 80% after 10 years from diagnosis . Early prediction of breast cancer is one of the most crucial works in the follow-up process. Data Science methods can help to reduce the number of false positive and false negative decisions . Machine Learning techniques can be used to discover hidden patterns and relationships. Models developed from these techniques are useful for medical practitioners to make right decisions and can improve our understanding of cancer prediction.

Our project that aims to better understand the data and predict if the person has breast cancer or not from a given dataset of various tests performed by professionals. We predict if the tumor which is present in the breast is benign or malignant .In most of the cases the benign tumor is not much of a danger to persons health where malignant tumor spread and invade the neighboring tissues so they pose a serious risk for a person's health and has to be treated immediately .

PROBLEM STATEMENT:

A problem statement should be able to answer all the five "Wh" questions. Breast Cancer. The early diagnosis and prognosis of a cancer type have become a necessity in cancer research, as it can facilitate the subsequent clinical management of patients. The tumor which is present in the breast could be benign or malignant base on some of the parameters like concavity, radius, perimeter, smoothness etc of the tumor. In most of the cases the benign tumor is not much of a danger to persons health where malignant tumor spread and invade the neighboring tissues so they pose a serious risk for a person's health and has to be treated immediately .

METHODOLOGY:

A problem stated is half-solved. We need to devise an approach to solve it fully. We have carefully analyzed the data and tried to bring out interesting and important pattems using various graphical notations. We brought out the relationships between different attributes like " concavity of tumor ", "radius of tumor" ","perimeter of tumor". We used Logistic Regression concept of Machine Leaming in order to make predictions as it suits our dataset and most of our attributes are numeric data.

HARDWARE AND SOFTWARE

- **System:** Intel® Core™ i5-6200U CPU @ 2.30GHz
- **Processor:** i5
- **Memory:** 8GB RAM.
- **Operating System:** Microsoft Windows Server, 64-bit OS.
- **Language:** Python
- **Software:** Anaconda – Jupyter
- **Anaconda Version:** Anaconda 3. Inc. 5.2.0.
- **Application version:** Jupyter Notebook 5.5.0.
- **Python Version:** Python 3.6.7.

LIBRARIES AND PACKAGES

```
import pandas as pd
import numpy as np
import seaborn as sb
import matplotlib.pyplot as plt
import pylab
from scipy.optimize import curve_fit
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix,accuracy_score,classification_report
from sklearn import tree
import pydot
from sklearn.model_selection import train_test_split
from IPython.display import Image,display
import pydotplus
from sklearn.metrics import accuracy_score,roc_curve,auc
```

pandas: It is a package which provides fast, flexible, and expressive data structures designed to make working with structured data easy.

seaborn: It is a visualization library based on matplotlib. It provides a high-level interface for drawing attractive statistical graphics.

matplotlib.pyplot: It is a plotting library used for 2D graphics.

numpy: It is package that supports large, multi-dimensional arrays and matrices, along with large library of functions to operate on these arrays.

sklearn: It is a library that contains various algorithms and is designed to interoperate with Python numerical and scientific libraries.

pylab: It combines the numerical module numpy with the graphical plotting module pyplot.

scipy: It includesmodules for statistics, optimization, integration, linear algebra, Fourier transforms, signal and image processing, ODE solvers, and more.

pydot: This package includes an interface to GraphViz , with classes to represent graphs and dump them in the DOT language , and a parser from DOT.

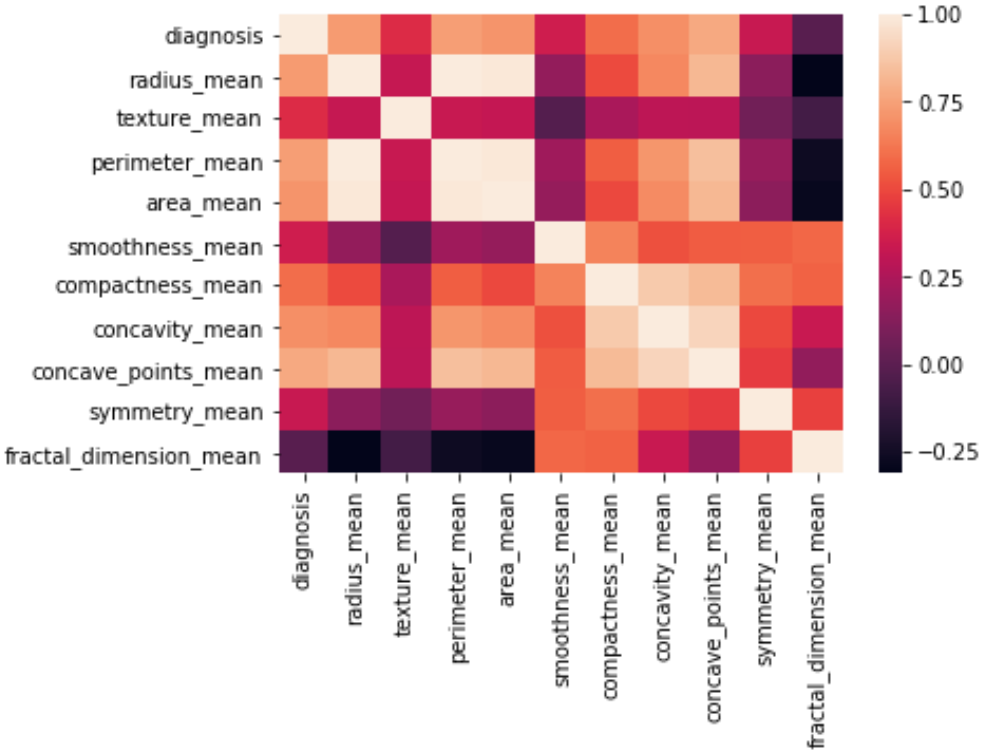
pydotplus:pydotplus is improved version of pydot.

EXPLORATORY DETAILS OF DATA ANALYSIS

EDA is a phenomenon under data analysis used for gaining a better understanding of data aspects like:
– main features of data
– variables and relationships that hold between them
– identifying which variables are important for our problem

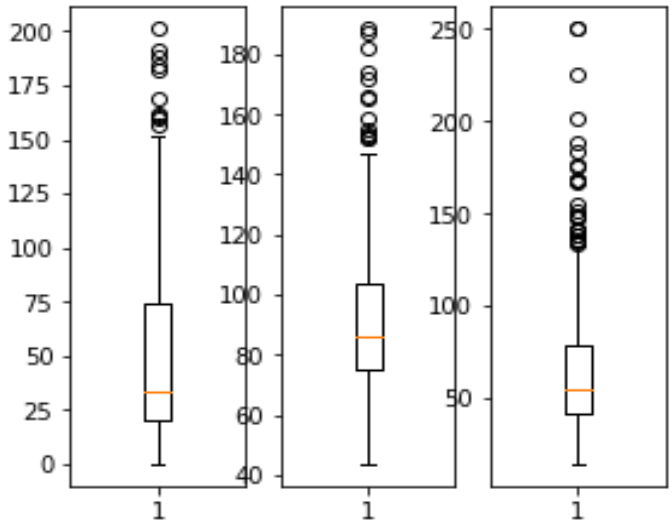
Heat Map:

It is often desirable to show data which depends on two independent variables as a color coded image plot. This is often referred to as a heatmap.



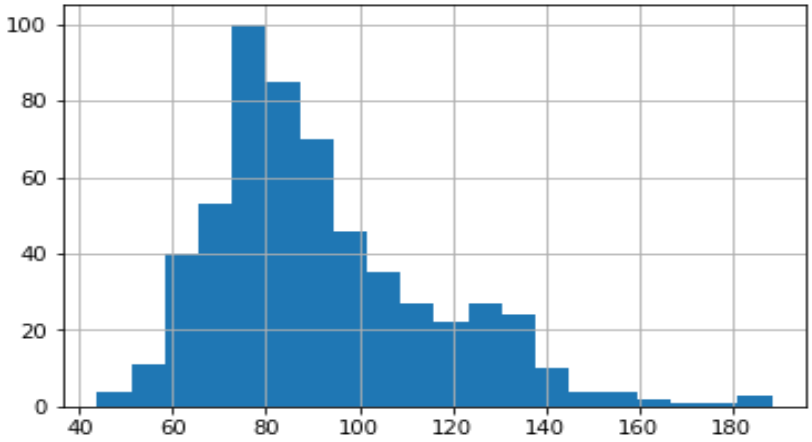
Box plot:

Boxplot can be drawn calling Series.plot.box() and DataFrame.plot.box(), or DataFrame.boxplot() to visualize the distribution of values within each column.Using boxplots we can analyse the outliers of data.



Histogram:

A histogram is a great tool for quickly assessing a probability distribution that is intuitively understood by almost any audience.



PREPROCESSING

In [14]:

```
import pandas as pd
import numpy as np
df=pd.read_csv("E:\\cancer.csv")
df1=df.iloc[:,1:]
print(df1.head(5))
print("no of rows and columns",df1.shape)
```

```
  diagnosis  radius_mean  texture_mean  perimeter_mean  area_mean  \
0         M        17.99         10.38          122.80     1001.0
1         M        20.57         17.77          132.90     1326.0
2         M        19.69         21.25          130.00     1203.0
3         M        11.42         20.38           77.58      386.1
4         M        20.29         14.34          135.10     1297.0

   smoothness_mean  compactness_mean  concavity_mean  concave_points_mean  \
0          0.11840           0.27760          0.3001              0.14710
1          0.08474           0.07864          0.0869              0.07017
2          0.10960           0.15990          0.1974              0.12790
3          0.14250           0.28390          0.2414              0.10520
4          0.10030           0.13280          0.1980              0.10430

   symmetry_mean            ...              radius_worst  texture_worst  \
0         0.2419            ...                     25.38          17.33
1         0.1812            ...                     24.99          23.41
2         0.2069            ...                     23.57          25.53
3         0.2597            ...                     14.91          26.50
4         0.1809            ...                     22.54          16.67

   perimeter_worst  area_worst  smoothness_worst  compactness_worst  \
0           184.60      2019.0            0.1622             0.6656
1           158.80      1956.0            0.1238             0.1866
2           152.50      1709.0            0.1444             0.4245
3            98.87       567.7            0.2098             0.8663
4           152.20      1575.0            0.1374             0.2050

   concavity_worst  concave points_worst  symmetry_worst  \
0           0.7119                0.2654          0.4601
1           0.2416                0.1860          0.2750
2           0.4504                0.2430          0.3613
3           0.6869                0.2575          0.6638
4           0.4000                0.1625          0.2364

   fractal_dimension_worst
0                  0.11890
1                  0.08902
2                  0.08758
3                  0.17300
4                  0.07678

[5 rows x 31 columns]
no of rows and columns (569, 31)
```

In [ ]:

Including 11 coloumns which are useful

In [15]:

```
df1=df.iloc[:,1:11]
print(df1.head(5))
print("no of rows and columns",df1.shape)
```

```
  diagnosis  radius_mean  texture_mean  perimeter_mean  area_mean  \
0         M        17.99         10.38          122.80     1001.0
1         M        20.57         17.77          132.90     1326.0
2         M        19.69         21.25          130.00     1203.0
3         M        11.42         20.38           77.58      386.1
4         M        20.29         14.34          135.10     1297.0

   smoothness_mean  compactness_mean  concavity_mean  concave_points_mean  \
0          0.11840           0.27760          0.3001              0.14710
1          0.08474           0.07864          0.0869              0.07017
2          0.10960           0.15990          0.1974              0.12790
3          0.14250           0.28390          0.2414              0.10520
4          0.10030           0.13280          0.1980              0.10430

   symmetry_mean
0         0.2419
1         0.1812
2         0.2069
3         0.2597
4         0.1809
no of rows and columns (569, 10)
```

converting categorial data to numeric data

In [16]:

```
df1['diagnosis']=df1['diagnosis'].astype('category')
df1['diagnosis']=df1['diagnosis'].cat.codes
df1.head()
```

Out[16]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	com
0	1	17.99	10.38	122.80	1001.0	0.11840	
1	1	20.57	17.77	132.90	1326.0	0.08474	
2	1	19.69	21.25	130.00	1203.0	0.10960	
3	1	11.42	20.38	77.58	386.1	0.14250	
4	1	20.29	14.34	135.10	1297.0	0.10030	

In [ ]:

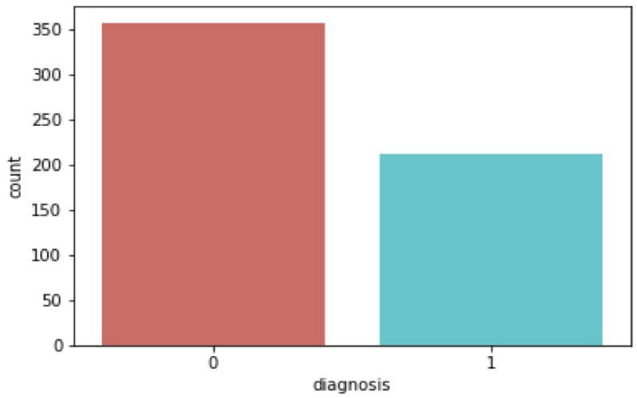
Checking about diagnosis

In [17]:

```
import seaborn as sb
sb.countplot(x="diagnosis",data=df1,palette='hls')
#1 means likely to have cancer
#0 means is not likely to have cancer
```

Out[17]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x17e5d346908>
```



Checking for null values

In [18]:

```
df1.isnull().sum()
#NO null values found
```

Out[18]:

```
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
dtype: int64
```

In [ ]:

```

```

In [ ]:

```

```

Finding covarience of the dataframe

In [19]:

```
df1.cov()
```

Out[19]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean
diagnosis	0.234177	1.244954	0.864145	8.732438	120.738222
radius_mean	1.244954	12.418920	4.907582	85.447142	1224.483409
texture_mean	0.864145	4.907582	18.498909	34.439759	485.993787
perimeter_mean	8.732438	85.447142	34.439759	590.440480	8435.772345
area_mean	120.738222	1224.483409	485.993787	8435.772345	123843.554318
smoothness_mean	0.002440	0.008454	-0.001415	0.070836	0.876178
compactness_mean	0.015246	0.094197	0.053767	0.714714	9.264931
concavity_mean	0.026864	0.190128	0.103692	1.387234	19.244924
concave_points_mean	0.014583	0.112475	0.048977	0.802360	11.241958
symmetry_mean	0.004384	0.014273	0.008419	0.121922	1.459596

Finding correlation matrix between each attribute

In [20]:

```
df1.corr()
```

Out[20]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smo
diagnosis	1.000000	0.730029	0.415185	0.742636	0.708984	
radius_mean	0.730029	1.000000	0.323782	0.997855	0.987357	
texture_mean	0.415185	0.323782	1.000000	0.329533	0.321086	
perimeter_mean	0.742636	0.997855	0.329533	1.000000	0.986507	
area_mean	0.708984	0.987357	0.321086	0.986507	1.000000	
smoothness_mean	0.358560	0.170581	-0.023389	0.207278	0.177028	
compactness_mean	0.596534	0.506124	0.236702	0.556936	0.498502	
concavity_mean	0.696360	0.676764	0.302418	0.716136	0.685983	
concave_points_mean	0.776614	0.822529	0.293464	0.850977	0.823269	
symmetry_mean	0.330499	0.147741	0.071401	0.183027	0.151293	

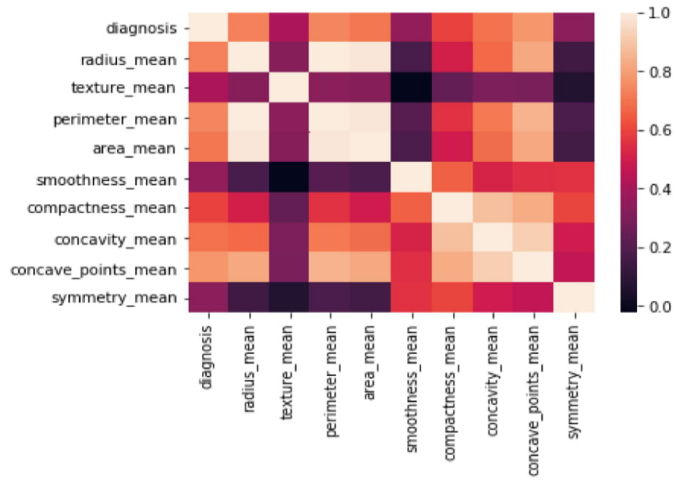
Plotting heatmap to understand the correlation

In [21]:

```
import seaborn as sb
sb.heatmap(df1.corr())
```

Out[21]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x17e5d3934e0>
```



Checking which attribute has the highest correlation with "diagnosis"

In [22]:

```
df1.corr()['diagnosis'].sort_values()[::-1]
```

Out[22]:

```
diagnosis              1.000000
concave_points_mean    0.776614
perimeter_mean         0.742636
radius_mean            0.730029
area_mean              0.708984
concavity_mean         0.696360
compactness_mean       0.596534
texture_mean           0.415185
smoothness_mean        0.358560
symmetry_mean          0.330499
Name: diagnosis, dtype: float64
```

In [ ]:

```

```

In [ ]:

```
```

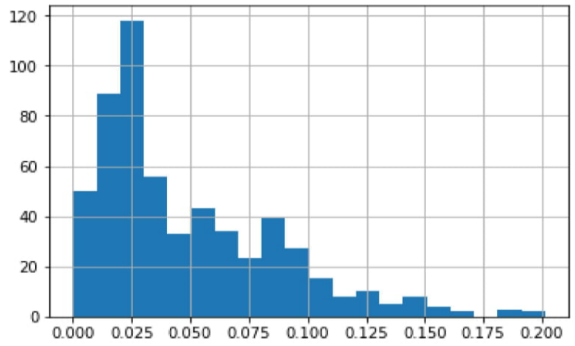
As concave_points_mean,perimter_mean,radius_mean have a greater correlation,plotting graphs for 3 attributes to analyse

In [23]:

```
df1['concave_points_mean'].hist(bins=20)
```

Out[23]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x17e5d4372b0>
```

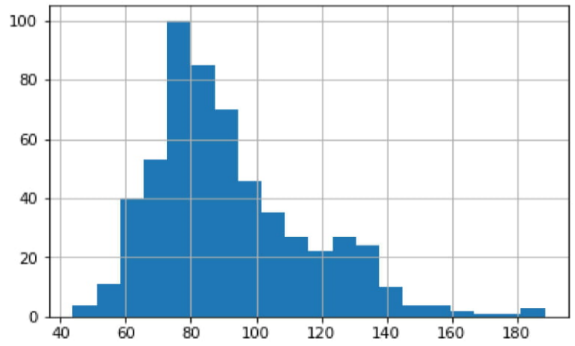


In [24]:

```
df1['perimeter_mean'].hist(bins=20)
```

Out[24]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x17e5d4a3828>
```

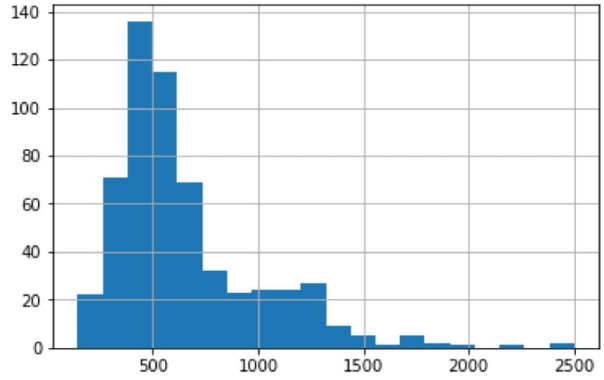


In [25]:

```
df1['area_mean'].hist(bins=20)
```

Out[25]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x17e5d55f710>
```



Normalizing the values to same scale

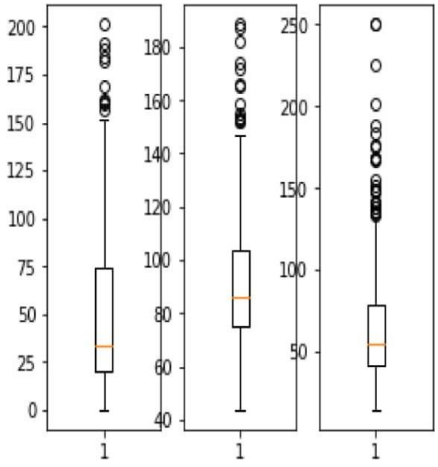
In [26]:

```
df2=df1
df2['area_mean']=df1["area_mean"]/10
df2['concave_points_mean']=df1["concave_points_mean"]*1000
```

Plotting Box plots to know the outliers

In [27]:

```
import matplotlib.pyplot as plt
plt.subplot(141)
plt.boxplot(df2.concave_points_mean)
plt.subplot(142)
plt.boxplot(df2.perimeter_mean)
plt.subplot(143)
plt.boxplot(df2.area_mean)

plt.show()
print("Before removing the outliers",df2.shape)
```




```
Before removing the outliers (569, 10)
```

Removing the outliers for concave_points_mean which is highly correlated to diagnosis

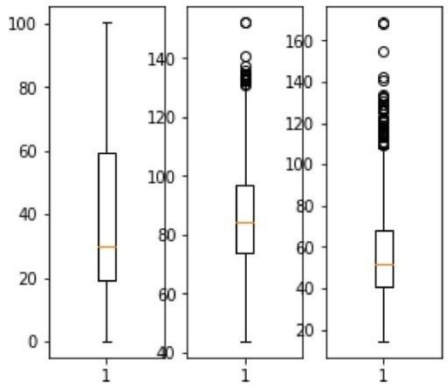
In [28]:

```
d2=df2[df2.concave_points_mean<df2.concave_points_mean.quantile(0.9)]
```

After removing outliers

In [29]:

```
plt.subplot(141)
plt.boxplot(d2.concave_points_mean)
plt.subplot(142)
plt.boxplot(d2.perimeter_mean)
plt.subplot(143)
plt.boxplot(d2.area_mean)

plt.show()
print("After removing outliers",d2.shape)
```




```
After removing outliers (512, 10)
```

spliting train,test data

In [30]:

```
testsize=(int)(d2.shape[0]*0.3)
train=d2[:-testsize]
print(train.shape)
test=d2[-testsize:]
print(test.shape)
```

```
(359, 10)
(153, 10)
```

MODEL FITTING

Logistic Regression:

Logistic Regression is the appropriate regression analysis to conduct when the dependent variable is dichotomous (binary). Like all regression analyses, the logistic regression is a predictive analysis. Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables. Logistic regression predicts the probability of an outcome that can only have two values (i.e. a dichotomy). The prediction is based on the use of one or several predictors (numerical and categorical). In logistic regression, we are only concerned about the probability of outcome dependent variable (success or failure). The goal of logistic regression is to find the best fitting (yet biologically reasonable) model to describe the relationship between the dichotomous characteristic of interest (dependent variable = response or outcome variable) and a set of independent (predictor or explanatory) variables.

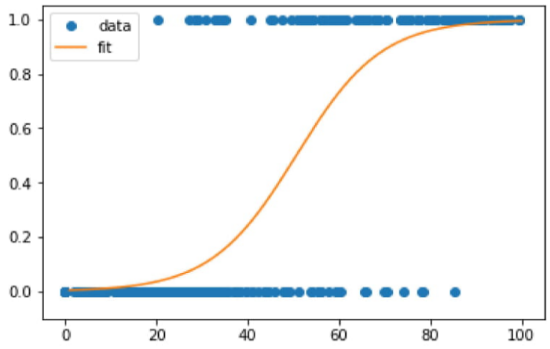
As our project is to predict whether a person has breast cancer or not it comes under the classification problem and logistic regression best fits it.

Plotting a sigmoid

In [17]:

```
import numpy as np
import pylab
from scipy.optimize import curve_fit

def sigmoid(x, x0, k):
     y = 1 / (1 + np.exp(-k*(x-x0)))
     return y

xdata = np.array(train['concave_points_mean'])
ydata = np.array(train['diagnosis'])

popt, pcov = curve_fit(sigmoid,xdata, ydata)
print (popt)

x = np.linspace(1,100, 10000)
y = sigmoid(x, *popt)

pylab.plot(xdata, ydata, 'o', label='data')
pylab.plot(x,y, label='fit')
pylab.ylim(-0.1, 1.05)
pylab.legend(loc='best')
pylab.show()
```

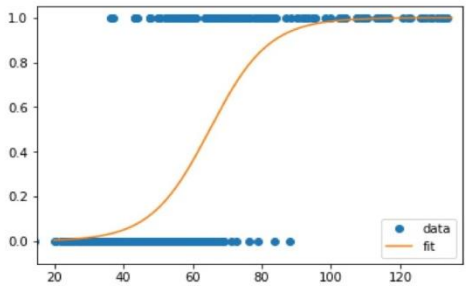
```
[50.63871095  0.10726184]
```



In [18]:

```
xdata = np.array(train['area_mean'])
ydata = np.array(train['diagnosis'])

popt, pcov = curve_fit(sigmoid,xdata, ydata)
print (popt)

x = np.linspace(20,135, 10000)
y = sigmoid(x, *popt)

pylab.plot(xdata, ydata, 'o', label='data')
pylab.plot(x,y, label='fit')
pylab.xlim(15, 138)
pylab.ylim(-0.1, 1.05)
pylab.legend(loc='best')
pylab.show()
```

```
[64.9681297   0.11736101]
```



Fitting model to the dataset

In [19]:

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
x=train[["concave_points_mean",'perimeter_mean','area_mean']]
y=train['diagnosis']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30,random_state=25)
#x_train=x_train[:,np.newaxis]
y_train=y_train[:,np.newaxis]
#x_test=x_test[:,np.newaxis]
y_test=y_test[:,np.newaxis]
lg=LogisticRegression()
lg.fit(x_train,y_train)
y_pred=lg.predict(x_test)
print(y_test.sum())
print(y_pred.sum())
```

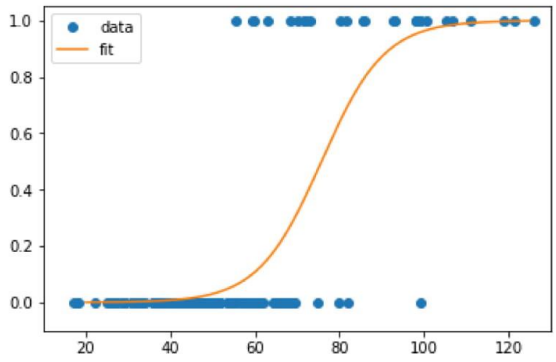
```
45
45

C:\Users\dell\Anaconda3\lib\site-packages\sklearn\utils\validation.py:578: D
ataConversionWarning: A column-vector y was passed when a 1d array was expec
ted. Please change the shape of y to (n_samples, ), for example using ravel
().
  y = column_or_1d(y, warn=True)
```

Sigmoid on test data

In [20]:

```
xdata = np.array(test['area_mean'])
ydata = np.array(test['diagnosis'])

popt, pcov = curve_fit(sigmoid,xdata, ydata)
print (popt)

x = np.linspace(20,125, 10000)
y = sigmoid(x, *popt)

pylab.plot(xdata, ydata, 'o', label='data')
pylab.plot(x,y, label='fit')
pylab.ylim(-0.1, 1.05)
pylab.xlim(10, 130)
pylab.legend(loc='best')
pylab.show()
```

```
[75.75914045  0.13411938]
```



Creating confusion matrix to analyse metrics

In [27]:

```
from sklearn.metrics import confusion_matrix,accuracy_score,classification_report
expected=y_test
pred=y_pred
matrix=confusion_matrix(pred,expected)
print(matrix)

report=classification_report(pred,expected)
print(report)
```

```
[[58  5]
 [ 5 40]]
             precision    recall  f1-score   support

          0       0.92      0.92      0.92        63
          1       0.89      0.89      0.89        45

avg / total       0.91      0.91      0.91       108
```

Fitting and plotting a decision tree based on entropy

In [30]:

```
from sklearn import tree
from sklearn.model_selection import train_test_split
def train_model ():
    clf=tree.DecisionTreeClassifier(criterion='entropy',max_depth=5)
    clf=clf.fit(x_train,y_train)
    return clf
```

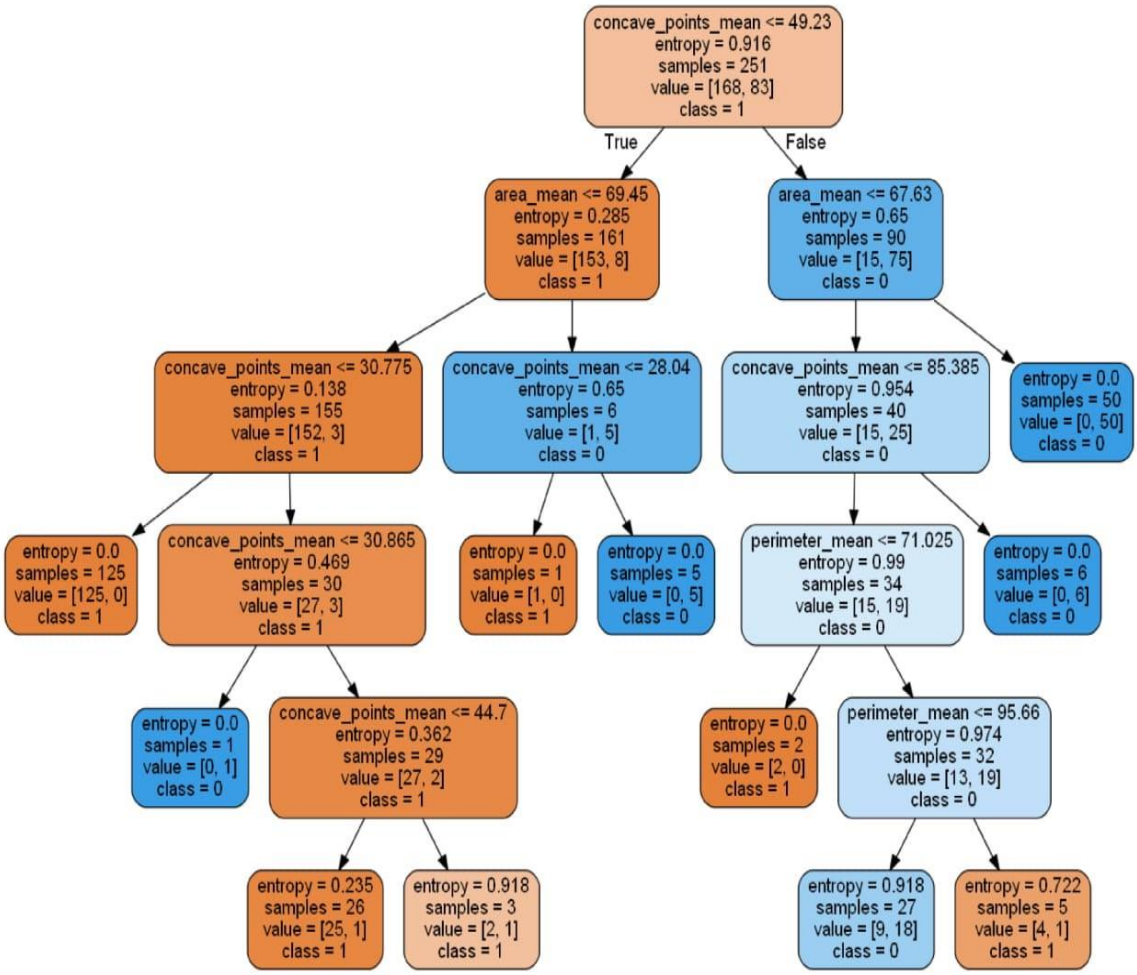
In [31]:

```
from IPython.display import Image,display
import pydotplus
def display_image():
    dot_data=tree.export_graphviz(train_model(),out_file=None,feature_names=['concave_point
    fn=input('create a graph name:')
    ext='.pdf'
    fn=fn+ext
    graph=pydotplus.graph_from_dot_data(dot_data)
    display(Image(data=graph.create_png()))
    graph.write_pdf(fn)
```

In [32]:

```
if __name__=='__main__':

    decision_tree_classifier=train_model()
    display_image()
```

create a graph name:l



RESULTS

By analyzing the data and applying various Machine Learning techniques now we could find whether a person or cancer or not when the concavity, perimeter and area of the tumor is given. These predictions are going to save person's life from cancer and we can reduce the death causes due to cancer.

Creating confusion matrix to analyse metrics

In [25]:

```
from sklearn.metrics import confusion_matrix,accuracy_score,classification_report
expected=y_test
pred=y_pred
matrix=confusion_matrix(pred,expected)
print(matrix)

report=classification_report(pred,expected)
print(report)
```

```
[[58  5]
 [ 5 40]]
             precision    recall  f1-score   support

          0       0.92      0.92      0.92        63
          1       0.89      0.89      0.89        45

avg / total       0.91      0.91      0.91       108
```

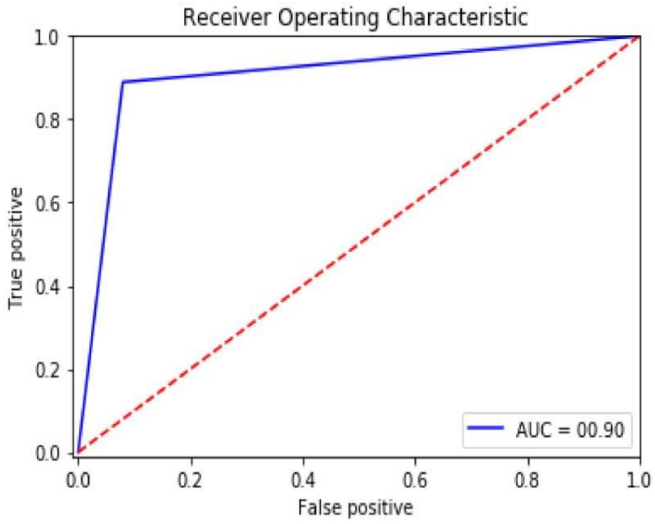
Plotting ROC curve to know the accuracy accurately

In [26]:

```
from sklearn.metrics import roc_curve,auc

false_positive_rate,true_positive_rate,threshold=roc_curve(y_pred,y_test)
roc_auc=auc(false_positive_rate,true_positive_rate)
plt.title('Receiver Operating Characteristic')
plt.plot(false_positive_rate,true_positive_rate,'blue',label='AUC = 0%.2f'%roc_auc)
plt.legend(loc='lower right')
plt.plot([0,1],[0,1],'r--')
plt.xlim([-0.01,1.0])
plt.ylim([-0.01,1.0])
plt.xlabel('False positive')
plt.ylabel('True positive')
plt.show
```

Out[26]:

```
<function matplotlib.pyplot.show(*args, **kw)>
```



AUC(Area Under Curve) of ROC curve is 0.90 which represents a good accuracy

Finding accuracy using sklearn metrics

In [27]:

```
from sklearn.metrics import accuracy_score
a=accuracy_score(y_pred,y_test)
print(a)
#0.9074074074074074
```

```
0.9074074074074074
```

References

[1] Sachin Kumar, Durga Toshniwal, "A data mining approach to characterize road accident locations", J. Mod. Transport. (2016) 24(1):62–72.

[2] S. Shanthi and Dr. R. Geetha Ramani, "Gender Specific Classification of Road Accident Patterns through Data Mining Techniques", IEEE-International Conference on Advances in Engineering, Science and Management (ICAESM -2012) March 30, 31, 2012.

[3] Tessa K. Anderson, "Kernel density estimation and K-means clustering to profile road accident hotspots", Accident Analysis and Prevention 41 (2009) 359–364.