In [8]:
```
import numpy as np
import pandas as pd
import seaborn as sns
```

In [9]:
```
data=pd.read_csv("data1.csv")
```

In [10]:
```
data.shape
```

Out[10]:
```
(569, 32)
```

In [11]:
```
data.head()
```

Out[11]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean
0	842302	M	17.99	10.38	122.80	1001.0	0.11840
1	842517	M	20.57	17.77	132.90	1326.0	0.08474
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960
3	84348301	M	11.42	20.38	77.58	386.1	0.14250
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030

5 rows × 32 columns

In [12]:
```
data.describe()
```

Out[12]:

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean
count	5.690000e+02	569.000000	569.000000	569.000000	569.000000	569.000000
mean	3.037183e+07	14.127292	19.289649	91.969033	654.889104	0.096360
std	1.250206e+08	3.524049	4.301036	24.298981	351.914129	0.014064
min	8.670000e+03	6.981000	9.710000	43.790000	143.500000	0.052630
25%	8.692180e+05	11.700000	16.170000	75.170000	420.300000	0.086370
50%	9.060240e+05	13.370000	18.840000	86.240000	551.100000	0.095870
75%	8.813129e+06	15.780000	21.800000	104.100000	782.700000	0.105300
max	9.113205e+08	28.110000	39.280000	188.500000	2501.000000	0.163400

8 rows × 31 columns

In [13]:
```
data.tail()
```

Out[13]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean
564	926424	M	21.56	22.39	142.00	1479.0	0.11100
565	926682	M	20.13	28.25	131.20	1261.0	0.09780
566	926954	M	16.60	28.08	108.30	858.1	0.08455
567	927241	M	20.60	29.33	140.10	1265.0	0.11780
568	92751	B	7.76	24.54	47.92	181.0	0.05263

5 rows × 32 columns

In [14]:
```
data.sample(10)
```

Out[14]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mea
525	91805	B	8.571	13.10	54.53	221.3	0.103
79	8610908	B	12.860	18.00	83.19	506.3	0.099
336	897604	B	12.990	14.23	84.08	514.3	0.094
462	9113156	B	14.400	26.99	92.25	646.1	0.069
432	908194	M	20.180	19.54	133.80	1250.0	0.113
75	8610404	M	16.070	19.65	104.10	817.7	0.091
125	86561	B	13.850	17.21	88.44	588.7	0.087
235	88249602	B	14.030	21.25	89.79	603.4	0.090
80	861103	B	11.450	20.97	73.81	401.5	0.110
413	905557	B	14.990	22.11	97.53	693.7	0.085

10 rows × 32 columns

In [17]:
```
df=data.copy()
df.head()
```

Out[17]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean
0	842302	M	17.99	10.38	122.80	1001.0	0.11840
1	842517	M	20.57	17.77	132.90	1326.0	0.08474
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960
3	84348301	M	11.42	20.38	77.58	386.1	0.14250
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030

5 rows × 32 columns

In [12]:
```
df.head()
```

Out[12]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean
0	842302	M	17.99	10.38	122.80	1001.0	0.11840
1	842517	M	20.57	17.77	132.90	1326.0	0.08474
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960
3	84348301	M	11.42	20.38	77.58	386.1	0.14250
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030

5 rows × 32 columns

In [13]:
```
df.columns
```

Out[13]:
```
Index(['id', 'diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean',
       'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean',
       'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean',
       'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_se',
       'compactness_se', 'concavity_se', 'concave points_se', 'symmetry_se',
       'fractal_dimension_se', 'radius_worst', 'texture_worst',
       'perimeter_worst', 'area_worst', 'smoothness_worst',
       'compactness_worst', 'concavity_worst', 'concave points_worst',
       'symmetry_worst', 'fractal_dimension_worst'],
      dtype='object')
```

In [14]:
```
df.info
```

Out[14]:
```
<bound method DataFrame.info of            id diagnosis  radius_mean  texture_mean
perimeter_mean  area_mean  \
0      842302         M        17.99         10.38          122.80     1001.0
1      842517         M        20.57         17.77          132.90     1326.0
2    84300903         M        19.69         21.25          130.00     1203.0
3    84348301         M        11.42         20.38           77.58      386.1
4    84358402         M        20.29         14.34          135.10     1297.0
..        ...       ...          ...           ...             ...        ...
564    926424         M        21.56         22.39          142.00     1479.0
565    926682         M        20.13         28.25          131.20     1261.0
566    926954         M        16.60         28.08          108.30      858.1
567    927241         M        20.60         29.33          140.10     1265.0
568     92751         B         7.76         24.54           47.92      181.0

     smoothness_mean  compactness_mean  concavity_mean  concave points_mean  \
0            0.11840           0.27760         0.30010              0.14710
1            0.08474           0.07864         0.08690              0.07017
2            0.10960           0.15990         0.19740              0.12790
3            0.14250           0.28390         0.24140              0.10520
4            0.10030           0.13280         0.19800              0.10430
..               ...               ...             ...                  ...
564          0.11100           0.11590         0.24390              0.13890
565          0.09780           0.10340         0.14400              0.09791
566          0.08455           0.10230         0.09251              0.05302
567          0.11780           0.27700         0.35140              0.15200
568          0.05263           0.04362         0.00000              0.00000

     ...  radius_worst  texture_worst  perimeter_worst  area_worst  \
```

```
0    ...        25.380          17.33           184.60      2019.0
1    ...        24.990          23.41           158.80      1956.0
2    ...        23.570          25.53           152.50      1709.0
3    ...        14.910          26.50            98.87       567.7
4    ...        22.540          16.67           152.20      1575.0
..   ...           ...            ...              ...         ...
564  ...        25.450          26.40           166.10      2027.0
565  ...        23.690          38.25           155.00      1731.0
566  ...        18.980          34.12           126.70      1124.0
567  ...        25.740          39.42           184.60      1821.0
568  ...         9.456          30.37            59.16       268.6

     smoothness_worst  compactness_worst  concavity_worst  \
0             0.16220            0.66560           0.7119
1             0.12380            0.18660           0.2416
2             0.14440            0.42450           0.4504
3             0.20980            0.86630           0.6869
4             0.13740            0.20500           0.4000
..                ...                ...              ...
564           0.14100            0.21130           0.4107
565           0.11660            0.19220           0.3215
566           0.11390            0.30940           0.3403
567           0.16500            0.86810           0.9387
568           0.08996            0.06444           0.0000

     concave points_worst  symmetry_worst  fractal_dimension_worst
0                  0.2654          0.4601                  0.11890
1                  0.1860          0.2750                  0.08902
2                  0.2430          0.3613                  0.08758
3                  0.2575          0.6638                  0.17300
4                  0.1625          0.2364                  0.07678
..                    ...             ...                      ...
564                0.2216          0.2060                  0.07115
565                0.1628          0.2572                  0.06637
566                0.1418          0.2218                  0.07820
567                0.2650          0.4087                  0.12400
568                0.0000          0.2871                  0.07039

[569 rows x 32 columns]>
```

In [18]:

```
from sklearn import preprocessing
le = preprocessing.LabelEncoder()
df.diagnosis=le.fit_transform(df['diagnosis'])
```

In [22]:

```
df.drop(["Unnamed: 32"], axis = 1, inplace=True)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
C:\Users\YASASW~1\AppData\Local\Temp/ipykernel_19676/404118148.py in <module>
----> 1 df.drop(["Unnamed: 32"], axis = 1, inplace=True)

~\anaconda3\lib\site-packages\pandas\util\_decorators.py in wrapper(*args, **kwargs)
    309                     stacklevel=stacklevel,
    310                 )
--> 311             return func(*args, **kwargs)
    312
    313         return wrapper

~\anaconda3\lib\site-packages\pandas\core\frame.py in drop(self, labels, axis, inde
x, columns, level, inplace, errors)
   4904                 weight  1.0     0.8
   4905         """
```

```
-> 4906         return super().drop(
   4907             labels=labels,
   4908             axis=axis,

~\anaconda3\lib\site-packages\pandas\core\generic.py in drop(self, labels, axis, ind
ex, columns, level, inplace, errors)
   4148         for axis, labels in axes.items():
   4149             if labels is not None:
-> 4150                 obj = obj._drop_axis(labels, axis, level=level, errors=error
s)
   4151
   4152         if inplace:

~\anaconda3\lib\site-packages\pandas\core\generic.py in _drop_axis(self, labels, axi
s, level, errors)
   4183                 new_axis = axis.drop(labels, level=level, errors=errors)
   4184             else:
-> 4185                 new_axis = axis.drop(labels, errors=errors)
   4186             result = self.reindex(**{axis_name: new_axis})
   4187

~\anaconda3\lib\site-packages\pandas\core\indexes\base.py in drop(self, labels, erro
rs)
   6015         if mask.any():
   6016             if errors != "ignore":
-> 6017                 raise KeyError(f"{labels[mask]} not found in axis")
   6018             indexer = indexer[~mask]
   6019         return self.delete(indexer)

KeyError: "['Unnamed: 32'] not found in axis"
```

In [18]:

```
df.head()
```

Out[18]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean
0	842302	1	17.99	10.38	122.80	1001.0	0.11840
1	842517	1	20.57	17.77	132.90	1326.0	0.08474
2	84300903	1	19.69	21.25	130.00	1203.0	0.10960
3	84348301	1	11.42	20.38	77.58	386.1	0.14250
4	84358402	1	20.29	14.34	135.10	1297.0	0.10030

5 rows × 32 columns

In [19]:

```
df.isnull().sum()
```

Out[19]:

```
id                         0
diagnosis                  0
radius_mean                0
texture_mean               0
perimeter_mean             0
area_mean                  0
smoothness_mean            0
compactness_mean           0
concavity_mean             0
concave points_mean        0
symmetry_mean              0
```

```
fractal_dimension_mean     0
radius_se                  0
texture_se                 0
perimeter_se               0
area_se                    0
smoothness_se              0
compactness_se             0
concavity_se               0
concave points_se          0
symmetry_se                0
fractal_dimension_se       0
radius_worst               0
texture_worst              0
perimeter_worst            0
area_worst                 0
smoothness_worst           0
compactness_worst          0
concavity_worst            0
concave points_worst       0
symmetry_worst             0
fractal_dimension_worst    0
dtype: int64
```

In [22]:

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier #import decision tree classifier
from sklearn.model_selection import train_test_split #importing train test split fun
from sklearn import metrics # importing scikit - learn module for accuracy calculati
```

In [24]:

```
x=df.drop(["diagnosis"],axis=1)
y=df["diagnosis"]
```

In [26]:

```
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.3,random_state=1
```

In [27]:

```
clf = DecisionTreeClassifier()
clf = clf.fit(x_train,y_train)
y_pred = clf.predict(x_test)
```

In [28]:

```
print("Accuracy :",metrics.accuracy_score(y_test,y_pred))
```

```
Accuracy : 0.9298245614035088
```

In [21]:

```
predictors = df.drop(["diagnosis"],axis=1)
target = df["diagnosis"]
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(predictors,target,test_size=0.25

from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.ensemble import RandomForestClassifier

models = []
models.append(('Logistic Regression',LogisticRegression()))
models.append(('Naive bayes',GaussianNB()))
```

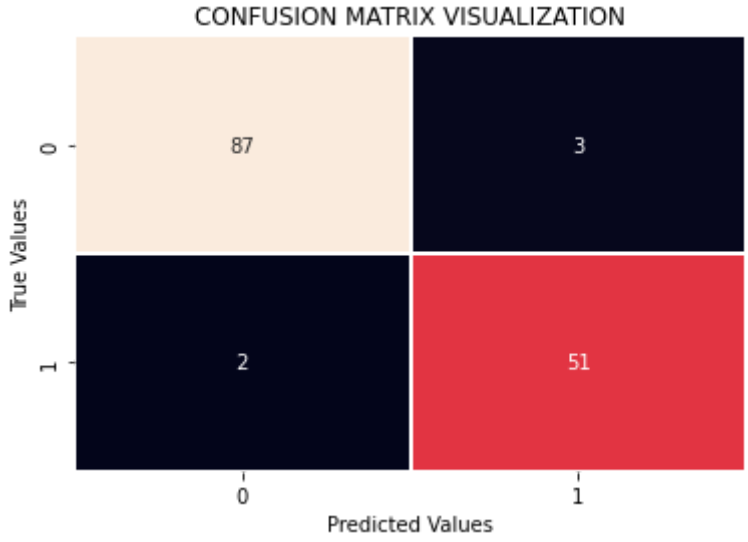
```
models.append(('DecisionTree',DecisionTreeClassifier()))
models.append(('KNeighborsClassifier',KNeighborsClassifier()))
models.append(('SVM',SVC()))
models.append(('RandomForestClassifier',RandomForestClassifier()))

for name, model in models:
    model = model.fit(x_train,y_train)
    y_pred = model.predict(x_test)
    from sklearn import metrics
    print("%s,->accuracy:%%%.2f" %(name,metrics.accuracy_score(y_test,y_pred)*100))
```

```
Logistic Regression,->accuracy:%62.94
Naive bayes,->accuracy:%63.64
DecisionTree,->accuracy:%88.11
KNeighborsClassifier,->accuracy:%75.52
SVM,->accuracy:%62.94
RandomForestClassifier,->accuracy:%96.50
```

In [24]:

```
from sklearn.metrics import confusion_matrix
print(confusion_matrix(y_test, y_pred,labels=[1,0]))
import seaborn as sns
import matplotlib.pyplot as plt
sns.heatmap(confusion_matrix(y_test, y_pred),annot=True,lw =2,cbar=False)
plt.ylabel("True Values")
plt.xlabel("Predicted Values")
plt.title("CONFUSION MATRIX VISUALIZATION")
plt.show()
```

```
[[51  2]
 [ 3 87]]
```



In [25]:

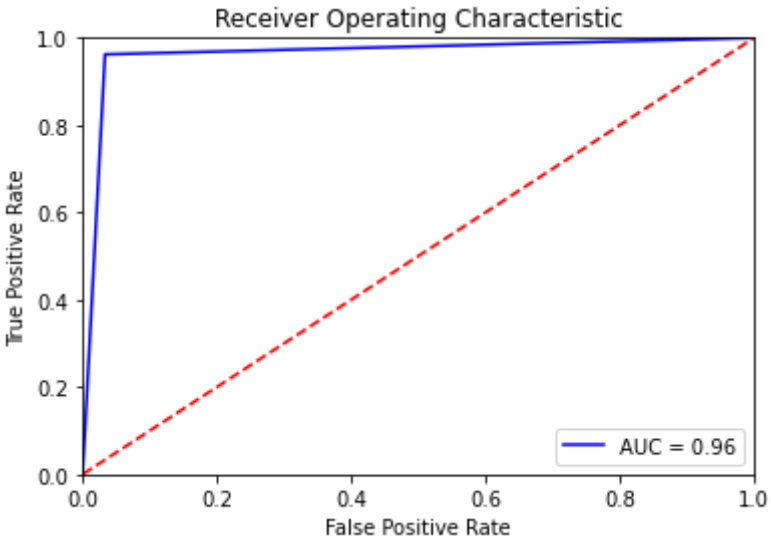
```
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.98      0.97      0.97        90
           1       0.94      0.96      0.95        53

    accuracy                           0.97       143
   macro avg       0.96      0.96      0.96       143
weighted avg       0.97      0.97      0.97       143
```

In [27]:

```
import sklearn.metrics as metrics
```

```
# calculate the fpr and tpr for all thresholds of the classification
probs = model.predict_proba(x_test)
preds = probs[:,1]
fpr, tpr, threshold = metrics.roc_curve(y_test, y_pred)
roc_auc = metrics.auc(fpr, tpr)
# method I: plt
import matplotlib.pyplot as plt
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```



In [28]:
```
from sklearn.metrics import log_loss
log_loss(y_test,y_pred)
```

Out[28]:
```
1.2076662990688398
```

In [ ]:
```
```