1. In this problem we have to predict the Stage of Breast Cancer M (Malignant) and B (Bengin)

2.Attribute Information:

1) ID number

2) Diagnosis (M = malignant, B = benign)

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

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

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

c) perimeter

d) area

e) smoothness (local variation in radius lengths)

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

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

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

i)symmetry

j)fractal dimension ("coastline approximation" - 1)

3.radius, texture,area, perimeter, smoothness,compactness,concavity,concave points,symmetry and fractal dimension these are the parameters which are very useful

In [126]:

```
import numpy as np
import pandas as pd
```

In [127]:

```
data = pd.read_csv(r"C:\Users\kotha\Downloads\cancer.csv",header=0)
```

In [128]:

```
print(data.head(2))
```

```
       id diagnosis  radius_mean  texture_mean  perimeter_mean  area_mean  \
0  842302         M        17.99         10.38           122.8     1001.0   
1  842517         M        20.57         17.77           132.9     1326.0   

   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   

   ...  radius_worst  texture_worst  perimeter_worst  area_worst  \
0  ...         25.38          17.33            184.6      2019.0   
1  ...         24.99          23.41            158.8      1956.0   

   smoothness_worst  compactness_worst  concavity_worst  concave points_wors
t  \
0            0.1622             0.6656           0.7119                0.265
4   
1            0.1238             0.1866           0.2416                0.186
0   

   symmetry_worst  fractal_dimension_worst  
0          0.4601                  0.11890  
1          0.2750                  0.08902  

[2 rows x 32 columns]
```

In [129]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 32 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   id                       569 non-null    int64  
 1   diagnosis                569 non-null    object 
 2   radius_mean              569 non-null    float64
 3   texture_mean             569 non-null    float64
 4   perimeter_mean           569 non-null    float64
 5   area_mean                569 non-null    float64
 6   smoothness_mean          569 non-null    float64
 7   compactness_mean         569 non-null    float64
 8   concavity_mean           569 non-null    float64
 9   concave points_mean      569 non-null    float64
 10  symmetry_mean            569 non-null    float64
 11  fractal_dimension_mean   569 non-null    float64
 12  radius_se                569 non-null    float64
 13  texture_se               569 non-null    float64
 14  perimeter_se             569 non-null    float64
 15  area_se                  569 non-null    float64
 16  smoothness_se            569 non-null    float64
 17  compactness_se           569 non-null    float64
 18  concavity_se             569 non-null    float64
 19  concave points_se        569 non-null    float64
 20  symmetry_se              569 non-null    float64
 21  fractal_dimension_se     569 non-null    float64
 22  radius_worst             569 non-null    float64
 23  texture_worst            569 non-null    float64
 24  perimeter_worst          569 non-null    float64
 25  area_worst               569 non-null    float64
 26  smoothness_worst         569 non-null    float64
 27  compactness_worst        569 non-null    float64
 28  concavity_worst          569 non-null    float64
 29  concave points_worst     569 non-null    float64
 30  symmetry_worst           569 non-null    float64
 31  fractal_dimension_worst  569 non-null    float64
dtypes: float64(30), int64(1), object(1)
memory usage: 142.4+ KB
```

In [130]:

```
data.drop("id",axis=1,inplace=True)
```

In [131]:

```
data.isnull().any()
```

Out[131]:

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

In [132]:

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

In [133]:

```
data.head(5)
```

Out[133]:

	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	

5 rows × 31 columns

In [134]:

```
data.shape
```

Out[134]:

```
(569, 31)
```

In [135]:

```
data.describe()
```

Out[135]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mear
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.00000
mean	0.372583	14.127292	19.289649	91.969033	654.889104	0.09636
std	0.483918	3.524049	4.301036	24.298981	351.914129	0.01406
min	0.000000	6.981000	9.710000	43.790000	143.500000	0.05263
25%	0.000000	11.700000	16.170000	75.170000	420.300000	0.08637
50%	0.000000	13.370000	18.840000	86.240000	551.100000	0.09587
75%	1.000000	15.780000	21.800000	104.100000	782.700000	0.10530
max	1.000000	28.110000	39.280000	188.500000	2501.000000	0.16340

8 rows × 31 columns

In [136]:

```
x=data.iloc[:,1:31].values
y=data.iloc[:,0:1].values
```

In [137]:

```
x.shape
```

Out[137]:

```
(569, 30)
```

In [138]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
```

In [139]:

```
x_train.shape
```

Out[139]:

```
(455, 30)
```

In [140]:

```
x_train
```

Out[140]:

```
array([[1.005e+01, 1.753e+01, 6.441e+01, ..., 6.499e-02, 2.894e-01,
        7.664e-02],
       [1.080e+01, 2.198e+01, 6.879e+01, ..., 7.485e-02, 2.965e-01,
        7.662e-02],
       [1.614e+01, 1.486e+01, 1.043e+02, ..., 1.129e-01, 2.778e-01,
        7.012e-02],
       ...,
       [9.436e+00, 1.832e+01, 5.982e+01, ..., 5.052e-02, 2.454e-01,
        8.136e-02],
       [9.720e+00, 1.822e+01, 6.073e+01, ..., 0.000e+00, 1.909e-01,
        6.559e-02],
       [1.151e+01, 2.393e+01, 7.452e+01, ..., 9.653e-02, 2.112e-01,
        8.732e-02]])
```

In [141]:

```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x_train=sc.fit_transform(x_train)
x_test=sc.fit_transform(x_test)
```

In [142]:

```
x_train
```

Out[142]:

```
array([[-1.15036482, -0.39064196, -1.12855021, ..., -0.75798367,
        -0.01614761, -0.38503402],
       [-0.93798972,  0.68051405, -0.94820146, ..., -0.60687023,
         0.09669004, -0.38615797],
       [ 0.574121  , -1.03333557,  0.51394098, ..., -0.02371948,
        -0.20050207, -0.75144254],
       ...,
       [-1.32422924, -0.20048168, -1.31754581, ..., -0.97974953,
        -0.71542314, -0.11978123],
       [-1.24380987, -0.2245526 , -1.28007609, ..., -1.75401433,
        -1.58157125, -1.00601779],
       [-0.73694129,  1.14989702, -0.71226578, ..., -0.27460457,
        -1.25895095,  0.21515662]])
```

In [143]:

```
from sklearn.linear_model import LogisticRegression
logistic=LogisticRegression()
logistic.fit(x_train,y_train)
```

```
C:\Users\kotha\anaconda3\lib\site-packages\sklearn\utils\validation.py:760:
DataConversionWarning: A column-vector y was passed when a 1d array was expe
cted. Please change the shape of y to (n_samples, ), for example using ravel
().
  y = column_or_1d(y, warn=True)
```

Out[143]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

In [144]:

```
lpred=logistic.predict(x_test)
```

In [145]:

```
lpred
```

Out[145]:

```
array([1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1,
       0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0,
       0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0,
       1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0,
       1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0,
       0, 1, 1, 0])
```

In [146]:

```
y_test
```

Out[146]:

```
array([[1],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [1],
       [0],
       [1],
```

In [147]:

```
from sklearn.metrics import accuracy_score
laccuracy=accuracy_score(y_test,lpred)
```

In [148]:

```
laccuracy
```

Out[148]:

```
0.956140350877193
```

In [149]:

```
from sklearn.metrics import confusion_matrix
lcm=confusion_matrix(y_test,lpred)
```

In [150]:

```
lcm
```

Out[150]:

```
array([[65,  2],
       [ 3, 44]], dtype=int64)
```

In [151]:

```
x_test.shape
```

Out[151]:

```
(114, 30)
```

In [152]:

```
import sklearn.metrics as metrics
lfpr,ltpr,lthreshold=metrics.roc_curve(y_test,lpred)
lroc_auc=metrics.auc(lfpr,ltpr)
```

In [153]:

```
lroc_auc
```

Out[153]:

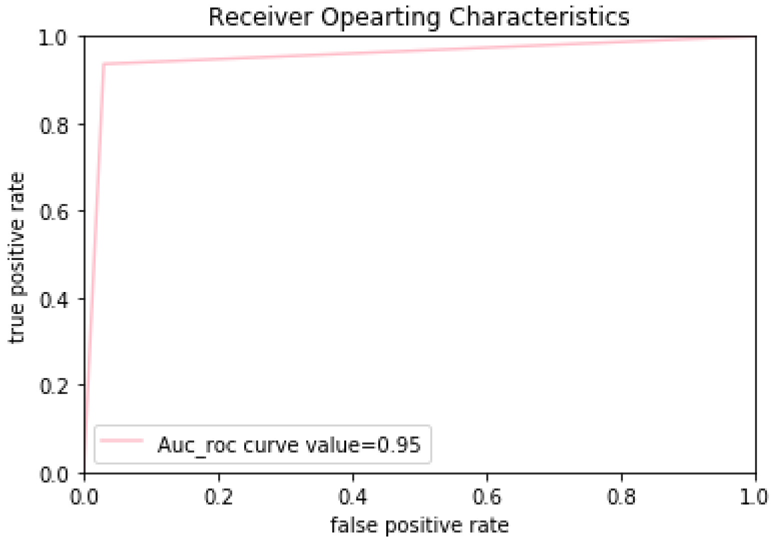
```
0.9531597332486503
```

In [154]:

```
import matplotlib.pyplot as plt
plt.title("Receiver Opearting Characteristics")
plt.plot(lfpr,ltpr,color="pink",label='Auc_roc curve value=%0.2f'%lroc_auc)
plt.legend()
plt.xlim([0,1])
plt.ylim([0,1])
plt.ylabel("true positive rate")
plt.xlabel("false positive rate")
```

Out[154]:

```
Text(0.5, 0, 'false positive rate')
```



In [155]:

```
from sklearn.tree import DecisionTreeClassifier
decisiontree=DecisionTreeClassifier(criterion="entropy",random_state=0)
decisiontree.fit(x_train,y_train)
```

Out[155]:

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='entrop
y',
                       max_depth=None, max_features=None, max_leaf_nodes=Non
e,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=0, splitter='best')
```

In [156]:

```
dpred=decisiontree.predict(x_test)
```

In [157]:

```
dpred
```

Out[157]:

```
array([1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1,
       0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0,
       0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0,
       1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0,
       1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0,
       0, 1, 1, 0])
```

In [158]:

```
y_test
```

Out[158]:

```
array([[1],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [1],
       [0],
       [1],
```

In [159]:

```
daccuracy=accuracy_score(y_test,dpred)
daccuracy
```

Out[159]:

```
0.9385964912280702
```

In [160]:

```
dcm=confusion_matrix(y_test,dpred)
```

In [161]:

```
dcm
```

Out[161]:

```
array([[64,  3],
       [ 4, 43]], dtype=int64)
```

In [210]:

```
import sklearn.metrics as metrics
dfpr,dtpr,dthreshold=metrics.roc_curve(y_test,dpred)
droc_auc=metrics.auc(dfpr,dtpr)
```

In [211]:

```
droc_auc
```

Out[211]:

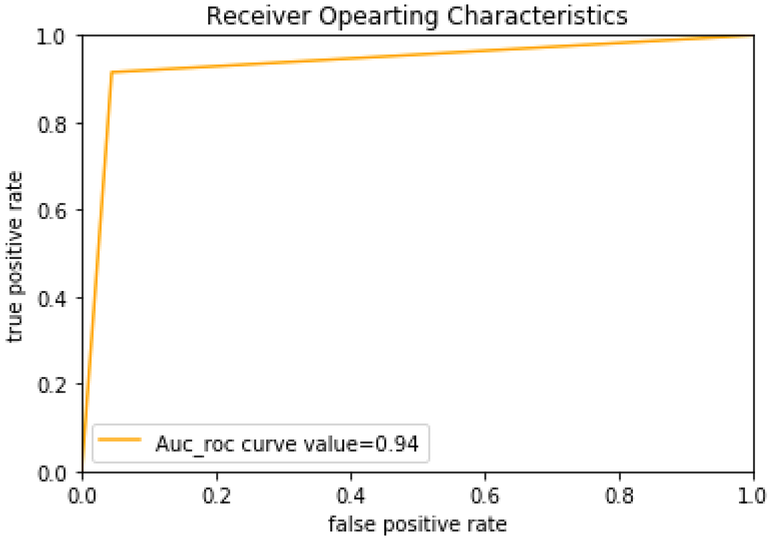
```
0.9350587488091456
```

In [163]:

```
import matplotlib.pyplot as plt
plt.title("Receiver Opearting Characteristics")
plt.plot(dfpr,dtpr,color="orange",label='Auc_roc curve value=%0.2f'%droc_auc)
plt.legend()
plt.xlim([0,1])
plt.ylim([0,1])
plt.ylabel("true positive rate")
plt.xlabel("false positive rate")
```

Out[163]:

```
Text(0.5, 0, 'false positive rate')
```



In [164]:

```
from sklearn.ensemble import RandomForestClassifier
random=RandomForestClassifier(n_estimators=10,criterion="entropy",random_state=0)
random.fit(x_train,y_train)
```

```
C:\Users\kotha\anaconda3\lib\site-packages\ipykernel_launcher.py:3: DataConv
ersionWarning: A column-vector y was passed when a 1d array was expected. Pl
ease change the shape of y to (n_samples,), for example using ravel().
  This is separate from the ipykernel package so we can avoid doing imports
until
```

Out[164]:

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='entropy', max_depth=None, max_features='au
to',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=10,
                       n_jobs=None, oob_score=False, random_state=0, verbose
=0,
                       warm_start=False)
```

In [165]:

```
rpred=random.predict(x_test)
```

In [166]:

```
rpred
```

Out[166]:

```
array([1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1,
       0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0,
       0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0,
       1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0,
       1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1,
       0, 1, 1, 0])
```

In [167]:

```
y_test
```

Out[167]:

```
array([[1],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [1],
       [0],
       [1],
```

In [168]:

```
raccuracy=accuracy_score(y_test,rpred)
raccuracy
```

Out[168]:

```
0.9736842105263158
```

In [169]:

```
rcm=confusion_matrix(y_test,rpred)
```

In [170]:

```
rcm
```

Out[170]:

```
array([[66,  1],
       [ 2, 45]], dtype=int64)
```

In [171]:

```
import sklearn.metrics as metrics
rfpr,rtpr,rthreshold=metrics.roc_curve(y_test,rpred)
rroc_auc=metrics.auc(rfpr,rtpr)
```

In [172]:

```
rroc_auc
```

Out[172]:

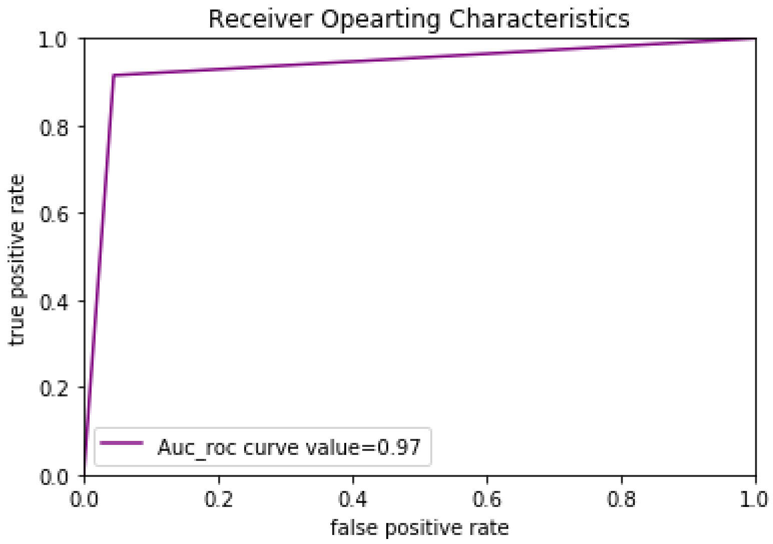
```
0.971260717688155
```

In [212]:

```
import matplotlib.pyplot as plt
plt.title("Receiver Opearting Characteristics")
plt.plot(dfpr,dtpr,color="purple",label='Auc_roc curve value=%0.2f'%rroc_auc)
plt.legend()
plt.xlim([0,1])
plt.ylim([0,1])
plt.ylabel("true positive rate")
plt.xlabel("false positive rate")
```

Out[212]:

```
Text(0.5, 0, 'false positive rate')
```



```
from sklearn.neighbors import KNeighborsClassifier
knn= KNeighborsClassifier(n_neighbors=5,metric="minkowski",p=2)
knn.fit(x_train,y_train)
```

In [175]:

```
kpred=knn.predict(x_test)
```

In [176]:

```
kpred
```

Out[176]:

```
array([1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1,
       0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0,
       0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0,
       1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0,
       1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0,
       0, 1, 1, 0])
```

In [177]:

```
y_test
```

Out[177]:

```
array([[1],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [1],
       [0],
       [1],
```

In [178]:

```
from sklearn.metrics import accuracy_score
kaccuracy=accuracy_score(y_test,kpred)
```

In [179]:

```
kaccuracy
```

Out[179]:

```
0.9649122807017544
```

In [180]:

```
from sklearn.metrics import confusion_matrix
kcm=confusion_matrix(y_test,kpred)
```

In [181]:

```
kcm
```

Out[181]:

```
array([[67,  0],
       [ 4, 43]], dtype=int64)
```

In [182]:

```
import sklearn.metrics as metrics
kfpr,ktpr,kthreshold=metrics.roc_curve(y_test,kpred)
kroc_auc=metrics.auc(kfpr,ktpr)
```

In [183]:

```
kroc_auc
```

Out[183]:

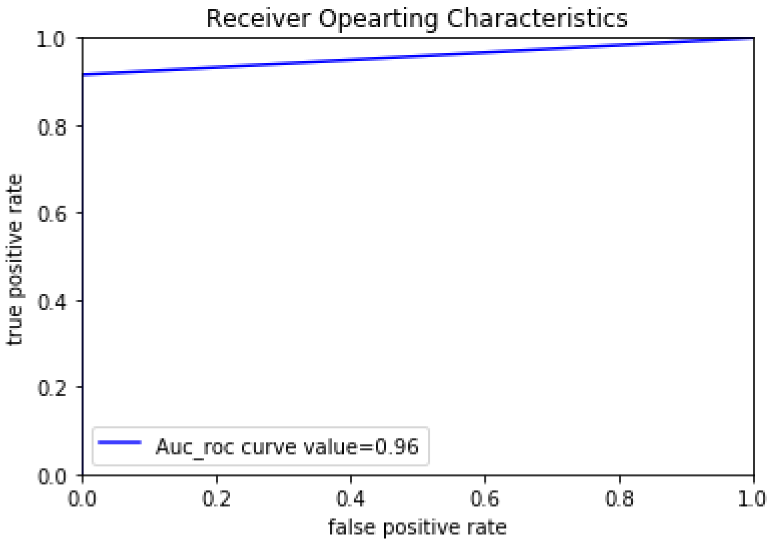
```
0.9574468085106382
```

In [184]:

```
import matplotlib.pyplot as plt
plt.title("Receiver Opearting Characteristics")
plt.plot(kfpr,ktpr,color="blue",label='Auc_roc curve value=%0.2f'%kroc_auc)
plt.legend()
plt.xlim([0,1])
plt.ylim([0,1])
plt.ylabel("true positive rate")
plt.xlabel("false positive rate")
```

Out[184]:

```
Text(0.5, 0, 'false positive rate')
```



In [185]:

```
ktpr
```

Out[185]:

```
array([0.        , 0.91489362, 1.        ])
```

In [186]:

```
kfpr
```

Out[186]:

```
array([0., 0., 1.])
```

In [187]:

```
kthreshold
```

Out[187]:

```
array([2, 1, 0])
```

In [188]:

```
from sklearn.naive_bayes import GaussianNB
naive=GaussianNB()
naive.fit(x_train,y_train)
```

```
C:\Users\kotha\anaconda3\lib\site-packages\sklearn\naive_bayes.py:206: DataC
onversionWarning: A column-vector y was passed when a 1d array was expected.
Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Out[188]:

```
GaussianNB(priors=None, var_smoothing=1e-09)
```

In [189]:

```
npred=naive.predict(x_test)
```

In [190]:

```
npred
```

Out[190]:

```
array([1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1,
       0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0,
       0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0,
       1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0,
       1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1,
       0, 1, 1, 0])
```

In [191]:

```
y_test
```

Out[191]:

```
array([[1],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [1],
       [0],
       [1],
```

In [192]:

```
naccuracy=accuracy_score(y_test,npred)
```

In [193]:

```
naccuracy
```

Out[193]:

```
0.9385964912280702
```

In [194]:

```
ncm=confusion_matrix(y_test,npred)
```

In [195]:

```
ncm
```

Out[195]:

```
array([[64,  3],
       [ 4, 43]], dtype=int64)
```

In [196]:

```
import sklearn.metrics as metrics
nfpr,ntpr,nthreshold=metrics.roc_curve(y_test,npred)
nroc_auc=metrics.auc(nfpr,ntpr)
```

In [197]:

```
nroc_auc
```

Out[197]:

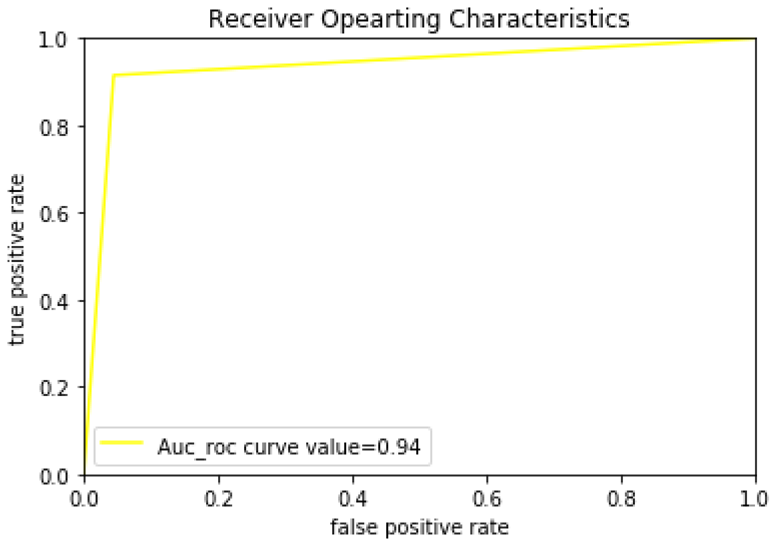
```
0.9350587488091456
```

In [198]:

```
import matplotlib.pyplot as plt
plt.title("Receiver Opearting Characteristics")
plt.plot(nfpr,ntpr,color="yellow",label='Auc_roc curve value=%0.2f'%nroc_auc)
plt.legend()
plt.xlim([0,1])
plt.ylim([0,1])
plt.ylabel("true positive rate")
plt.xlabel("false positive rate")
```

Out[198]:

```
Text(0.5, 0, 'false positive rate')
```



In [199]:

```
from sklearn.svm import SVC
svm=SVC(kernel="linear")
svm.fit(x_train,y_train)
```

```
C:\Users\kotha\anaconda3\lib\site-packages\sklearn\utils\validation.py:760:
DataConversionWarning: A column-vector y was passed when a 1d array was expe
cted. Please change the shape of y to (n_samples, ), for example using ravel
().
  y = column_or_1d(y, warn=True)
```

Out[199]:

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='linear',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

In [200]:

```
spred=svm.predict(x_test)
```

In [201]:

```
spred
```

Out[201]:

```
array([1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1,
       0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0,
       0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0,
       1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0,
       1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0,
       0, 1, 1, 0])
```

In [202]:

```
y_test
```

Out[202]:

```
array([[1],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [1],
       [0],
       [1],
```

In [203]:

```
saccuracy=accuracy_score(y_test,spred)
```

In [204]:

```
saccuracy
```

Out[204]:

```
0.9736842105263158
```

In [205]:

```
scm=confusion_matrix(y_test,spred)
```

In [206]:

```
scm
```

Out[206]:

```
array([[66,  1],
       [ 2, 45]], dtype=int64)
```

In [207]:

```
import sklearn.metrics as metrics
sfpr,stpr,sthreshold=metrics.roc_curve(y_test,spred)
sroc_auc=metrics.auc(sfpr,stpr)
```

In [208]:

```
sroc_auc
```

Out[208]:

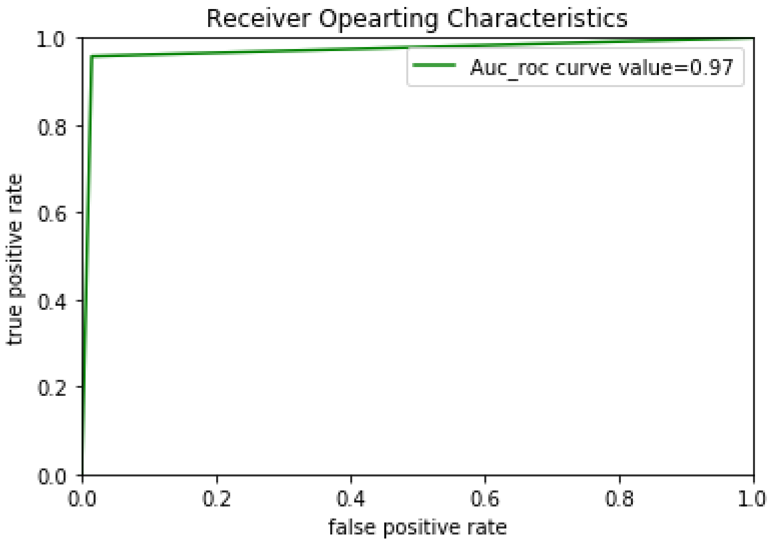
```
0.971260717688155
```

In [209]:

```
import matplotlib.pyplot as plt
plt.title("Receiver Opearting Characteristics")
plt.plot(sfpr,stpr,color="green",label='Auc_roc curve value=%0.2f'%sroc_auc)
plt.legend()
plt.xlim([0,1])
plt.ylim([0,1])
plt.ylabel("true positive rate")
plt.xlabel("false positive rate")
```

Out[209]:

```
Text(0.5, 0, 'false positive rate')
```



```
Observation:
LOGISTIC:                  DECISIONTREE:              RANDOMFOREST:        KNN:
          NAIVE_BAYES:
laccuracy=0.95             daccuracy=0.938            raccuracy=0.97       kaccuracy=0.96
          naccuracy=0.93
```

```
Auc_roc curve value=0.95   Auc_roc curve value=0.94   Auc_roc value=0.97  Auc_roc curve
value=0.96 Auc_roc curve value=0.94

SVM:
saccuracy=0.97
Auc_roc curve value=0.97
```

In [ ]: