In [1]:

```
import pandas as pd
from sklearn import datasets
```

In [2]:

```
cancer_data=datasets.load_breast_cancer()
cancer_data
```

Out[2]:

```
{'data': array([[1.799e+01, 1.038e+01, 1.228e+02, ..., 2.654e-01, 4.601e-01,
          1.189e-01],
         [2.057e+01, 1.777e+01, 1.329e+02, ..., 1.860e-01, 2.750e-01,
          8.902e-02],
         [1.969e+01, 2.125e+01, 1.300e+02, ..., 2.430e-01, 3.613e-01,
          8.758e-02],
         ...,
         [1.660e+01, 2.808e+01, 1.083e+02, ..., 1.418e-01, 2.218e-01,
          7.820e-02],
         [2.060e+01, 2.933e+01, 1.401e+02, ..., 2.650e-01, 4.087e-01,
          1.240e-01],
         [7.760e+00, 2.454e+01, 4.792e+01, ..., 0.000e+00, 2.871e-01,
          7.039e-02]]),
 'target': array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
1, 1, 1,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
        0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0,
        1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0,
        1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1,
        1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0,
        0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1,
        1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0,
        0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0,
        1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1,
        1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0,
        0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0,
        0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0,
        1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1,
        1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1,
        1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0,
        1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1,
        1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1,
        1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1]),
 'frame': None,
 'target_names': array(['malignant', 'benign'], dtype='<U9'),
 'DESCR': '.. _breast_cancer_dataset:\n\nBreast cancer wisconsin (diagnosti
c) dataset\n--------------------------------------------\n\n**Data Set Chara
cteristics:**\n\n    :Number of Instances: 569\n\n    :Number of Attributes:
30 numeric, predictive attributes and the class\n\n    :Attribute Informatio
n:\n        - radius (mean of distances from center to points on the perimet
er)\n        - texture (standard deviation of gray-scale values)\n        -
perimeter\n        - area\n        - smoothness (local variation in radius l
engths)\n        - compactness (perimeter^2 / area - 1.0)\n        - concavi
ty (severity of concave portions of the contour)\n        - concave points
(number of concave portions of the contour)\n        - symmetry\n        - f
ractal dimension ("coastline approximation" - 1)\n\n        The mean, standa
rd error, and "worst" or largest (mean of the three\n        worst/largest v
```

```
alues) of these features were computed for each image,\n        resulting in
30 features.  For instance, field 0 is Mean Radius, field\n        10 is Rad
ius SE, field 20 is Worst Radius.\n\n        - class:\n                - WDB
C-Malignant\n                - WDBC-Benign\n\n    :Summary Statistics:\n\n
===================================== ====== ======\n
Min    Max\n    ===================================== ====== ======\n    rad
ius (mean):                        6.981  28.11\n    texture (mean):
9.71   39.28\n    perimeter (mean):                     43.79  188.5\n    ar
ea (mean):                          143.5  2501.0\n    smoothness (mean):
0.053  0.163\n    compactness (mean):                   0.019  0.345\n    co
ncavity (mean):                     0.0    0.427\n    concave points (mean):
0.0    0.201\n    symmetry (mean):                      0.106  0.304\n    fr
actal dimension (mean):             0.05   0.097\n    radius (standard erro
r):              0.112  2.873\n    texture (standard error):            0.3
6   4.885\n    perimeter (standard error):          0.757  21.98\n    area
(standard error):                6.802  542.2\n    smoothness (standard erro
r):          0.002  0.031\n    compactness (standard error):         0.002
0.135\n    concavity (standard error):           0.0    0.396\n    concave p
oints (standard error):      0.0    0.053\n    symmetry (standard error):
0.008  0.079\n    fractal dimension (standard error):   0.001  0.03\n    rad
ius (worst):                        7.93   36.04\n    texture (worst):
12.02  49.54\n    perimeter (worst):                    50.41  251.2\n    ar
ea (worst):                         185.2  4254.0\n    smoothness (worst):
0.071  0.223\n    compactness (worst):                  0.027  1.058\n    co
ncavity (worst):                    0.0    1.252\n    concave points (wors
t):              0.0    0.291\n    symmetry (worst):                     0.
156  0.664\n    fractal dimension (worst):            0.055  0.208\n    ====
================================= ====== ======\n\n    :Missing Attribute Va
lues: None\n\n    :Class Distribution: 212 - Malignant, 357 - Benign\n\n
:Creator:  Dr. William H. Wolberg, W. Nick Street, Olvi L. Mangasarian\n\n
:Donor: Nick Street\n\n    :Date: November, 1995\n\nThis is a copy of UCI ML
Breast Cancer Wisconsin (Diagnostic) datasets.\nhttps://goo.gl/U2Uwz2\n\nFea
tures are computed from a digitized image of a fine needle\naspirate (FNA) o
f a breast mass.  They describe\ncharacteristics of the cell nuclei present
in the image.\n\nSeparating plane described above was obtained using\nMultis
urface Method-Tree (MSM-T) [K. P. Bennett, "Decision Tree\nConstruction Via
Linear Programming." Proceedings of the 4th\nMidwest Artificial Intelligence
and Cognitive Science Society,\npp. 97-101, 1992], a classification method w
hich uses linear\nprogramming to construct a decision tree.  Relevant featur
es\nwere selected using an exhaustive search in the space of 1-4\nfeatures a
nd 1-3 separating planes.\n\nThe actual linear program used to obtain the se
parating plane\nin the 3-dimensional space is that described in:\n[K. P. Ben
nett and O. L. Mangasarian: "Robust Linear\nProgramming Discrimination of Tw
o Linearly Inseparable Sets",\nOptimization Methods and Software 1, 1992, 23
-34].\n\nThis database is also available through the UW CS ftp server:\n\nft
p ftp.cs.wisc.edu\ncd math-prog/cpo-dataset/machine-learn/WDBC/\n\n.. topi
c:: References\n\n   - W.N. Street, W.H. Wolberg and O.L. Mangasarian. Nucle
ar feature extraction \n     for breast tumor diagnosis. IS&T/SPIE 1993 Inte
rnational Symposium on \n     Electronic Imaging: Science and Technology, vo
lume 1905, pages 861-870,\n     San Jose, CA, 1993.\n   - O.L. Mangasarian,
W.N. Street and W.H. Wolberg. Breast cancer diagnosis and \n     prognosis v
ia linear programming. Operations Research, 43(4), pages 570-577, \n     Jul
y-August 1995.\n   - W.H. Wolberg, W.N. Street, and O.L. Mangasarian. Machin
e learning techniques\n     to diagnose breast cancer from fine-needle aspir
ates. Cancer Letters 77 (1994) \n     163-171.',
 'feature_names': array(['mean radius', 'mean texture', 'mean perimeter', 'm
ean area',
        'mean smoothness', 'mean compactness', 'mean concavity',
        'mean concave points', 'mean symmetry', 'mean fractal dimension',
        'radius error', 'texture error', 'perimeter error', 'area error',
        'smoothness error', 'compactness error', 'concavity error',
```

```
        'concave points error', 'symmetry error',
        'fractal dimension error', 'worst radius', 'worst texture',
        'worst perimeter', 'worst area', 'worst smoothness',
        'worst compactness', 'worst concavity', 'worst concave points',
        'worst symmetry', 'worst fractal dimension'], dtype='<U23'),
 'filename': 'C:\\Users\\ROOBA\\anaconda3\\lib\\site-packages\\sklearn\\data
sets\\data\\breast_cancer.csv'}
```

In [3]:

```
x=cancer_data.data
y=cancer_data.target
breast_cancer=pd.DataFrame(x,columns=cancer_data.feature_names)
breast_cancer['Target']=y
breast_cancer.head()
```

Out[3]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mear symmetry
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.241
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.181
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.206
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.259
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.180

5 rows × 31 columns

In [4]:

```
breast_cancer.shape
```

Out[4]:

```
(569, 31)
```

In [5]:

```
breast_cancer.dtypes
```

Out[5]:

```
mean radius                float64
mean texture               float64
mean perimeter             float64
mean area                  float64
mean smoothness            float64
mean compactness           float64
mean concavity             float64
mean concave points        float64
mean symmetry              float64
mean fractal dimension     float64
radius error               float64
texture error              float64
perimeter error            float64
area error                 float64
smoothness error           float64
compactness error          float64
concavity error            float64
concave points error       float64
symmetry error             float64
fractal dimension error    float64
worst radius               float64
worst texture              float64
worst perimeter            float64
worst area                 float64
worst smoothness           float64
worst compactness          float64
worst concavity            float64
worst concave points       float64
worst symmetry             float64
worst fractal dimension    float64
Target                       int32
dtype: object
```

In [6]:

```
breast_cancer.isnull().sum()
```

Out[6]:

```
mean radius                0
mean texture               0
mean perimeter             0
mean area                  0
mean smoothness            0
mean compactness           0
mean concavity             0
mean concave points        0
mean symmetry              0
mean fractal dimension     0
radius error               0
texture error              0
perimeter error            0
area error                 0
smoothness error           0
compactness error          0
concavity error            0
concave points error       0
symmetry error             0
fractal dimension error    0
worst radius               0
worst texture              0
worst perimeter            0
worst area                 0
worst smoothness           0
worst compactness          0
worst concavity            0
worst concave points       0
worst symmetry             0
worst fractal dimension    0
Target                     0
dtype: int64
```

In [7]:

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

In [8]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=123)
```

In [9]:

```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(455, 30)
(114, 30)
(455,)
(114,)
```

In [10]:

```
y_test.shape
```

Out[10]:

```
(114,)
```

Adaboostclassifier

In [11]:

```
from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
classifier=AdaBoostClassifier()
```

In [12]:

```
classifier.fit(x_train,y_train)
```

Out[12]:

```
AdaBoostClassifier()
```

In [13]:

```
y_pred_train=classifier.predict(x_train)
y_pred_test=classifier.predict(x_test)
```

In [68]:

```
from sklearn.metrics import confusion_matrix,classification_report,roc_auc_score,roc_curve
```

In [15]:

```
print(confusion_matrix(y_train,y_pred_train))
```

```
[[171   0]
 [  0 284]]
```

In [16]:

```
confusion_matrix(y_test,y_pred_test)
```

Out[16]:

```
array([[39,  2],
       [ 1, 72]], dtype=int64)
```

In [17]:

```
print(classification_report(y_train,y_pred_train))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       171
           1       1.00      1.00      1.00       284

    accuracy                           1.00       455
   macro avg       1.00      1.00      1.00       455
weighted avg       1.00      1.00      1.00       455
```

In [18]:

```
print(classification_report(y_test,y_pred_test))
```

```
              precision    recall  f1-score   support

           0       0.97      0.95      0.96        41
           1       0.97      0.99      0.98        73

    accuracy                           0.97       114
   macro avg       0.97      0.97      0.97       114
weighted avg       0.97      0.97      0.97       114
```

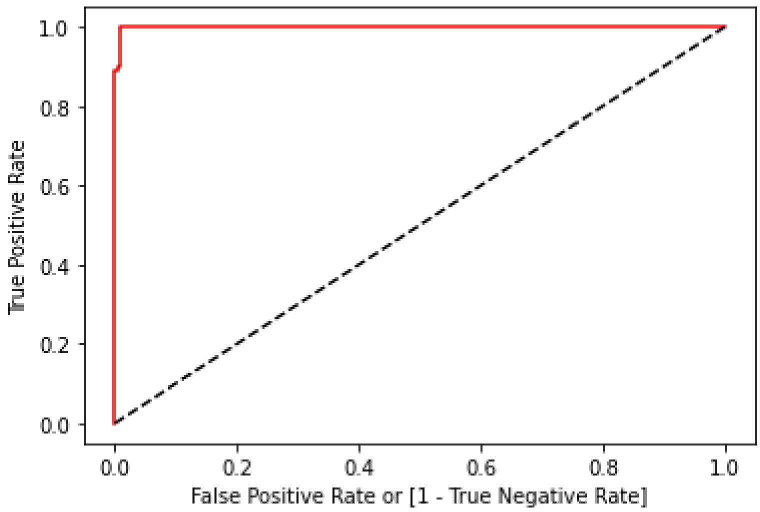
In [73]:

```
fpr, tpr, thresholds = roc_curve(y, classifier.predict_proba (x)[:,1])

auc_train = roc_auc_score(y_train, y_pred_train)
print('Auc value for train data: ',auc_train)
auc_test= roc_auc_score(y_test, y_pred_test)
print('Auc value for test data: ',auc_test)

import matplotlib.pyplot as plt
plt.plot(fpr, tpr, color='red')
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False Positive Rate or [1 - True Negative Rate]')
plt.ylabel('True Positive Rate')
```

```
Auc value for train data:  1.0
Auc value for test data:  0.9687604410290678
```

Out[73]:

```
Text(0, 0.5, 'True Positive Rate')
```



Inferences

1.Accuracy of train data=100% test data = 97%

2.Tradeoff between precision anf recall is balanced

3.FN and FP for test values are 2 and 1 respectively

4.Separability between positive and negative value is train data=100%,test data=96.87%

Decision tree

In [19]:

```
classifier_decisiontree=DecisionTreeClassifier( max_depth=5,random_state=123)
```

In [20]:

```
classifier_decisiontree.fit(x_train,y_train)
```

Out[20]:

```
DecisionTreeClassifier(max_depth=5, random_state=123)
```

In [21]:

```
y_pred_decisiontree_train=classifier_decisiontree.predict(x_train)
y_pred_decisiontree_test=classifier_decisiontree.predict(x_test)
```

In [22]:

```
confusion_matrix(y_train,y_pred_decisiontree_train)
```

Out[22]:

```
array([[169,   2],
       [  0, 284]], dtype=int64)
```

In [23]:

```
confusion_matrix(y_test,y_pred_decisiontree_test)
```

Out[23]:

```
array([[39,  2],
       [ 2, 71]], dtype=int64)
```

In [24]:

```
print(classification_report(y_train,y_pred_decisiontree_train))
```

```
              precision    recall  f1-score   support

           0       1.00      0.99      0.99       171
           1       0.99      1.00      1.00       284

    accuracy                           1.00       455
   macro avg       1.00      0.99      1.00       455
weighted avg       1.00      1.00      1.00       455
```

In [25]:

```
print(classification_report(y_test,y_pred_decisiontree_test))
```

```
              precision    recall  f1-score   support

           0       0.95      0.95      0.95        41
           1       0.97      0.97      0.97        73

    accuracy                           0.96       114
   macro avg       0.96      0.96      0.96       114
weighted avg       0.96      0.96      0.96       114
```

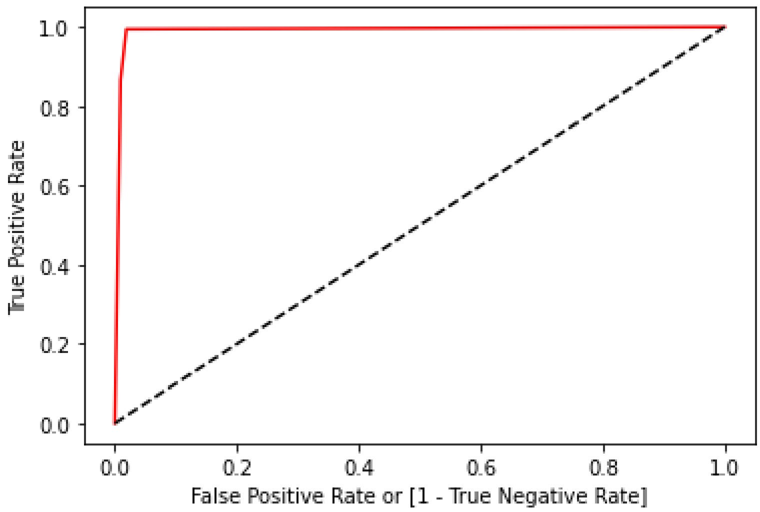
In [74]:

```
fpr, tpr, thresholds = roc_curve(y, classifier_decisiontree.predict_proba (x)[:,1])

auc_train = roc_auc_score(y_train, y_pred_decisiontree_train)
print('Auc value for train data: ',auc_train)
auc_test= roc_auc_score(y_test, y_pred_decisiontree_test)
print('Auc value for test data: ',auc_test)

import matplotlib.pyplot as plt
plt.plot(fpr, tpr, color='red')
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False Positive Rate or [1 - True Negative Rate]')
plt.ylabel('True Positive Rate')
```

```
Auc value for train data:  0.9941520467836258
Auc value for test data:  0.9619111259605746
```

Out[74]:

```
Text(0, 0.5, 'True Positive Rate')
```



GridsearchCv --- Decision tree

In [41]:

```
from sklearn.model_selection import GridSearchCV
grid_model_decisiontree = GridSearchCV(estimator = classifier_decisiontree,param_grid = {'c
                                                         'max_depth' :[3,5,7,8,10],
                                                         'min_samples_split' :[2,3,4]})
grid_model_decisiontree.fit(x_train,y_train)
print(grid_model_decisiontree.best_params_)
print(grid_model_decisiontree.best_score_)
```

```
{'criterion': 'entropy', 'max_depth': 7, 'min_samples_split': 2}
0.9428571428571428
```

In [42]:

```
classifier_decisiontree1=DecisionTreeClassifier( max_depth=7,criterion='entropy')
```

In [43]:

```
classifier_decisiontree1.fit(x_train,y_train)
```

Out[43]:

```
DecisionTreeClassifier(criterion='entropy', max_depth=7)
```

In [44]:

```
y_pred_decisiontree_train1=classifier_decisiontree1.predict(x_train)
y_pred_decisiontree_test1=classifier_decisiontree1.predict(x_test)
```

In [45]:

```
confusion_matrix(y_train,y_pred_decisiontree_train1)
```

Out[45]:

```
array([[171,   0],
       [  0, 284]], dtype=int64)
```

In [46]:

```
confusion_matrix(y_test,y_pred_decisiontree_test1)
```

Out[46]:

```
array([[39,  2],
       [ 1, 72]], dtype=int64)
```

In [47]:

```
print(classification_report(y_train,y_pred_decisiontree_train1))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       171
           1       1.00      1.00      1.00       284

    accuracy                           1.00       455
   macro avg       1.00      1.00      1.00       455
weighted avg       1.00      1.00      1.00       455
```

In [48]:

```
print(classification_report(y_test,y_pred_decisiontree_test1))
```

```
              precision    recall  f1-score   support

           0       0.97      0.95      0.96        41
           1       0.97      0.99      0.98        73

    accuracy                           0.97       114
   macro avg       0.97      0.97      0.97       114
weighted avg       0.97      0.97      0.97       114
```

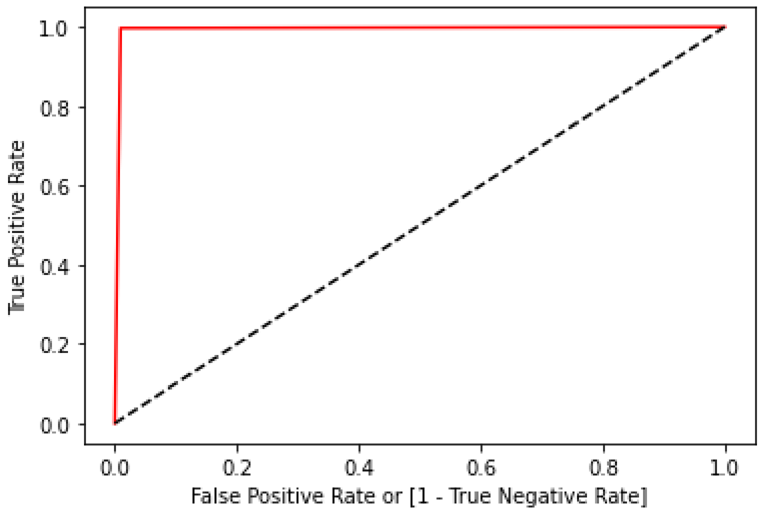
In [75]:

```
fpr, tpr, thresholds = roc_curve(y, classifier_decisiontree1.predict_proba (x)[:,1])

auc_train = roc_auc_score(y_train, y_pred_decisiontree_train1)
print('Auc value for train data: ',auc_train)
auc_test= roc_auc_score(y_test, y_pred_decisiontree_test1)
print('Auc value for test data: ',auc_test)

import matplotlib.pyplot as plt
plt.plot(fpr, tpr, color='red')
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False Positive Rate or [1 - True Negative Rate]')
plt.ylabel('True Positive Rate')
```

```
Auc value for train data:  1.0
Auc value for test data:  0.9687604410290678
```

Out[75]:

```
Text(0, 0.5, 'True Positive Rate')
```



Inferences

Before GridsearchCV

1.Accuracy of train data=100% test data = 96%

2.Tradeoff between precision anf recall is balanced

3.FN and FP for test values are 2 and 2 respectively

4.Separability between positive and negative value is train data=99.4%,test data=96.19%

After GridsearchCV

1.Accuracy of train data=100% test data = 97%

2.Tradeoff between precision anf recall is balanced

3.FN and FP for test values are 2 and 1 respectively

4.Separability between positive and negative value is train data=100%,test data=96.87%

Random forest

In [58]:

```
from sklearn.ensemble import RandomForestClassifier
classifier_randomforest=RandomForestClassifier()
```

In [59]:

```
classifier_randomforest.fit(x_train,y_train)
```

Out[59]:

```
RandomForestClassifier()
```

In [60]:

```
y_pred_rf_train=classifier_randomforest.predict(x_train)
y_pred_rf_test=classifier_randomforest.predict(x_test)
```

In [61]:

```
confusion_matrix(y_train,y_pred_rf_train)
```

Out[61]:

```
array([[171,   0],
       [  0, 284]], dtype=int64)
```

In [62]:

```
confusion_matrix(y_test,y_pred_rf_test)
```

Out[62]:

```
array([[40,  1],
       [ 0, 73]], dtype=int64)
```

In [63]:

```
print(classification_report(y_train,y_pred_rf_train))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       171
           1       1.00      1.00      1.00       284

    accuracy                           1.00       455
   macro avg       1.00      1.00      1.00       455
weighted avg       1.00      1.00      1.00       455
```

In [64]:

```
print(classification_report(y_test,y_pred_rf_test))
```

```
              precision    recall  f1-score   support

           0       1.00      0.98      0.99        41
           1       0.99      1.00      0.99        73

    accuracy                           0.99       114
   macro avg       0.99      0.99      0.99       114
weighted avg       0.99      0.99      0.99       114
```

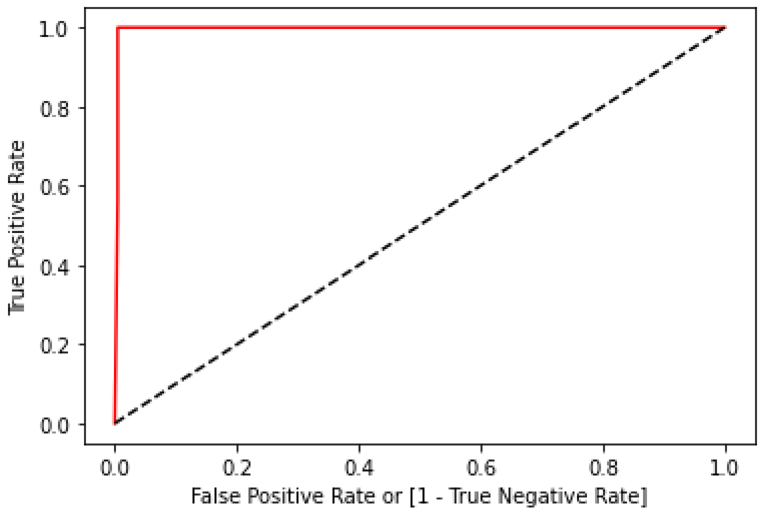
In [76]:

```
fpr, tpr, thresholds = roc_curve(y, classifier_randomforest.predict_proba (x)[:,1])

auc_train = roc_auc_score(y_train, y_pred_rf_train)
print('Auc value for train data: ',auc_train)
auc_test= roc_auc_score(y_test, y_pred_rf_test)
print('Auc value for test data: ',auc_test)

import matplotlib.pyplot as plt
plt.plot(fpr, tpr, color='red')
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False Positive Rate or [1 - True Negative Rate]')
plt.ylabel('True Positive Rate')
```

```
Auc value for train data:  1.0
Auc value for test data:  0.9878048780487805
```

Out[76]:

```
Text(0, 0.5, 'True Positive Rate')
```



Grid searchCv--Random forest

In [49]:

```
from sklearn.model_selection import GridSearchCV
grid_model_rf = GridSearchCV(estimator = classifier_randomforest,param_grid = {'n_estimator
                                                             'criterion' :['gini','entropy'
                                                             'max_depth' :[3,5,7,8,10],
                                                             'min_samples_split' :[2,3,4]})
grid_model_rf.fit(x_train,y_train)
print(grid_model_rf.best_params_)
print(grid_model_rf.best_score_)
```

```
{'criterion': 'gini', 'max_depth': 8, 'min_samples_split': 4, 'n_estimator
s': 20}
0.9626373626373625
```

In [52]:

```
classifier_randomforest1=RandomForestClassifier(n_estimators=20,max_depth=8,criterion='gini
classifier_randomforest1.fit(x_train,y_train)
```

Out[52]:

```
RandomForestClassifier(max_depth=8, min_samples_split=4, n_estimators=20)
```

In [53]:

```
y_pred_rf_train1=classifier_randomforest1.predict(x_train)
y_pred_rf_test1=classifier_randomforest1.predict(x_test)
```

In [54]:

```
confusion_matrix(y_train,y_pred_rf_train1)
```

Out[54]:

```
array([[169,   2],
       [  0, 284]], dtype=int64)
```

In [55]:

```
confusion_matrix(y_test,y_pred_rf_test1)
```

Out[55]:

```
array([[40,  1],
       [ 1, 72]], dtype=int64)
```

In [56]:

```
print(classification_report(y_train,y_pred_rf_train1))
```

```
              precision    recall  f1-score   support

           0       1.00      0.99      0.99       171
           1       0.99      1.00      1.00       284

    accuracy                           1.00       455
   macro avg       1.00      0.99      1.00       455
weighted avg       1.00      1.00      1.00       455
```

In [57]:

```
print(classification_report(y_test,y_pred_rf_test1))
```

```
              precision    recall  f1-score   support

           0       0.98      0.98      0.98        41
           1       0.99      0.99      0.99        73

    accuracy                           0.98       114
   macro avg       0.98      0.98      0.98       114
weighted avg       0.98      0.98      0.98       114
```

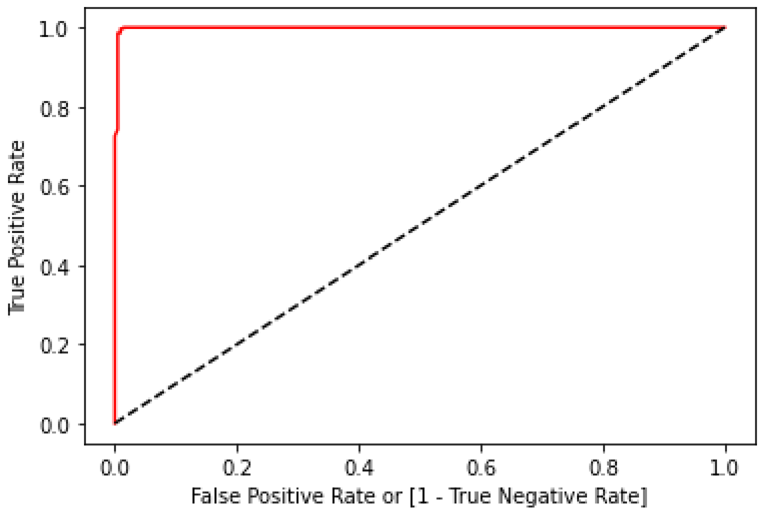
In [77]:

```
fpr, tpr, thresholds = roc_curve(y, classifier_randomforest1.predict_proba (x)[:,1])

auc_train = roc_auc_score(y_train, y_pred_rf_train1)
print('Auc value for train data: ',auc_train)
auc_test= roc_auc_score(y_test, y_pred_rf_test1)
print('Auc value for test data: ',auc_test)

import matplotlib.pyplot as plt
plt.plot(fpr, tpr, color='red')
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False Positive Rate or [1 - True Negative Rate]')
plt.ylabel('True Positive Rate')
```

```
Auc value for train data:  0.9941520467836258
Auc value for test data:  0.9809555629802873
```

Out[77]:

```
Text(0, 0.5, 'True Positive Rate')
```



Inferences

Before GridsearchCV

1.Accuracy of train data=100% test data = 99%

2.Tradeoff between precision anf recall is balanced

3.FN and FP for test values are 1 and 0 respectively

4.Separability between positive and negative value is train data=100%,test data=98.7%

After GridsearchCV

1.Accuracy of train data=100% test data = 98%

2.Tradeoff between precision anf recall is balanced

3.FN and FP for test values are 1 and 1 respectively

4.Separability between positive and negative value is train data=99.41%,test data=98.09%

Logistic regression

In [33]:

```
from sklearn.linear_model import LogisticRegression
classifier_log=LogisticRegression()
```

In [34]:

```
classifier_log.fit(x_train,y_train)
```

```
C:\Users\ROOBA\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.p
y:763: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scik
it-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regre
ssion (https://scikit-learn.org/stable/modules/linear_model.html#logistic-re
gression)
  n_iter_i = _check_optimize_result(
```

Out[34]:

```
LogisticRegression()
```

In [35]:

```
y_pred_loy_train=classifier_log.predict(x_train)
y_pred_log_test=classifier_log.predict(x_test)
```

In [36]:

```
confusion_matrix(y_train,y_pred_loy_train)
```

Out[36]:

```
array([[160,  11],
       [  9, 275]], dtype=int64)
```

In [37]:

```
confusion_matrix(y_test,y_pred_log_test)
```

Out[37]:

```
array([[39,  2],
       [ 0, 73]], dtype=int64)
```

In [38]:

```
print(classification_report(y_train,y_pred_loy_train))
```

```
              precision    recall  f1-score   support

           0       0.95      0.94      0.94       171
           1       0.96      0.97      0.96       284

    accuracy                           0.96       455
   macro avg       0.95      0.95      0.95       455
weighted avg       0.96      0.96      0.96       455
```

In [39]:

```
print(classification_report(y_test,y_pred_log_test))
```

```
              precision    recall  f1-score   support

           0       1.00      0.95      0.97        41
           1       0.97      1.00      0.99        73

    accuracy                           0.98       114
   macro avg       0.99      0.98      0.98       114
weighted avg       0.98      0.98      0.98       114
```

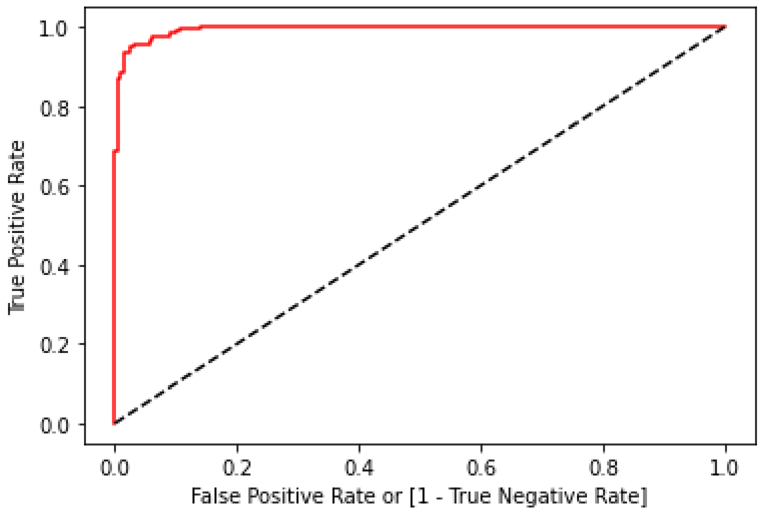
In [78]:

```
fpr, tpr, thresholds = roc_curve(y, classifier_log.predict_proba (x)[:,1])

auc_train = roc_auc_score(y_train, y_pred_loy_train)
print('Auc value for train data: ',auc_train)
auc_test= roc_auc_score(y_test, y_pred_log_test)
print('Auc value for test data: ',auc_test)

import matplotlib.pyplot as plt
plt.plot(fpr, tpr, color='red')
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False Positive Rate or [1 - True Negative Rate]')
plt.ylabel('True Positive Rate')
```

```
Auc value for train data:  0.9519911868874062
Auc value for test data:  0.975609756097561
```

Out[78]:

```
Text(0, 0.5, 'True Positive Rate')
```



Inference

1.Accuracy of train value=96% test value = 98%

2.Tradeoff between precision anf recall is balanced

3.FN and FP for test values are 2 and 0 respectively

4.Separability between positive and negative value is train data=95.19%,test data=97.5%

In [ ]: