Name : Arul Kumar ARK

Roll No. : 225229103

Lab - 5

```
1                          Diabetes Classification using Logistic Regression
```

Step : 1

In [2]:
```
1 import pandas as pd
```

In [3]:
```
1 data = pd.read_csv("diabetes.csv")
```

In [4]:
```
1 data.head(10)
```

Out[4]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
5	5	116	74	0	0	25.6	0.201	30	0
6	3	78	50	32	88	31.0	0.248	26	1
7	10	115	0	0	0	35.3	0.134	29	0
8	2	197	70	45	543	30.5	0.158	53	1
9	8	125	96	0	0	0.0	0.232	54	1

In [5]:
```
1 print(data.columns)
```

```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

In [6]:
```
1 print(data.shape)
```

```
(768, 9)
```

In [7]:
```
1 print(data.dtypes)
```

```
Pregnancies                 int64
Glucose                     int64
BloodPressure               int64
SkinThickness               int64
Insulin                     int64
BMI                       float64
DiabetesPedigreeFunction  float64
Age                         int64
Outcome                     int64
dtype: object
```

In [8]:
```
1 print(data.info)
```

```
<bound method DataFrame.info of      Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0              6      148             72             35        0  33.6
1              1       85             66             29        0  26.6
2              8      183             64              0        0  23.3
3              1       89             66             23       94  28.1
4              0      137             40             35      168  43.1
..           ...      ...            ...            ...      ...   ...
763           10      101             76             48      180  32.9
764            2      122             70             27        0  36.8
765            5      121             72             23      112  26.2
766            1      126             60              0        0  30.1
767            1       93             70             31        0  30.4

     DiabetesPedigreeFunction  Age  Outcome
0                       0.627   50        1
1                       0.351   31        0
2                       0.672   32        1
3                       0.167   21        0
4                       2.288   33        1
..                        ...  ...      ...
763                     0.171   63        0
764                     0.340   27        0
765                     0.245   30        0
766                     0.349   47        1
767                     0.315   23        0

[768 rows x 9 columns]>
```

In [13]:
```
1 print(data.value_counts)
```

```
<bound method DataFrame.value_counts of      Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0              6      148             72             35        0  33.6
1              1       85             66             29        0  26.6
2              8      183             64              0        0  23.3
3              1       89             66             23       94  28.1
4              0      137             40             35      168  43.1
..           ...      ...            ...            ...      ...   ...
763           10      101             76             48      180  32.9
764            2      122             70             27        0  36.8
765            5      121             72             23      112  26.2
766            1      126             60              0        0  30.1
767            1       93             70             31        0  30.4

     DiabetesPedigreeFunction  Age  Outcome
0                       0.627   50        1
1                       0.351   31        0
2                       0.672   32        1
3                       0.167   21        0
4                       2.288   33        1
..                        ...  ...      ...
763                     0.171   63        0
764                     0.340   27        0
765                     0.245   30        0
766                     0.349   47        1
767                     0.315   23        0

[768 rows x 9 columns]>
```

step : 2

In [14]:
```
1 X = data.drop(['Outcome'],axis=1)
```

In [15]:
```
1 X
```

Out[15]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	6	148	72	35	0	33.6	0.627	50
1	1	85	66	29	0	26.6	0.351	31
2	8	183	64	0	0	23.3	0.672	32
3	1	89	66	23	94	28.1	0.167	21
4	0	137	40	35	168	43.1	2.288	33
...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63
764	2	122	70	27	0	36.8	0.340	27
765	5	121	72	23	112	26.2	0.245	30
766	1	126	60	0	0	30.1	0.349	47
767	1	93	70	31	0	30.4	0.315	23

768 rows × 8 columns

In [16]:
```
1 y = data['Outcome']
```

In [17]:
```
1 print(y)
```

```
0      1
1      0
2      1
3      0
4      1
      ..
763    0
764    0
765    0
766    1
767    0
Name: Outcome, Length: 768, dtype: int64
```

In [23]:
```
1 from sklearn.model_selection import StratifiedShuffleSplit
2 splitter=StratifiedShuffleSplit(n_splits=1,test_size=0.25,random_state=12)
3 for train,test in splitter.split(X,y):
4     X_train = X.loc[train]
5     y_train = y.loc[train]
6     X_test = X.loc[test]
7     y_test = y.loc[test]
```

In [27]:
```
1 from sklearn.linear_model import LogisticRegression
2 model=LogisticRegression()
3 model.fit(X_train,y_train)
4 y_pred_m=model.predict(X_test)
```

```
C:\Users\arulk\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:814: ConvergenceWarning: lbfgs failed to conver
ge (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://scikit-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-regression (https://scikit-learn.org/stable/modules/l
inear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

In [29]:
```
1 y_pred_m
```

Out[29]:
```
array([1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0,
       1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0,
       0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1,
       0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1], dtype=int64)
```

Step : 3

In [31]:
```
new=model.predict([[6,200,90,10,25,23.3,.672,42]])
if new==0:
    print("Non-diabetic :",new)
else:
    print("Diabetic : ",new)
```

```
Diabetic :  [1]

C:\Users\arulk\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature names, but LogisticRegression was fitted with feature names
  warnings.warn(
```

In [72]:
```
print('Accuracy : ' ,sum(y_test==y_pred_m)/float(y_test.shape[0]))
```

```
Accuracy :  0.796875
```

In [73]:
```
from sklearn.metrics import precision_score
print('Precision : ',precision_score(y_test,y_pred_m))
```

```
Precision :  0.7258064516129032
```

In [74]:
```
from sklearn.metrics import recall_score
print('Recall : ',recall_score(y_test,y_pred_m))
```

```
Recall :  0.6716417910447762
```

In [75]:
```
from sklearn.metrics import roc_auc_score
print('AUC Scoure : ',roc_auc_score(y_test,y_pred_m))
```

```
AUC Scoure :  0.767820895522388
```

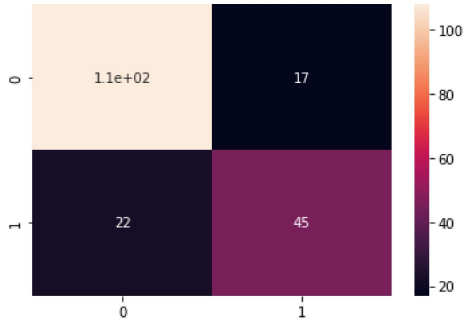
Step : 4

In [76]:
```
from sklearn.metrics import confusion_matrix
matrix=confusion_matrix(y_test,y_pred_m)
matrix
```

Out[76]:
```
array([[108,  17],
       [ 22,  45]], dtype=int64)
```

In [77]:
```
import seaborn as sns
sns.heatmap(matrix, annot=True)
```

Out[77]: <AxesSubplot:>



Step : 5

In [78]:
```
from sklearn.preprocessing import MinMaxScaler
mm=MinMaxScaler()
mm_X_train=mm.fit_transform(X_train)
mm_X_train
```

Out[78]:
```
array([[0.        , 0.62311558, 0.57377049, ..., 0.40834575, 0.07514944,
        0.25      ],
       [0.17647059, 0.96984925, 0.57377049, ..., 0.52011923, 0.06959863,
        0.06666667],
       [0.58823529, 0.61306533, 0.63934426, ..., 0.41132638, 0.1853117 ,
        0.4       ],
       ...,
       [0.29411765, 0.73869347, 0.63934426, ..., 0.50223547, 0.05977797,
        0.73333333],
       [0.17647059, 0.64824121, 0.52459016, ..., 0.39344262, 0.06020495,
        0.11666667],
       [0.        , 0.63316583, 0.70491803, ..., 0.40834575, 0.18659266,
        0.        ]])
```

In [79]:
```
mm_X_test=mm.transform(X_test)
mm_X_test
```

Out[79]:
```
array([[0.23529412, 0.77386935, 0.59016393, ..., 0.46646796, 0.11101623,
        0.26666667],
       [0.05882353, 0.78894472, 0.59016393, ..., 0.38152012, 0.01921435,
        0.05      ],
       [0.76470588, 0.79396985, 0.93442623, ..., 0.63040238, 0.0764304 ,
        0.38333333],
       ...,
       [0.05882353, 0.61306533, 0.73770492, ..., 0.74068554, 0.10546541,
        0.16666667],
       [0.17647059, 0.75376884, 0.62295082, ..., 0.31296572, 0.05508113,
        0.26666667],
       [0.11764706, 0.64321608, 0.52459016, ..., 0.59612519, 0.43680615,
        0.05      ]])
```

In [80]:
```
mm_lor=LogisticRegression()
mm_lor=mm_lor.fit(mm_X_train,y_train)
mm_y_pred=mm_lor.predict(mm_X_test)
mm_y_pred
```

Out[80]:
```
array([0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0,
       0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1,
       0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0], dtype=int64)
```

In [81]:
```
print('Accuracy : ' ,sum(y_test==mm_y_pred)/float(y_test.shape[0]))

```

```
Accuracy :  0.7864583333333334
```

In [82]:
```
from sklearn.metrics import precision_score
print('Precision : ',precision_score(mm_y_pred,y_pred_m))
```

```
Precision :  0.8064516129032258
```

In [83]:
```
from sklearn.metrics import recall_score
print('Recall : ',recall_score(mm_y_pred,y_pred_m))
```

```
Recall :  1.0
```

In [84]:
```
from sklearn.metrics import roc_auc_score
print('AUC Scoure : ',roc_auc_score(mm_y_pred,y_pred_m))
```

```
AUC Scoure :  0.9577464788732395
```

Step : 6

In [85]:
```
from sklearn.preprocessing import StandardScaler
ss=StandardScaler()
ss_X_train=ss.fit_transform(X_train)
ss_X_train
```

Out[85]:
```
array([[-1.12598572,  0.11956974,  0.06558538, ..., -0.59413103,
        -0.65316149,  0.23834618],
       [-0.23468782,  2.31457195,  0.06558538, ...,  0.34990533,
        -0.69286658, -0.70256434],
       [ 1.84500729,  0.05594649,  0.46376382, ..., -0.56895673,
         0.13483187,  1.00818206],
       ...,
       [ 0.35951079,  0.85123714,  0.46376382, ...,  0.19885951,
        -0.76311405,  2.71892846],
       [-0.23468782,  0.27862787, -0.23304845, ..., -0.72000254,
        -0.76005981, -0.44595238],
       [-1.12598572,  0.18319299,  0.86194226, ..., -0.59413103,
         0.14399459, -1.04471362]])
```

In [86]:
```
ss_X_test=ss.transform(X_test)
ss_X_test
```

Out[86]:
```
array([[ 0.06241149,  1.07391853,  0.16512999, ..., -0.10323212,
        -0.39660551,  0.3238835 ],
       [-0.82888642,  1.1693534 ,  0.16512999, ..., -0.82069975,
        -1.05326665, -0.78810166],
       [ 2.73630519,  1.20116503,  2.2555668 , ...,  1.28135453,
        -0.64399878,  0.92264474],
       ...,
       [-0.82888642,  0.05594649,  1.06103148, ...,  2.21280374,
        -0.4363106 , -0.18934042],
       [-0.23468782,  0.94667202,  0.36421921, ..., -1.39970872,
        -0.79671067,  0.3238835 ],
       [-0.53178712,  0.24681625, -0.23304845, ...,  0.99185005,
         1.93377797, -0.78810166]])
```

In [87]:
```
ss_lor=LogisticRegression()
ss_lor.fit(ss_X_train,y_train)
ss_y_pred=ss_lor.predict(ss_X_test)
```

In [88]:
```
print('Accuracy : ' ,sum(y_test==ss_y_pred)/float(y_test.shape[0]))

from sklearn.metrics import precision_score
print('Precision : ',precision_score(ss_y_pred,y_pred_m))

from sklearn.metrics import recall_score
print('Recall : ',recall_score(ss_y_pred,y_pred_m))

from sklearn.metrics import roc_auc_score
print('AUC Scoure : ',roc_auc_score(ss_y_pred,y_pred_m))
```

```
Accuracy :  0.7864583333333334
Precision :  0.967741935483871
Recall :  1.0
AUC Scoure :  0.9924242424242424
```

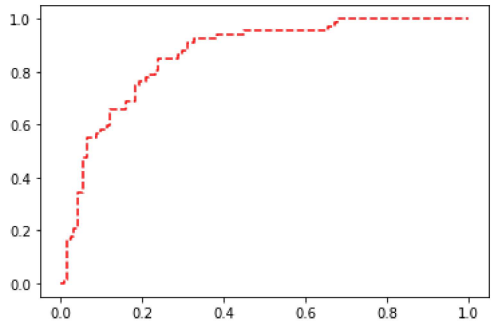
Step : 7

In [89]:
```
pred_prob1=mm_lor.predict_proba(mm_X_test)
```

In [90]:
```
from sklearn.metrics import roc_curve
fpr, tpr, thresh = roc_curve(y_test, pred_prob1[:,1], pos_label=1)
```

In [91]:
```
import matplotlib.pyplot as plt
plt.plot(fpr,tpr,linestyle='--',color='red',label='MinMaxScaler values')
```

Out[91]: [<matplotlib.lines.Line2D at 0x1d79c33be80>]



Step : 8

In [94]:
```
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier(n_neighbors=4)
knn=knn.fit(X_train,y_train)
```

In [101]:
```
knn_y_pred=knn.predict(X_test)
```

In [102]:
```
from sklearn.preprocessing import MinMaxScaler
m=MinMaxScaler()
m_X_train=m.fit_transform(X_train)
m_X_train
```

Out[102]:
```
array([[0.        , 0.62311558, 0.57377049, ..., 0.40834575, 0.07514944,
        0.25      ],
       [0.17647059, 0.96984925, 0.57377049, ..., 0.52011923, 0.06959863,
        0.06666667],
       [0.58823529, 0.61306533, 0.63934426, ..., 0.41132638, 0.1853117 ,
        0.4       ],
       ...,
       [0.29411765, 0.73869347, 0.63934426, ..., 0.50223547, 0.05977797,
        0.73333333],
       [0.17647059, 0.64824121, 0.52459016, ..., 0.39344262, 0.06020495,
        0.11666667],
       [0.        , 0.63316583, 0.70491803, ..., 0.40834575, 0.18659266,
        0.        ]])
```

In [103]:
```
m_X_test=m.transform(X_test)
m_X_test
```

Out[103]:
```
array([[0.23529412, 0.77386935, 0.59016393, ..., 0.46646796, 0.11101623,
        0.26666667],
       [0.05882353, 0.78894472, 0.59016393, ..., 0.38152012, 0.01921435,
        0.05      ],
       [0.76470588, 0.79396985, 0.93442623, ..., 0.63040238, 0.0764304 ,
        0.38333333],
       ...,
       [0.05882353, 0.61306533, 0.73770492, ..., 0.74068554, 0.10546541,
        0.16666667],
       [0.17647059, 0.75376884, 0.62295082, ..., 0.31296572, 0.05508113,
        0.26666667],
       [0.11764706, 0.64321608, 0.52459016, ..., 0.59612519, 0.43680615,
        0.05      ]])
```

In [104]:
```
m_knn=KNeighborsClassifier()
m_knn=m_knn.fit(m_X_train,y_train)
```

In [105]:
```
m_y_pred=m_knn.predict(m_X_test)
m_y_pred
```

Out[105]:
```
array([0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0,
       1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1,
       0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0], dtype=int64)
```

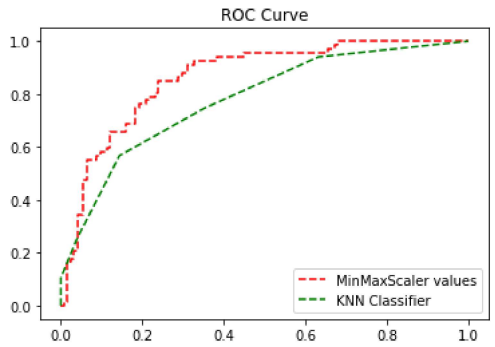
In [108]:
```
print('Accuracy : ' ,sum(y_test==m_y_pred)/float(y_test.shape[0]))

from sklearn.metrics import precision_score
print('Precision : ',precision_score(m_y_pred,y_pred_m))

from sklearn.metrics import recall_score
print('Recall : ',recall_score(m_y_pred,y_pred_m))

from sklearn.metrics import roc_auc_score
print('AUC Scoure : ',roc_auc_score(m_y_pred,y_pred_m))
```

```
Accuracy :  0.7552083333333334
Precision :  0.6935483870967742
Recall :  0.7678571428571429
AUC Scoure :  0.8140756302521008
```

Step : 9

In [109]:
```
pred_prob2=m_knn.predict_proba(m_X_test)
```

In [110]:
```
from sklearn.metrics import roc_curve
fpr1,tpr1,thresh1=roc_curve(y_test,pred_prob2[:,1],pos_label=1)
import matplotlib.pyplot as plt
plt.plot(fpr,tpr,linestyle='--',color='red',label='MinMaxScaler values')
plt.plot(fpr1,tpr1,linestyle='--',color='green',label='KNN Classifier')
plt.title('ROC Curve')
plt.legend(loc='best')
plt.savefig('ROC',dpi=300)
plt.show()
```



Step : 10

In [114]:
```
from sklearn.linear_model import LogisticRegressionCV
model1=LogisticRegressionCV(Cs=10,cv=4,penalty='l1',solver='liblinear')
model2=LogisticRegressionCV(Cs=10,cv=4,penalty='l2')
model1.fit(mm_X_train,y_train)
model2.fit(mm_X_train,y_train)
```

Out[114]:
```
LogisticRegressionCV(cv=4)
```

In [116]:
```
rg_y_pred1 = model1.predict(mm_X_test)
rg_y_pred2 = model2.predict(mm_X_test)
```

AUC SCORE OF L1

In [118]:
```
from sklearn.metrics import roc_auc_score
l1auc = roc_auc_score(y_test, rg_y_pred1)
l1auc = (' LOR L1 MinMax AUC', l1auc)
print(l1auc)
```

```
(' LOR L1 MinMax AUC', 0.7465074626865671)
```

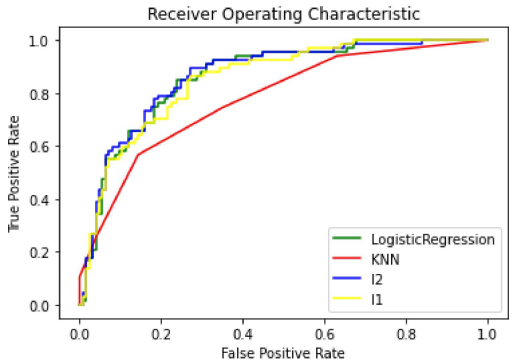
AUC SCORE OF L2

In [119]:
```
l2auc = roc_auc_score(y_test, rg_y_pred2)
l2auc = (' LOR L2 MinMax AUC', l2auc)
l2auc
```

Out[119]:
```
(' LOR L2 MinMax AUC', 0.7568955223880597)
```

Step : 11

In [122]:
```
pred_prb7 = model1.predict_proba(mm_X_test)
pred_prb8 = model2.predict_proba(mm_X_test)
fpr,tbr,threshold = roc_curve(y_test, pred_prob1[:,1],pos_label=1)
fpr1,tbr1,threshold1 = roc_curve(y_test, pred_prob2[:,1],pos_label=1)
fpr2,tbr2,threshold2= roc_curve(y_test, pred_prb7[:,1],pos_label=1)
fpr3,tbr3,threshold3 = roc_curve(y_test, pred_prb8[:,1],pos_label=1)
plt.plot(fpr, tbr, linestyle='-', color='green', label='LogisticRegression')
plt.plot(fpr1, tbr1, linestyle='-', color='red', label='KNN')
plt.plot(fpr3, tbr3, linestyle='-', color='blue', label='l2')
plt.plot(fpr2, tbr2, linestyle='-', color='yellow', label='l1')
plt.title('Receiver Operating Characteristic')
plt.legend(loc = 'best')
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```



In [ ]:
```

```