225229108

DINESH KUMAR

Importing csv file

STEP1:

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

In [2]:
```
data=pd.read_csv('diabetes.csv')
data
```

Out[2]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction
0	6	148	72	35	0	33.6	0.627
1	1	85	66	29	0	26.6	0.351
2	8	183	64	0	0	23.3	0.672
3	1	89	66	23	94	28.1	0.167
4	0	137	40	35	168	43.1	2.288
5	5	116	74	0	0	25.6	0.201
6	3	78	50	32	88	31.0	0.248
7	10	115	0	0	0	35.3	0.134
8	2	197	70	45	543	30.5	0.158
9	8	125	96	0	0	0.0	0.232
10	4	110	92	0	0	37.6	0.191
11	10	168	74	0	0	38.0	0.537
12	10	139	80	0	0	27.1	1.441
13	1	189	60	23	846	30.1	0.398
14	5	166	72	19	175	25.8	0.587
15	7	100	0	0	0	30.0	0.484
16	0	118	84	47	230	45.8	0.551
17	7	107	74	0	0	29.6	0.254
18	1	103	30	38	83	43.3	0.183
19	1	115	70	30	96	34.6	0.529
20	3	126	88	41	235	39.3	0.704
21	8	99	84	0	0	35.4	0.388
22	7	196	90	0	0	39.8	0.451
23	9	119	80	35	0	29.0	0.263
24	11	143	94	33	146	36.6	0.254
25	10	125	70	26	115	31.1	0.205
26	7	147	76	0	0	39.4	0.257
27	1	97	66	15	140	23.2	0.487
28	13	145	82	19	110	22.2	0.245
29	5	117	92	0	0	34.1	0.337
...	...	...	...	...	...	...	...
738	2	99	60	17	160	36.6	0.453
739	1	102	74	0	0	39.5	0.293

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction
740	11	120	80	37	150	42.3	0.785
741	3	102	44	20	94	30.8	0.400
742	1	109	58	18	116	28.5	0.219
743	9	140	94	0	0	32.7	0.734
744	13	153	88	37	140	40.6	1.174
745	12	100	84	33	105	30.0	0.488
746	1	147	94	41	0	49.3	0.358
747	1	81	74	41	57	46.3	1.096
748	3	187	70	22	200	36.4	0.408
749	6	162	62	0	0	24.3	0.178
750	4	136	70	0	0	31.2	1.182
751	1	121	78	39	74	39.0	0.261
752	3	108	62	24	0	26.0	0.223
753	0	181	88	44	510	43.3	0.222
754	8	154	78	32	0	32.4	0.443
755	1	128	88	39	110	36.5	1.057
756	7	137	90	41	0	32.0	0.391
757	0	123	72	0	0	36.3	0.258
758	1	106	76	0	0	37.5	0.197
759	6	190	92	0	0	35.5	0.278
760	2	88	58	26	16	28.4	0.766
761	9	170	74	31	0	44.0	0.403
762	9	89	62	0	0	22.5	0.142
763	10	101	76	48	180	32.9	0.171
764	2	122	70	27	0	36.8	0.340
765	5	121	72	23	112	26.2	0.245
766	1	126	60	0	0	30.1	0.349
767	1	93	70	31	0	30.4	0.315

768 rows × 9 columns

In [3]:
```
data.head()
```

Out[3]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	A
0	6	148	72	35	0	33.6	0.627	
1	1	85	66	29	0	26.6	0.351	
2	8	183	64	0	0	23.3	0.672	
3	1	89	66	23	94	28.1	0.167	
4	0	137	40	35	168	43.1	2.288	

In [5]:
```
data.shape
```

Out[5]:
```
(768, 9)
```

In [6]:
```
data.columns
```

Out[6]:
```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

In [7]:
```
data.dtypes
```

Out[7]:
```
Pregnancies                 int64
Glucose                     int64
BloodPressure               int64
SkinThickness               int64
Insulin                     int64
BMI                       float64
DiabetesPedigreeFunction  float64
Age                         int64
Outcome                     int64
dtype: object
```

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
Pregnancies                 768 non-null int64
Glucose                     768 non-null int64
BloodPressure               768 non-null int64
SkinThickness               768 non-null int64
Insulin                     768 non-null int64
BMI                         768 non-null float64
DiabetesPedigreeFunction    768 non-null float64
Age                         768 non-null int64
Outcome                     768 non-null int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

In [12]:
```
data.BloodPressure.value_counts
```

Out[12]:
```
<bound method IndexOpsMixin.value_counts of 0      72
1      66
2      64
3      66
4      40
5      74
6      50
7       0
8      70
9      96
10     92
11     74
12     80
13     60
14     72
15      0
16     84
17     74
18     30
19     70
20     88
21     84
22     90
23     80
24     94
25     70
26     76
27     66
28     82
29     92
       ..
738    60
739    74
740    80
741    44
742    58
743    94
744    88
745    84
746    94
747    74
748    70
749    62
750    70
751    78
752    62
753    88
754    78
755    88
756    90
757    72
758    76
759    92
760    58
761    74
```

```
762    62
763    76
764    70
765    72
766    60
767    70
Name: BloodPressure, Length: 768, dtype: int64>
```

Step-2.[Build Logistic Regression model]

In [13]:
```
X=data[['Pregnancies','Glucose','BloodPressure','SkinThickness','Insulin','BMI','
```

In [15]:
```
Y=data[['Outcome']]
```

In [16]:
```
from sklearn.model_selection import train_test_split
```

In [18]:
```
X_train,X_test,y_train,y_test=train_test_split(X,Y,test_size=0.25,random_state=42
```

In [19]:
```
X_train
```

Out[19]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction
357	13	129	0	30	0	39.9	0.569
73	4	129	86	20	270	35.1	0.231
352	3	61	82	28	0	34.4	0.243
497	2	81	72	15	76	30.1	0.547
145	0	102	75	23	0	0.0	0.572
514	3	99	54	19	86	25.6	0.154
291	0	107	62	30	74	36.6	0.757
132	3	170	64	37	225	34.5	0.356
559	11	85	74	0	0	30.1	0.300
631	0	102	78	40	90	34.5	0.238
719	5	97	76	27	0	35.6	0.378
395	2	127	58	24	275	27.7	1.600
41	7	133	84	0	0	40.2	0.696
637	2	94	76	18	66	31.6	0.649
108	3	83	58	31	18	34.3	0.336
481	0	123	88	37	0	35.2	0.197
56	7	187	68	39	304	37.7	0.254
323	13	152	90	33	29	26.8	0.731
685	2	129	74	26	205	33.2	0.591
758	1	106	76	0	0	37.5	0.197
572	3	111	58	31	44	29.5	0.430
529	0	111	65	0	0	24.6	0.660
24	11	143	94	33	146	36.6	0.254
465	0	124	56	13	105	21.8	0.452
247	0	165	90	33	680	52.3	0.427
443	8	108	70	0	0	30.5	0.955
351	4	137	84	0	0	31.2	0.252
327	10	179	70	0	0	35.1	0.200
110	3	171	72	33	135	33.3	0.199
82	7	83	78	26	71	29.3	0.767
...	...	...	...	...	...	...	...
21	8	99	84	0	0	35.4	0.388
313	3	113	50	10	85	29.5	0.626
459	9	134	74	33	60	25.9	0.460

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction
160	4	151	90	38	0	29.7	0.294
276	7	106	60	24	0	26.5	0.296
191	9	123	70	44	94	33.1	0.374
385	1	119	54	13	50	22.3	0.205
413	1	143	74	22	61	26.2	0.256
491	2	89	90	30	0	33.5	0.292
343	5	122	86	0	0	34.7	0.290
308	0	128	68	19	180	30.5	1.391
661	1	199	76	43	0	42.9	1.394
130	4	173	70	14	168	29.7	0.361
663	9	145	80	46	130	37.9	0.637
99	1	122	90	51	220	49.7	0.325
372	0	84	64	22	66	35.8	0.545
87	2	100	68	25	71	38.5	0.324
458	10	148	84	48	237	37.6	1.001
330	8	118	72	19	0	23.1	1.476
214	9	112	82	32	175	34.2	0.260
466	0	74	52	10	36	27.8	0.269
121	6	111	64	39	0	34.2	0.260
614	11	138	74	26	144	36.1	0.557
20	3	126	88	41	235	39.3	0.704
700	2	122	76	27	200	35.9	0.483
71	5	139	64	35	140	28.6	0.411
106	1	96	122	0	0	22.4	0.207
270	10	101	86	37	0	45.6	1.136
435	0	141	0	0	0	42.4	0.205
102	0	125	96	0	0	22.5	0.262

576 rows × 8 columns

In [20]:
```
X_test
```

Out[20]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction
668	6	98	58	33	190	34.0	0.430
324	2	112	75	32	0	35.7	0.148
624	2	108	64	0	0	30.8	0.158
690	8	107	80	0	0	24.6	0.856
473	7	136	90	0	0	29.9	0.210
204	6	103	72	32	190	37.7	0.324
97	1	71	48	18	76	20.4	0.323
336	0	117	0	0	0	33.8	0.932
568	4	154	72	29	126	31.3	0.338
148	5	147	78	0	0	33.7	0.218
667	10	111	70	27	0	27.5	0.141
212	7	179	95	31	0	34.2	0.164
199	4	148	60	27	318	30.9	0.150
265	5	96	74	18	67	33.6	0.997
760	2	88	58	26	16	28.4	0.766
356	1	125	50	40	167	33.3	0.962
501	3	84	72	32	0	37.2	0.267
457	5	86	68	28	71	30.2	0.364
604	4	183	0	0	0	28.4	0.212
213	0	140	65	26	130	42.6	0.431
636	5	104	74	0	0	28.8	0.153
544	1	88	78	29	76	32.0	0.365
86	13	106	72	54	0	36.6	0.178
208	1	96	64	27	87	33.2	0.289
281	10	129	76	28	122	35.9	0.280
209	7	184	84	33	0	35.5	0.355
581	6	109	60	27	0	25.0	0.206
639	1	100	74	12	46	19.5	0.149
328	2	102	86	36	120	45.5	0.127
431	3	89	74	16	85	30.4	0.551
...	...	...	...	...	...	...	...
718	1	108	60	46	178	35.5	0.415
90	1	80	55	0	0	19.1	0.258
377	1	87	60	37	75	37.2	0.509

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction
235	4	171	72	0	0	43.6	0.479
158	2	88	74	19	53	29.0	0.229
69	4	146	85	27	100	28.9	0.189
260	3	191	68	15	130	30.9	0.299
131	9	122	56	0	0	33.3	1.114
44	7	159	64	0	0	27.4	0.294
70	2	100	66	20	90	32.9	0.867
264	4	123	62	0	0	32.0	0.226
673	3	123	100	35	240	57.3	0.880
286	5	155	84	44	545	38.7	0.619
640	0	102	86	17	105	29.3	0.695
135	2	125	60	20	140	33.8	0.088
745	12	100	84	33	105	30.0	0.488
165	6	104	74	18	156	29.9	0.722
164	0	131	88	0	0	31.6	0.743
28	13	145	82	19	110	22.2	0.245
608	0	152	82	39	272	41.5	0.270
583	8	100	76	0	0	38.7	0.190
746	1	147	94	41	0	49.3	0.358
292	2	128	78	37	182	43.3	1.224
136	0	100	70	26	50	30.8	0.597
432	1	80	74	11	60	30.0	0.527
554	1	84	64	23	115	36.9	0.471
319	6	194	78	0	0	23.5	0.129
594	6	123	72	45	230	33.6	0.733
6	3	78	50	32	88	31.0	0.248
615	3	106	72	0	0	25.8	0.207

192 rows × 8 columns

In [21]:
```
from sklearn import linear_model
```

In [23]:
```
logr = linear_model.LogisticRegression()
logr.fit(X,Y)
```

```
C:\Program Files (x86)\Microsoft Visual Studio\Shared\Anaconda3_64\lib\site-pac
kages\sklearn\utils\validation.py:578: DataConversionWarning: A column-vector y
was passed when a 1d array was expected. Please change the shape of y to (n_sam
ples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Out[23]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)
```

In [24]:
```
y_pred=logr.predict(X_test)
y_pred
```

Out[24]:
```
array([0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0,
       0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0,
       0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0,
       0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1,
       0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0,
       0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0], dtype=int64)
```

Step-3. [Predict on a new sample]

In [28]:
```
new=logr.predict([[6,200,90,10,25,23.3,.672,42]])
if new==0:
    print("Non-diabetic patient",new)
else:
    print("Diabetic patient",new)
```

```
Diabetic patient [1]
```

Precision

In [31]:
```
from sklearn.metrics import precision_score
print(precision_score(y_test,y_pred))
```

```
0.7
```

Recall

In [32]:
```
from sklearn.metrics import recall_score
print(recall_score(y_test,y_pred))
```

```
0.6086956521739131
```

AUC Scores

In [33]:
```
from sklearn.metrics import roc_auc_score
print(roc_auc_score(y_test,y_pred))
```

```
0.7311770943796395
```

In [37]:
```
from sklearn import metrics
from sklearn.linear_model import LogisticRegression
```

In [39]:
```
model = LogisticRegression(random_state=0, solver='lbfgs',multi_class='multinomia
model.fit(X_train,y_train)
prediction=model.predict(X_test)
print('The accuracy of the Logistic Regression is', metrics.accuracy_score(predic
```

```
C:\Program Files (x86)\Microsoft Visual Studio\Shared\Anaconda3_64\lib\site-pac
kages\sklearn\utils\validation.py:578: DataConversionWarning: A column-vector y
was passed when a 1d array was expected. Please change the shape of y to (n_sam
ples, ), for example using ravel().
  y = column_or_1d(y, warn=True)

The accuracy of the Logistic Regression is 0.7291666666666666
```

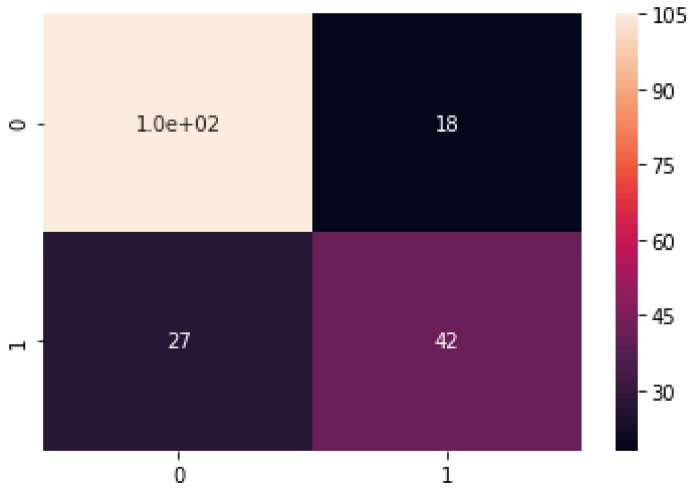
Step-4. [Understand Correlation]

In [34]:
```
from sklearn.metrics import confusion_matrix
cfm=confusion_matrix(y_test,y_pred)
cfm
```

Out[34]:
```
array([[105,  18],
       [ 27,  42]], dtype=int64)
```

In [36]:
```
import seaborn as sns
sns.heatmap(cfm, annot=True)
```

Out[36]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2d4d7f94cf8>
```



Step-5. [Normalization using MinMaxScaler and rebuild LOR]

In [40]:
```
#Normalizing using MinMaxScaler
from sklearn.preprocessing import MinMaxScaler
mm=MinMaxScaler()
mm_X_train=mm.fit_transform(X_train)
mm_X_train
```

Out[40]:
```
array([[0.76470588, 0.64824121, 0.        , ..., 0.59463487, 0.20964987,
        0.38333333],
       [0.23529412, 0.64824121, 0.70491803, ..., 0.52309985, 0.06532878,
        0.03333333],
       [0.17647059, 0.30653266, 0.67213115, ..., 0.51266766, 0.0704526 ,
        0.41666667],
       ...,
       [0.58823529, 0.50753769, 0.70491803, ..., 0.67958271, 0.45175064,
        0.28333333],
       [0.        , 0.70854271, 0.        , ..., 0.6318927 , 0.05422716,
        0.13333333],
       [0.        , 0.6281407 , 0.78688525, ..., 0.33532042, 0.07856533,
        0.        ]])
```

In [41]:
```
mm_X_test=mm.transform(X_test)
mm_X_test
```

Out[41]:
```
array([[0.35294118, 0.49246231, 0.47540984, ..., 0.50670641, 0.15029889,
        0.36666667],
       [0.11764706, 0.56281407, 0.6147541 , ..., 0.53204173, 0.02988898,
        0.        ],
       [0.11764706, 0.54271357, 0.52459016, ..., 0.45901639, 0.03415884,
        0.        ],
       ...,
       [0.35294118, 0.61809045, 0.59016393, ..., 0.50074516, 0.27967549,
        0.21666667],
       [0.17647059, 0.3919598 , 0.40983607, ..., 0.46199702, 0.07258753,
        0.08333333],
       [0.17647059, 0.53266332, 0.59016393, ..., 0.38450075, 0.05508113,
        0.1       ]])
```

In [42]:
```
#ReBuild LOR Model
mm_lor=LogisticRegression()
mm_lor=mm_lor.fit(mm_X_train,y_train)
```

```
C:\Program Files (x86)\Microsoft Visual Studio\Shared\Anaconda3_64\lib\site-pac
kages\sklearn\utils\validation.py:578: DataConversionWarning: A column-vector y
was passed when a 1d array was expected. Please change the shape of y to (n_sam
ples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

In [43]:
```
mm_y_pred=mm_lor.predict(mm_X_test)
mm_y_pred
```

Out[43]:
```
array([0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0,
       0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 1, 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, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0], dtype=int64)
```

Accuracy

In [44]:
```
model = LogisticRegression(random_state=0, solver='lbfgs',multi_class='multinomia
model.fit(X_train,y_train)
prediction=model.predict(X_test)
print('The accuracy of the Logistic Regression is', metrics.accuracy_score(y_test
```

```
The accuracy of the Logistic Regression is 0.7239583333333334

C:\Program Files (x86)\Microsoft Visual Studio\Shared\Anaconda3_64\lib\site-pac
kages\sklearn\utils\validation.py:578: DataConversionWarning: A column-vector y
was passed when a 1d array was expected. Please change the shape of y to (n_sam
ples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Precision

In [45]:
```
print(precision_score(y_test,mm_y_pred))
```

```
0.6538461538461539
```

Recall

In [46]:
```
print(recall_score(y_test,mm_y_pred))
```

```
0.4927536231884058
```

AUC scores

In [47]:
```
mm_auc=print(roc_auc_score(y_test,mm_y_pred))
mm_auc
```

```
0.6732060798868859
```

Step-6: [Normalization using StandardScaler and rebuild LOR]

In [48]:
```
from sklearn.preprocessing import StandardScaler
ss=StandardScaler()
ss_X_train=ss.fit_transform(X_train)
ss_X_train
```

Out[48]:
```
array([[ 2.80346794,  0.25977903, -3.78077929, ...,  1.03974028,
         0.29608546,  0.96352088],
       [ 0.07832678,  0.25977903,  0.89724451, ...,  0.40945373,
        -0.70087555, -0.86295593],
       [-0.22446668, -1.85825286,  0.67966201, ...,  0.31753694,
        -0.66548048,  1.13747105],
       ...,
       [ 1.89508755, -0.61235174,  0.89724451, ...,  1.78820556,
         1.96850229,  0.44167036],
       [-1.13284707,  0.63354937, -3.78077929, ...,  1.36801453,
        -0.77756486, -0.34110542],
       [-1.13284707,  0.13518892,  1.44120077, ..., -1.24504846,
        -0.6094383 , -1.03690611]])
```

In [49]:
```
ss_X_test=ss.transform(X_test)
ss_X_test
```

Out[49]:
```
array([[ 0.6839137 , -0.70579433, -0.625833  , ...,  0.26501306,
        -0.11390738,  0.87654579],
       [-0.52726014, -0.26972894,  0.29889263, ...,  0.48823955,
        -0.94569142, -1.03690611],
       [-0.52726014, -0.39431905, -0.29945925, ..., -0.15517797,
        -0.91619553, -1.03690611],
       ...,
       [ 0.6839137 ,  0.07289387,  0.13570575, ...,  0.21248918,
         0.77981801,  0.09377001],
       [-0.22446668, -1.32874488, -1.06099801, ..., -0.12891603,
        -0.65073254, -0.60203068],
       [-0.22446668, -0.45661411,  0.13570575, ..., -0.81172646,
        -0.77166568, -0.51505559]])
```

In [50]:
```
#Rebuild LOR
ss_lor=LogisticRegression()
ss_lor.fit(ss_X_train,y_train)
ss_y_pred=ss_lor.predict(ss_X_test)
```

```
C:\Program Files (x86)\Microsoft Visual Studio\Shared\Anaconda3_64\lib\site-pac
kages\sklearn\utils\validation.py:578: DataConversionWarning: A column-vector y
was passed when a 1d array was expected. Please change the shape of y to (n_sam
ples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Accuracy

In [51]:
```
model = LogisticRegression(random_state=0, solver='lbfgs',multi_class='multinomia
model.fit(X_train,y_train)
prediction=model.predict(X_test)
print('The accuracy of the Logistic Regression is', metrics.accuracy_score(y_test
```

```
The accuracy of the Logistic Regression is 0.7291666666666666

C:\Program Files (x86)\Microsoft Visual Studio\Shared\Anaconda3_64\lib\site-pac
kages\sklearn\utils\validation.py:578: DataConversionWarning: A column-vector y
was passed when a 1d array was expected. Please change the shape of y to (n_sam
ples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Precision

In [52]:
```
print(precision_score(y_test,ss_y_pred))
```

```
0.6164383561643836
```

Recall

In [53]:
```
print(recall_score(y_test,ss_y_pred))
```

```
0.6521739130434783
```

AUC scores

In [54]:
```
auc_ss=print(roc_auc_score(y_test,ss_y_pred))
auc_ss
```

```
0.7122658183103571
```

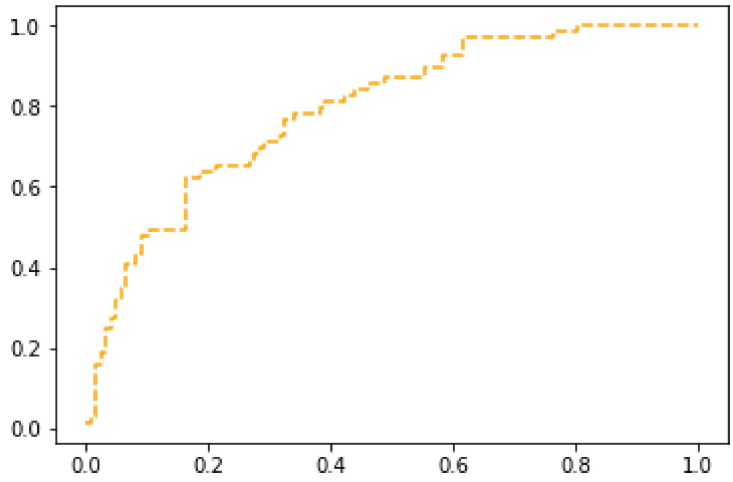
Step-7. [Plot ROC Curve]

In [55]:
```
pred_prob1=mm_lor.predict_proba(mm_X_test)
```

In [56]:
```
from sklearn.metrics import roc_curve
fpr1, tpr1, thresh1 = roc_curve(y_test, pred_prob1[:,1], pos_label=1)
```

In [57]:
```
import matplotlib.pyplot as plt
plt.plot(fpr1,tpr1,linestyle='--',color='orange',label='MinMaxScaler values')
```

Out[57]:
```
[<matplotlib.lines.Line2D at 0x2d4d805d0f0>]
```



Step-8. [Comparison with KNN classifier]

In [58]:
```
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier(n_neighbors=4)
knn=knn.fit(X_train,y_train)
```

```
C:\Program Files (x86)\Microsoft Visual Studio\Shared\Anaconda3_64\lib\site-pac
kages\ipykernel_launcher.py:3: DataConversionWarning: A column-vector y was pas
sed when a 1d array was expected. Please 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 unt
il
```

In [59]:
```
knn_y_pred=knn.predict(X_test)
```

In [60]:
```
from sklearn.preprocessing import MinMaxScaler
m=MinMaxScaler()
m_X_train=m.fit_transform(X_train)
m_X_train
```

Out[60]:
```
array([[0.76470588, 0.64824121, 0.        , ..., 0.59463487, 0.20964987,
        0.38333333],
       [0.23529412, 0.64824121, 0.70491803, ..., 0.52309985, 0.06532878,
        0.03333333],
       [0.17647059, 0.30653266, 0.67213115, ..., 0.51266766, 0.0704526 ,
        0.41666667],
       ...,
       [0.58823529, 0.50753769, 0.70491803, ..., 0.67958271, 0.45175064,
        0.28333333],
       [0.        , 0.70854271, 0.        , ..., 0.6318927 , 0.05422716,
        0.13333333],
       [0.        , 0.6281407 , 0.78688525, ..., 0.33532042, 0.07856533,
        0.        ]])
```

In [61]:
```
m_X_test=m.transform(X_test)
m_X_test
```

Out[61]:
```
array([[0.35294118, 0.49246231, 0.47540984, ..., 0.50670641, 0.15029889,
        0.36666667],
       [0.11764706, 0.56281407, 0.6147541 , ..., 0.53204173, 0.02988898,
        0.        ],
       [0.11764706, 0.54271357, 0.52459016, ..., 0.45901639, 0.03415884,
        0.        ],
       ...,
       [0.35294118, 0.61809045, 0.59016393, ..., 0.50074516, 0.27967549,
        0.21666667],
       [0.17647059, 0.3919598 , 0.40983607, ..., 0.46199702, 0.07258753,
        0.08333333],
       [0.17647059, 0.53266332, 0.59016393, ..., 0.38450075, 0.05508113,
        0.1       ]])
```

In [62]:
```
m_knn=KNeighborsClassifier()
m_knn=m_knn.fit(m_X_train,y_train)
```

```
C:\Program Files (x86)\Microsoft Visual Studio\Shared\Anaconda3_64\lib\site-pac
kages\ipykernel_launcher.py:2: DataConversionWarning: A column-vector y was pas
sed when a 1d array was expected. Please change the shape of y to (n_samples,
), for example using ravel().
```

In [63]:
```
m_y_pred=m_knn.predict(m_X_test)
m_y_pred
```

Out[63]:
```
array([0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0,
       0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1,
       0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,
       0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0,
       0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0], dtype=int64)
```

Classification Metrics

Accuracy

In [64]:
```
model = LogisticRegression(random_state=0, solver='lbfgs',multi_class='multinomia
model.fit(X_train,y_train)
prediction=model.predict(X_test)
print('The accuracy of the Logistic Regression is', metrics.accuracy_score(y_test
```

```
The accuracy of the Logistic Regression is 0.6770833333333334

C:\Program Files (x86)\Microsoft Visual Studio\Shared\Anaconda3_64\lib\site-pac
kages\sklearn\utils\validation.py:578: DataConversionWarning: A column-vector y
was passed when a 1d array was expected. Please change the shape of y to (n_sam
ples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Precision

In [65]:
```
print(precision_score(y_test,m_y_pred))
```

```
0.5522388059701493
```

Recall

In [66]:
```
print(recall_score(y_test,m_y_pred))
```

```
0.5362318840579711
```

AUC Scores

In [67]:
```
knn_auc=print(roc_auc_score(y_test,m_y_pred))
knn_auc
```

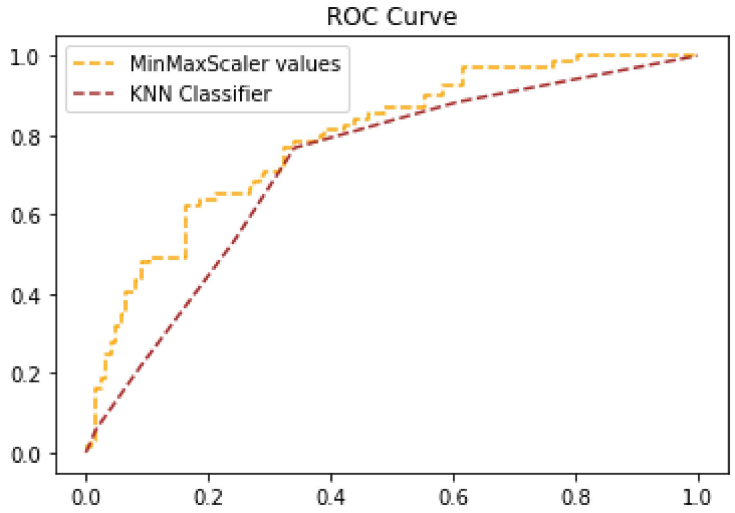
```
0.6461647225167905
```

Step-9. [Update ROC Curve]

In [68]:
```
pred_prob2=m_knn.predict_proba(m_X_test)
```

In [69]:
```
from sklearn.metrics import roc_curve
fpr2,tpr2,thresh2=roc_curve(y_test,pred_prob2[:,1],pos_label=1)
```

In [70]:
```
import matplotlib.pyplot as plt
plt.plot(fpr1,tpr1,linestyle='--',color='orange',label='MinMaxScaler values')
plt.plot(fpr2,tpr2,linestyle='--',color='brown',label='KNN Classifier')
plt.title('ROC Curve')
plt.legend(loc='best')
plt.savefig('ROC',dpi=300)
plt.show()
```



Step-10. [Regularization]

In [71]:
```
from sklearn.linear_model import LogisticRegressionCV
model1=LogisticRegressionCV(Cs=10,cv=4,penalty='l1',solver='liblinear')
model2=LogisticRegressionCV(Cs=10,cv=4,penalty='l2')
```

In [72]:
```
model1.fit(mm_X_train,y_train)
model2.fit(mm_X_train,y_train)
```

```
C:\Program Files (x86)\Microsoft Visual Studio\Shared\Anaconda3_64\lib\site-pac
kages\sklearn\utils\validation.py:578: DataConversionWarning: A column-vector y
was passed when a 1d array was expected. Please change the shape of y to (n_sam
ples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Out[72]:
```
LogisticRegressionCV(Cs=10, class_weight=None, cv=4, dual=False,
           fit_intercept=True, intercept_scaling=1.0, max_iter=100,
           multi_class='ovr', n_jobs=1, penalty='l2', random_state=None,
           refit=True, scoring=None, solver='lbfgs', tol=0.0001, verbose=0)
```

In [74]:
```
rg_y_pred1 = model1.predict(mm_X_test)
rg_y_pred2 = model2.predict(mm_X_test)
```

AUC SCORE OF L1

In [75]:
```
from sklearn.metrics import roc_auc_score
l1_auc = roc_auc_score(y_test, rg_y_pred1)
l1_auc = (' LOR L1 MINMAX AUC', l1_auc)
l1_auc
```

Out[75]: (' LOR L1 MINMAX AUC', 0.694591728525981)

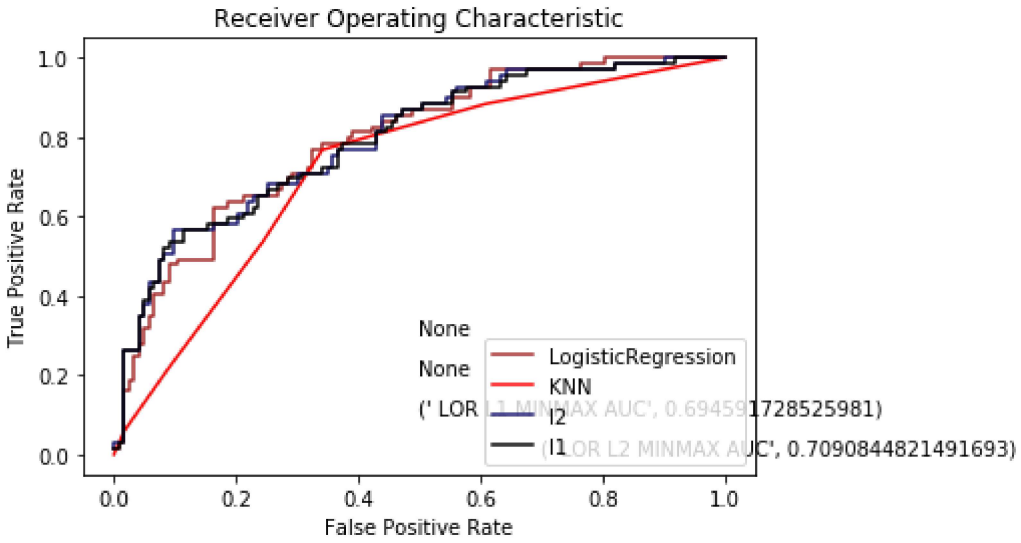
AUC SCORE OF L2

In [76]:
```
from sklearn.metrics import roc_auc_score
l2_auc = roc_auc_score(y_test, rg_y_pred2)
l2_auc = (' LOR L2 MINMAX AUC', l2_auc)
l2_auc
```

Out[76]: (' LOR L2 MINMAX AUC', 0.7090844821491693)

STEP 12 : UPDATE ROC CURVE

In [77]:
```
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='brown', label='LogisticRegression')
plt.plot(fpr1, tbr1, linestyle='-', color='red', label='KNN')
plt.plot(fpr3, tbr3, linestyle='-', color='midnightblue', label='l2')
plt.plot(fpr2, tbr2, linestyle='-', color='black', label='l1')
plt.annotate(xy=[0.5,0.3],s= auc_ss)
plt.annotate(xy=[0.5,0.2],s= knn_auc)
plt.annotate(xy=[0.5,0.1],s= l1_auc)
plt.annotate(xy=[0.7,0],s= l2_auc)
plt.title('Receiver Operating Characteristic')
plt.legend(loc = 'best')
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```



In [ ]:
```
```