In [1]:
```
import pandas as pd
import pickle
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
a=pd.read_csv("C:\\Users\\reshma_koduri\\OneDrive\\Documents\\nba_logreg.csv")
```

In [3]:
```
a
```

Out[3]:

	Name	GP	MIN	PTS	FGM	FGA	FG%	3P Made	3PA	3P%	...	FTA	FT%	OREB	DREB	R
0	Brandon Ingram	36	27.4	7.4	2.6	7.6	34.7	0.5	2.1	25.0	...	2.3	69.9	0.7	3.4	
1	Andrew Harrison	35	26.9	7.2	2.0	6.7	29.6	0.7	2.8	23.5	...	3.4	76.5	0.5	2.0	
2	JaKarr Sampson	74	15.3	5.2	2.0	4.7	42.2	0.4	1.7	24.4	...	1.3	67.0	0.5	1.7	
3	Malik Sealy	58	11.6	5.7	2.3	5.5	42.6	0.1	0.5	22.6	...	1.3	68.9	1.0	0.9	
4	Matt Geiger	48	11.5	4.5	1.6	3.0	52.4	0.0	0.1	0.0	...	1.9	67.4	1.0	1.5	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1335	Chris Smith	80	15.8	4.3	1.6	3.6	43.3	0.0	0.2	14.3	...	1.5	79.2	0.4	0.8	
1336	Brent Price	68	12.6	3.9	1.5	4.1	35.8	0.1	0.7	16.7	...	1.0	79.4	0.4	1.1	
1337	Marlon Maxey	43	12.1	5.4	2.2	3.9	55.0	0.0	0.0	0.0	...	1.6	64.3	1.5	2.3	
1338	Litterial Green	52	12.0	4.5	1.7	3.8	43.9	0.0	0.2	10.0	...	1.8	62.5	0.2	0.4	
1339	Jon Barry	47	11.7	4.4	1.6	4.4	36.9	0.4	1.3	33.3	...	1.0	67.3	0.2	0.7	

1340 rows × 21 columns

In [4]:
```
a.head(10)
```

Out[4]:

	Name	GP	MIN	PTS	FGM	FGA	FG%	3P Made	3PA	3P%	...	FTA	FT%	OREB	DREB	REB
0	Brandon Ingram	36	27.4	7.4	2.6	7.6	34.7	0.5	2.1	25.0	...	2.3	69.9	0.7	3.4	4.1
1	Andrew Harrison	35	26.9	7.2	2.0	6.7	29.6	0.7	2.8	23.5	...	3.4	76.5	0.5	2.0	2.4
2	JaKarr Sampson	74	15.3	5.2	2.0	4.7	42.2	0.4	1.7	24.4	...	1.3	67.0	0.5	1.7	2.2

	Name	GP	MIN	PTS	FGM	FGA	FG%	3P Made	3PA	3P%	...	FTA	FT%	OREB	DREB	REB
3	Malik Sealy	58	11.6	5.7	2.3	5.5	42.6	0.1	0.5	22.6	...	1.3	68.9	1.0	0.9	1.9
4	Matt Geiger	48	11.5	4.5	1.6	3.0	52.4	0.0	0.1	0.0	...	1.9	67.4	1.0	1.5	2.5
5	Tony Bennett	75	11.4	3.7	1.5	3.5	42.3	0.3	1.1	32.5	...	0.5	73.2	0.2	0.7	0.8
6	Don MacLean	62	10.9	6.6	2.5	5.8	43.5	0.0	0.1	50.0	...	1.8	81.1	0.5	1.4	2.0
7	Tracy Murray	48	10.3	5.7	2.3	5.4	41.5	0.4	1.5	30.0	...	0.8	87.5	0.8	0.9	1.7
8	Duane Cooper	65	9.9	2.4	1.0	2.4	39.2	0.1	0.5	23.3	...	0.5	71.4	0.2	0.6	0.8
9	Dave Johnson	42	8.5	3.7	1.4	3.5	38.3	0.1	0.3	21.4	...	1.4	67.8	0.4	0.7	1.1

10 rows × 21 columns

In [5]:
```
a.tail()
```

Out[5]:

	Name	GP	MIN	PTS	FGM	FGA	FG%	3P Made	3PA	3P%	...	FTA	FT%	OREB	DREB	REI
1335	Chris Smith	80	15.8	4.3	1.6	3.6	43.3	0.0	0.2	14.3	...	1.5	79.2	0.4	0.8	1.2
1336	Brent Price	68	12.6	3.9	1.5	4.1	35.8	0.1	0.7	16.7	...	1.0	79.4	0.4	1.1	1.5
1337	Marlon Maxey	43	12.1	5.4	2.2	3.9	55.0	0.0	0.0	0.0	...	1.6	64.3	1.5	2.3	3.8
1338	Litterial Green	52	12.0	4.5	1.7	3.8	43.9	0.0	0.2	10.0	...	1.8	62.5	0.2	0.4	0.7
1339	Jon Barry	47	11.7	4.4	1.6	4.4	36.9	0.4	1.3	33.3	...	1.0	67.3	0.2	0.7	0.9

5 rows × 21 columns

In [6]:
```
a.describe()
```

Out[6]:

	GP	MIN	PTS	FGM	FGA	FG%	3P Made	1
count	1340.000000	1340.000000	1340.000000	1340.000000	1340.000000	1340.000000	1340.000000	
mean	60.414179	17.624627	6.801493	2.629104	5.885299	44.169403	0.247612	
std	17.433992	8.307964	4.357545	1.683555	3.593488	6.137679	0.383688	
min	11.000000	3.100000	0.700000	0.300000	0.800000	23.800000	0.000000	
25%	47.000000	10.875000	3.700000	1.400000	3.300000	40.200000	0.000000	

	GP	MIN	PTS	FGM	FGA	FG%	3P Made
50%	63.000000	16.100000	5.550000	2.100000	4.800000	44.100000	0.100000
75%	77.000000	22.900000	8.800000	3.400000	7.500000	47.900000	0.400000
max	82.000000	40.900000	28.200000	10.200000	19.800000	73.700000	2.300000

In [7]:
```
a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1340 entries, 0 to 1339
Data columns (total 21 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   Name         1340 non-null   object 
 1   GP           1340 non-null   int64  
 2   MIN          1340 non-null   float64
 3   PTS          1340 non-null   float64
 4   FGM          1340 non-null   float64
 5   FGA          1340 non-null   float64
 6   FG%          1340 non-null   float64
 7   3P Made      1340 non-null   float64
 8   3PA          1340 non-null   float64
 9   3P%          1329 non-null   float64
 10  FTM          1340 non-null   float64
 11  FTA          1340 non-null   float64
 12  FT%          1340 non-null   float64
 13  OREB         1340 non-null   float64
 14  DREB         1340 non-null   float64
 15  REB          1340 non-null   float64
 16  AST          1340 non-null   float64
 17  STL          1340 non-null   float64
 18  BLK          1340 non-null   float64
 19  TOV          1340 non-null   float64
 20  TARGET_5Yrs  1340 non-null   float64
dtypes: float64(19), int64(1), object(1)
memory usage: 220.0+ KB
```

In [8]:
```
a.shape
```

Out[8]:
```
(1340, 21)
```

In [9]:
```
list(a)
```

Out[9]:
```
['Name',
 'GP',
 'MIN',
 'PTS',
 'FGM',
 'FGA',
 'FG%',
 '3P Made',
 '3PA',
 '3P%',
 'FTM',
 'FTA',
 'FT%',
 'OREB',
 'DREB',
```

```
 'REB',
 'AST',
 'STL',
 'BLK',
 'TOV',
 'TARGET_5Yrs']
```

In [10]:
```
a.isna().sum()
```

Out[10]:
```
Name           0
GP             0
MIN            0
PTS            0
FGM            0
FGA            0
FG%            0
3P Made        0
3PA            0
3P%           11
FTM            0
FTA            0
FT%            0
OREB           0
DREB           0
REB            0
AST            0
STL            0
BLK            0
TOV            0
TARGET_5Yrs    0
dtype: int64
```

In [11]:
```
a.fillna(35,inplace=True)
```

In [12]:
```
a.isna().sum()
```

Out[12]:
```
Name           0
GP             0
MIN            0
PTS            0
FGM            0
FGA            0
FG%            0
3P Made        0
3PA            0
3P%            0
FTM            0
FTA            0
FT%            0
OREB           0
DREB           0
REB            0
AST            0
STL            0
BLK            0
TOV            0
TARGET_5Yrs    0
dtype: int64
```

In [13]:
```
b=a.drop(['Name'],axis=1)
b
```

Out[13]:

	GP	MIN	PTS	FGM	FGA	FG%	3P Made	3PA	3P%	FTM	FTA	FT%	OREB	DREB	REB	AST
0	36	27.4	7.4	2.6	7.6	34.7	0.5	2.1	25.0	1.6	2.3	69.9	0.7	3.4	4.1	1.9
1	35	26.9	7.2	2.0	6.7	29.6	0.7	2.8	23.5	2.6	3.4	76.5	0.5	2.0	2.4	3.7
2	74	15.3	5.2	2.0	4.7	42.2	0.4	1.7	24.4	0.9	1.3	67.0	0.5	1.7	2.2	1.0
3	58	11.6	5.7	2.3	5.5	42.6	0.1	0.5	22.6	0.9	1.3	68.9	1.0	0.9	1.9	0.8
4	48	11.5	4.5	1.6	3.0	52.4	0.0	0.1	0.0	1.3	1.9	67.4	1.0	1.5	2.5	0.3
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1335	80	15.8	4.3	1.6	3.6	43.3	0.0	0.2	14.3	1.2	1.5	79.2	0.4	0.8	1.2	2.5
1336	68	12.6	3.9	1.5	4.1	35.8	0.1	0.7	16.7	0.8	1.0	79.4	0.4	1.1	1.5	2.3
1337	43	12.1	5.4	2.2	3.9	55.0	0.0	0.0	0.0	1.0	1.6	64.3	1.5	2.3	3.8	0.3
1338	52	12.0	4.5	1.7	3.8	43.9	0.0	0.2	10.0	1.2	1.8	62.5	0.2	0.4	0.7	2.2
1339	47	11.7	4.4	1.6	4.4	36.9	0.4	1.3	33.3	0.7	1.0	67.3	0.2	0.7	0.9	1.4

1340 rows × 20 columns

In [14]:
```
b.groupby(['TARGET_5Yrs']).count()
```

Out[14]:

TARGET_5Yrs	GP	MIN	PTS	FGM	FGA	FG%	3P Made	3PA	3P%	FTM	FTA	FT%	OREB	DREB	F
0.0	509	509	509	509	509	509	509	509	509	509	509	509	509	509	
1.0	831	831	831	831	831	831	831	831	831	831	831	831	831	831	

In [15]:
```
c=pd.get_dummies(b,dtype=int)
c
```

Out[15]:

	GP	MIN	PTS	FGM	FGA	FG%	3P Made	3PA	3P%	FTM	FTA	FT%	OREB	DREB	REB	AST
0	36	27.4	7.4	2.6	7.6	34.7	0.5	2.1	25.0	1.6	2.3	69.9	0.7	3.4	4.1	1.9
1	35	26.9	7.2	2.0	6.7	29.6	0.7	2.8	23.5	2.6	3.4	76.5	0.5	2.0	2.4	3.7
2	74	15.3	5.2	2.0	4.7	42.2	0.4	1.7	24.4	0.9	1.3	67.0	0.5	1.7	2.2	1.0
3	58	11.6	5.7	2.3	5.5	42.6	0.1	0.5	22.6	0.9	1.3	68.9	1.0	0.9	1.9	0.8
4	48	11.5	4.5	1.6	3.0	52.4	0.0	0.1	0.0	1.3	1.9	67.4	1.0	1.5	2.5	0.3
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1335	80	15.8	4.3	1.6	3.6	43.3	0.0	0.2	14.3	1.2	1.5	79.2	0.4	0.8	1.2	2.5
1336	68	12.6	3.9	1.5	4.1	35.8	0.1	0.7	16.7	0.8	1.0	79.4	0.4	1.1	1.5	2.3
1337	43	12.1	5.4	2.2	3.9	55.0	0.0	0.0	0.0	1.0	1.6	64.3	1.5	2.3	3.8	0.3

	GP	MIN	PTS	FGM	FGA	FG%	3P Made	3PA	3P%	FTM	FTA	FT%	OREB	DREB	REB	AST
1338	52	12.0	4.5	1.7	3.8	43.9	0.0	0.2	10.0	1.2	1.8	62.5	0.2	0.4	0.7	2.2
1339	47	11.7	4.4	1.6	4.4	36.9	0.4	1.3	33.3	0.7	1.0	67.3	0.2	0.7	0.9	1.4

1340 rows × 20 columns

In [16]:
```
y=c['TARGET_5Yrs']
y
```

Out[16]:
```
0       0.0
1       0.0
2       0.0
3       1.0
4       1.0
       ...
1335    0.0
1336    1.0
1337    0.0
1338    1.0
1339    1.0
Name: TARGET_5Yrs, Length: 1340, dtype: float64
```

In [17]:
```
x=c.drop(['TARGET_5Yrs'],axis=1)
x
```

Out[17]:

	GP	MIN	PTS	FGM	FGA	FG%	3P Made	3PA	3P%	FTM	FTA	FT%	OREB	DREB	REB	AST
0	36	27.4	7.4	2.6	7.6	34.7	0.5	2.1	25.0	1.6	2.3	69.9	0.7	3.4	4.1	1.9
1	35	26.9	7.2	2.0	6.7	29.6	0.7	2.8	23.5	2.6	3.4	76.5	0.5	2.0	2.4	3.7
2	74	15.3	5.2	2.0	4.7	42.2	0.4	1.7	24.4	0.9	1.3	67.0	0.5	1.7	2.2	1.0
3	58	11.6	5.7	2.3	5.5	42.6	0.1	0.5	22.6	0.9	1.3	68.9	1.0	0.9	1.9	0.8
4	48	11.5	4.5	1.6	3.0	52.4	0.0	0.1	0.0	1.3	1.9	67.4	1.0	1.5	2.5	0.3
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1335	80	15.8	4.3	1.6	3.6	43.3	0.0	0.2	14.3	1.2	1.5	79.2	0.4	0.8	1.2	2.5
1336	68	12.6	3.9	1.5	4.1	35.8	0.1	0.7	16.7	0.8	1.0	79.4	0.4	1.1	1.5	2.3
1337	43	12.1	5.4	2.2	3.9	55.0	0.0	0.0	0.0	1.0	1.6	64.3	1.5	2.3	3.8	0.3
1338	52	12.0	4.5	1.7	3.8	43.9	0.0	0.2	10.0	1.2	1.8	62.5	0.2	0.4	0.7	2.2
1339	47	11.7	4.4	1.6	4.4	36.9	0.4	1.3	33.3	0.7	1.0	67.3	0.2	0.7	0.9	1.4

1340 rows × 19 columns

In [18]:
```
from sklearn.model_selection import train_test_split
(x_train,x_test,y_train,y_test)=train_test_split(x,y,test_size=0.35,random_state=42)
```

In [19]:
```
from sklearn.linear_model import LogisticRegression
reg=LogisticRegression()
reg.fit(x_train,y_train)
```

Out[19]:
```
LogisticRegression()
```

In [20]:
```
ypred=reg.predict(x_test)
ypred
```

Out[20]:
```
array([0., 0., 1., 0., 1., 1., 1., 0., 0., 1., 1., 1., 1., 0., 1., 1., 1.,
       1., 1., 1., 0., 1., 1., 1., 0., 0., 0., 1., 0., 0., 1., 1., 0., 0.,
       1., 0., 1., 1., 1., 1., 1., 0., 1., 1., 1., 0., 0., 0., 1., 1., 1.,
       1., 0., 1., 0., 1., 0., 0., 1., 0., 1., 0., 1., 1., 1., 1., 1., 1.,
       1., 0., 1., 0., 1., 1., 1., 1., 1., 1., 1., 0., 1., 0., 1., 1., 1.,
       1., 0., 1., 1., 0., 1., 1., 0., 0., 1., 1., 1., 1., 1., 0., 1., 1.,
       1., 1., 1., 0., 1., 0., 1., 0., 0., 0., 0., 1., 1., 0., 1., 1., 0.,
       0., 1., 1., 1., 1., 1., 1., 0., 0., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 0., 1., 1., 1., 1., 0., 1., 1., 1., 1., 0., 0.,
       0., 1., 1., 0., 0., 1., 1., 0., 1., 0., 1., 0., 1., 0., 1., 1., 1.,
       1., 1., 0., 1., 0., 1., 1., 1., 0., 1., 1., 0., 1., 1., 1., 0., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 0., 1., 1.,
       1., 1., 1., 1., 1., 1., 0., 1., 1., 1., 1., 1., 1., 0., 1., 1., 1.,
       1., 1., 1., 1., 1., 0., 1., 1., 1., 0., 1., 0., 1., 1., 1., 0., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 0., 1., 0., 1., 1., 0., 1., 0., 0.,
       1., 1., 1., 1., 1., 0., 1., 1., 0., 1., 1., 1., 1., 1., 0., 1., 1.,
       0., 1., 0., 0., 1., 0., 1., 1., 0., 1., 1., 1., 1., 1., 0., 1., 0.,
       1., 0., 1., 1., 1., 1., 1., 0., 0., 0., 1., 1., 1., 1., 0., 1., 0.,
       1., 1., 1., 1., 1., 1., 1., 0., 1., 1., 1., 0., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 0., 0., 1., 1., 1., 1., 1., 1., 0., 0., 0., 1., 1.,
       0., 1., 0., 0., 1., 1., 1., 1., 1., 0., 1., 1., 0., 0., 0., 1., 1.,
       0., 1., 1., 1., 1., 1., 1., 1., 1., 0., 0., 0., 1., 0., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 0., 0., 0., 1., 0., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 1., 1., 1.,
       1., 1., 0., 0., 1., 1., 1., 1., 0., 1., 0., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 0., 1., 1., 0., 0., 1., 1., 1., 0., 1.,
       0., 1., 0., 1., 1., 1., 1., 1., 1., 0., 1., 0., 1., 1., 1., 0., 0.,
       0., 0., 1., 1., 1., 0., 1., 1., 0., 0.])
```

In [21]:
```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test,ypred)
```

Out[21]:
```
array([[ 96,  88],
       [ 44, 241]], dtype=int64)
```

In [22]:
```
from sklearn.metrics import accuracy_score
accuracy_score(ypred,y_test)
```

Out[22]:
```
0.7185501066098081
```

In [23]:
```
import seaborn as sb
import matplotlib.pyplot as plt
```

In [24]:
```
results=pd.DataFrame(columns=['Actual','Predicted'])
results['Actual']=y_test
results["Predicted"]=ypred
results=results.reset_index()
```

```
results['Id']=results.index
results.head(5)
```

Out[24]:

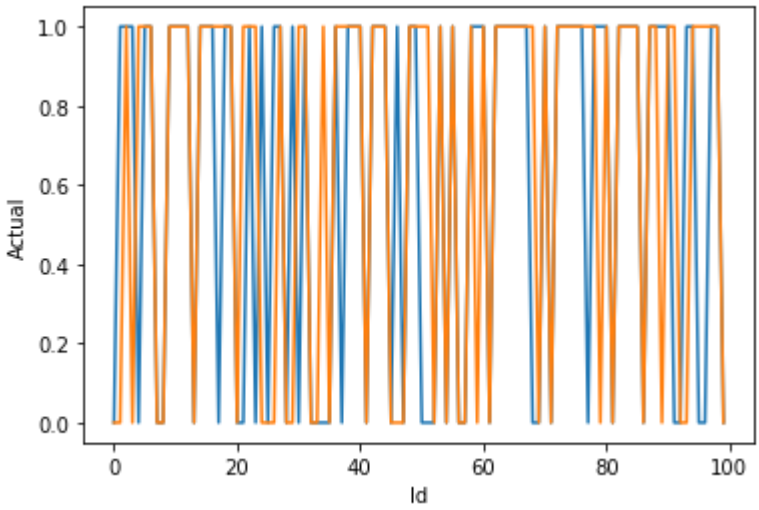
	index	Actual	Predicted	Id
0	394	0.0	0.0	0
1	881	1.0	0.0	1
2	358	1.0	1.0	2
3	367	1.0	0.0	3
4	259	0.0	1.0	4

In [25]:

```
sb.lineplot(x='Id',y='Actual',data=results.head(100))
sb.lineplot(x='Id',y='Predicted',data=results.head(100))
plt.plot()
```

Out[25]:

```
[]
```



In [26]:

```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier
reg=RandomForestClassifier()
n_estimators=[25,50,75,100,125,150,175,200]
criterion=['gini','entropy']
max_depth=[3,5,10]
parameters={'n_estimators': n_estimators,'criterion':criterion,'max_depth':max_depth
rfc_reg = GridSearchCV(reg, parameters)
rfc_reg.fit(x_train,y_train)
```

Out[26]:

```
GridSearchCV(estimator=RandomForestClassifier(),
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [3, 5, 10],
                         'n_estimators': [25, 50, 75, 100, 125, 150, 175, 200]})
```

In [27]:

```
rfc_reg.best_params_
```

Out[27]:

```
{'criterion': 'gini', 'max_depth': 5, 'n_estimators': 25}
```

In [31]:

```
reg=RandomForestClassifier(n_estimators=25,criterion='gini',max_depth=5)
reg.fit(x_train,y_train)
```

Out[31]:
```
RandomForestClassifier(max_depth=5, n_estimators=25)
```

In [32]:
```
ypred=reg.predict(x_test)
ypred
```

Out[32]:
```
array([0., 0., 1., 0., 0., 1., 1., 0., 0., 1., 1., 1., 1., 0., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 0., 1., 0., 1., 1., 0., 1., 1., 0., 0.,
       1., 0., 0., 1., 1., 0., 1., 0., 1., 1., 1., 0., 0., 0., 0., 1., 1.,
       1., 1., 1., 0., 1., 0., 1., 1., 0., 1., 0., 1., 1., 1., 1., 1., 1.,
       1., 0., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 0., 1., 1., 1.,
       1., 1., 1., 1., 0., 1., 1., 1., 0., 1., 1., 1., 1., 1., 0., 1., 1.,
       1., 1., 1., 0., 1., 0., 1., 0., 0., 1., 1., 1., 1., 1., 1., 1., 0.,
       0., 1., 1., 1., 1., 1., 1., 1., 0., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 0., 1., 1., 1., 1., 1., 1., 0., 1., 0., 0., 1.,
       0., 1., 1., 1., 0., 1., 1., 0., 1., 0., 1., 0., 1., 1., 1., 1., 1.,
       1., 0., 0., 1., 0., 1., 1., 1., 1., 0., 1., 0., 1., 1., 1., 0., 1.,
       1., 0., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 0., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 0., 1., 0., 0., 0., 0.,
       0., 1., 0., 1., 1., 1., 1., 1., 1., 0., 1., 0., 1., 1., 1., 1., 1.,
       1., 1., 0., 1., 1., 1., 1., 1., 0., 0., 0., 1., 1., 1., 1., 0., 0.,
       1., 1., 1., 1., 1., 1., 1., 1., 0., 1., 1., 1., 1., 1., 0., 1., 1.,
       0., 1., 0., 0., 1., 0., 1., 1., 0., 1., 1., 1., 0., 1., 0., 0., 1.,
       1., 0., 1., 1., 0., 1., 0., 0., 0., 0., 1., 1., 1., 1., 1., 1., 0.,
       1., 1., 1., 1., 1., 1., 1., 0., 1., 0., 1., 0., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 0., 0., 1., 1., 1., 1., 1., 1., 1., 0., 1., 1., 1.,
       0., 1., 1., 0., 1., 1., 1., 0., 0., 0., 1., 1., 1., 1., 0., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 0., 0., 0., 1., 0., 1., 0., 1.,
       1., 1., 1., 1., 1., 1., 0., 0., 0., 0., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 0., 1., 0., 0., 1., 1., 0., 1., 1., 1., 1.,
       1., 1., 1., 0., 1., 1., 1., 1., 1., 0., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 0., 1., 1., 0., 1., 1., 0., 0., 1., 1., 1., 0., 1.,
       0., 1., 0., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 0., 0.,
       1., 0., 1., 1., 1., 0., 1., 1., 1., 0.])
```

In [33]:
```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,ypred)
```

Out[33]:
```
0.6801705756929638
```

In [ ]:
```

```