PROBLEM DEFINITION

In [1]:
```
# The research questions we would work upon are:
#  1.Did PERCEPTION about covid vaccines played any role in cases worldwide?
#  2.Which continent's population was more inclined towards the perception of not taking a vaccine?

# The data we intend to use is based on the perception of the people about Covid Vaccines.
# There has always been a conflict in the population about the vaccines.
# Some had positive perceptions and the others, negative. We have taken into consideration the following scenario:
# The risks of COVID-19 disease are greater than the risks of the vaccine.
# The people those who are in favour of this statement took the vaccines, and those who aren't, did the opposite.
```

DATA DESCRIPTION

In [2]:
```
#To get to the bottom of the situation above, my approach includes usage of two datasets:
#(1) owid-covid-data.csv (renamed to Covid Data New).
#    It contains the data about Covid-19 and its effects on the world (deaths, cases, etc)
#(2) Raw_Data_Vaccine_hesitancy_globally.csv (renamed to Perception)
#    This one contains the data about the public survey held in June of 2021. This specifically contains how people were perceptive about Covid vaccines. Some people had positive pe
```

In [3]:
```
## To import the libraries

import seaborn as sns
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

from collections import Counter
from imblearn.over_sampling import RandomOverSampler

from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split

from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
```

In [4]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [5]:
```
#Importing the datasets.

Perceptions = pd.read_csv('D:\Group Projects 4070\Group Project New\Perception.csv')
Covid=pd.read_csv('Covid Data New.csv')

##Renaming the column for ease of use
Perceptions.rename(columns={'Perception/The risks of Covid-19 disease are greater then the risks of vaccine': 'Perception'}, inplace=True)
```

In [6]:
```
Perceptions.head()
```

Out[6]:

	Country	Perception
0	India	Somewhat agree
1	India	Somewhat agree
2	India	Somewhat agree
3	India	Somewhat agree
4	India	Somewhat agree

In [7]:
```
Covid.head()
```

Out[7]:

	iso_code	continent	Country	date	Years	total_cases	total_deaths	new_cases_smoothed	total_deaths.1	new_deaths	...	hospital_beds_per_thousand	life_expectancy	human_development_index	population	excess_mort
0	BRA	South America	Brazil	26/02/2020	2020	1.0	NaN	NaN	NaN	NaN	...	2.2	75.88	0.765	215313504	
1	BRA	South America	Brazil	27/02/2020	2020	1.0	NaN	NaN	NaN	NaN	...	2.2	75.88	0.765	215313504	
2	BRA	South America	Brazil	28/02/2020	2020	1.0	NaN	NaN	NaN	NaN	...	2.2	75.88	0.765	215313504	
3	BRA	South America	Brazil	29/02/2020	2020	2.0	NaN	NaN	NaN	NaN	...	2.2	75.88	0.765	215313504	
4	BRA	South America	Brazil	01/03/2020	2020	2.0	NaN	NaN	NaN	NaN	...	2.2	75.88	0.765	215313504	

5 rows × 70 columns

FEATURE ENGINEERING

In [8]:
```
## The perception column has 5 Categories
# Strongly agree,
# Somewhat agree,
# Unsure/no opinion,
# Somewhat disagree,
# Strongly disagree.
```

In [11]:
```
## We've removed the 'unsure/nop opinion', as we don't know whether they took the vaccines or not, and their number was less.
```

In [12]:
```
Perceptions=Perceptions[Perceptions['Perception']!='Unsure/no opinion']
```

In [13]:
```
## We've further divided the groups-
#    Strongly Agree + Somewhat Agree = TOOK VACCINES.
#    Strongly Disagree + Somewhat Disagree = DID NOT TAKE VACCINES
```

In [14]:
```
Perceptions['Perception']=Perceptions['Perception'].replace({'Somewhat disagree':'Strongly disagree'})
Perceptions['Perception']=Perceptions['Perception'].replace({'Somewhat agree':'Strongly agree'})
```

In [15]:
```
## Keeping only the necessary columns
Covid_columns=Covid[['continent','Country','date','Years','total_cases','total_deaths','new_cases']]

## Merging the two datasets
## Combining both datasets on common columns.
## Note that They can be joined basis one column 'COUNTRY'.

Perceptions_Covid = pd.merge(Perceptions,Covid_columns, on='Country')

Perceptions_Covid
```

Out[15]:

	Country	Perception	continent	date	Years	total_cases	total_deaths	new_cases
0	India	Strongly agree	Asia	30/01/2020	2020	1.0	NaN	1.0
1	India	Strongly agree	Asia	31/01/2020	2020	1.0	NaN	0.0
2	India	Strongly agree	Asia	01/02/2020	2020	1.0	NaN	0.0
3	India	Strongly agree	Asia	02/02/2020	2020	2.0	NaN	1.0
4	India	Strongly agree	Asia	03/02/2020	2020	3.0	NaN	1.0
...	...	...	...	...	...	...	...	...
23191073	Brazil	Strongly agree	South America	04/02/2023	2023	36867401.0	697365.0	1118.0
23191074	Brazil	Strongly agree	South America	05/02/2023	2023	36867401.0	697365.0	0.0
23191075	Brazil	Strongly agree	South America	06/02/2023	2023	36878774.0	697439.0	11373.0
23191076	Brazil	Strongly agree	South America	07/02/2023	2023	36887991.0	697533.0	9217.0
23191077	Brazil	Strongly agree	South America	08/02/2023	2023	36897683.0	697626.0	9692.0

23191078 rows × 8 columns

In [16]:
```
## Data Mapping - Mapping the data such that
# STRONGLY AGREE = 1
# STRONGLY DISAGREE = 0

Perceptions_Covid['Perception']=Perceptions_Covid['Perception'].map({'Strongly agree':1,'Strongly disagree':0})
Perceptions_Covid
```

Out[16]:

	Country	Perception	continent	date	Years	total_cases	total_deaths	new_cases
0	India	1	Asia	30/01/2020	2020	1.0	NaN	1.0
1	India	1	Asia	31/01/2020	2020	1.0	NaN	0.0
2	India	1	Asia	01/02/2020	2020	1.0	NaN	0.0
3	India	1	Asia	02/02/2020	2020	2.0	NaN	1.0
4	India	1	Asia	03/02/2020	2020	3.0	NaN	1.0
...	...	...	...	...	...	...	...	...
23191073	Brazil	1	South America	04/02/2023	2023	36867401.0	697365.0	1118.0
23191074	Brazil	1	South America	05/02/2023	2023	36867401.0	697365.0	0.0
23191075	Brazil	1	South America	06/02/2023	2023	36878774.0	697439.0	11373.0
23191076	Brazil	1	South America	07/02/2023	2023	36887991.0	697533.0	9217.0
23191077	Brazil	1	South America	08/02/2023	2023	36897683.0	697626.0	9692.0

23191078 rows × 8 columns

In [17]:
```
## Taking care of the NULL VALUES
## Dropping the Null Values

Perceptions_Covid.isnull().sum().sum()
no_null=Perceptions_Covid.dropna()
no_null.isnull().sum().sum()
```

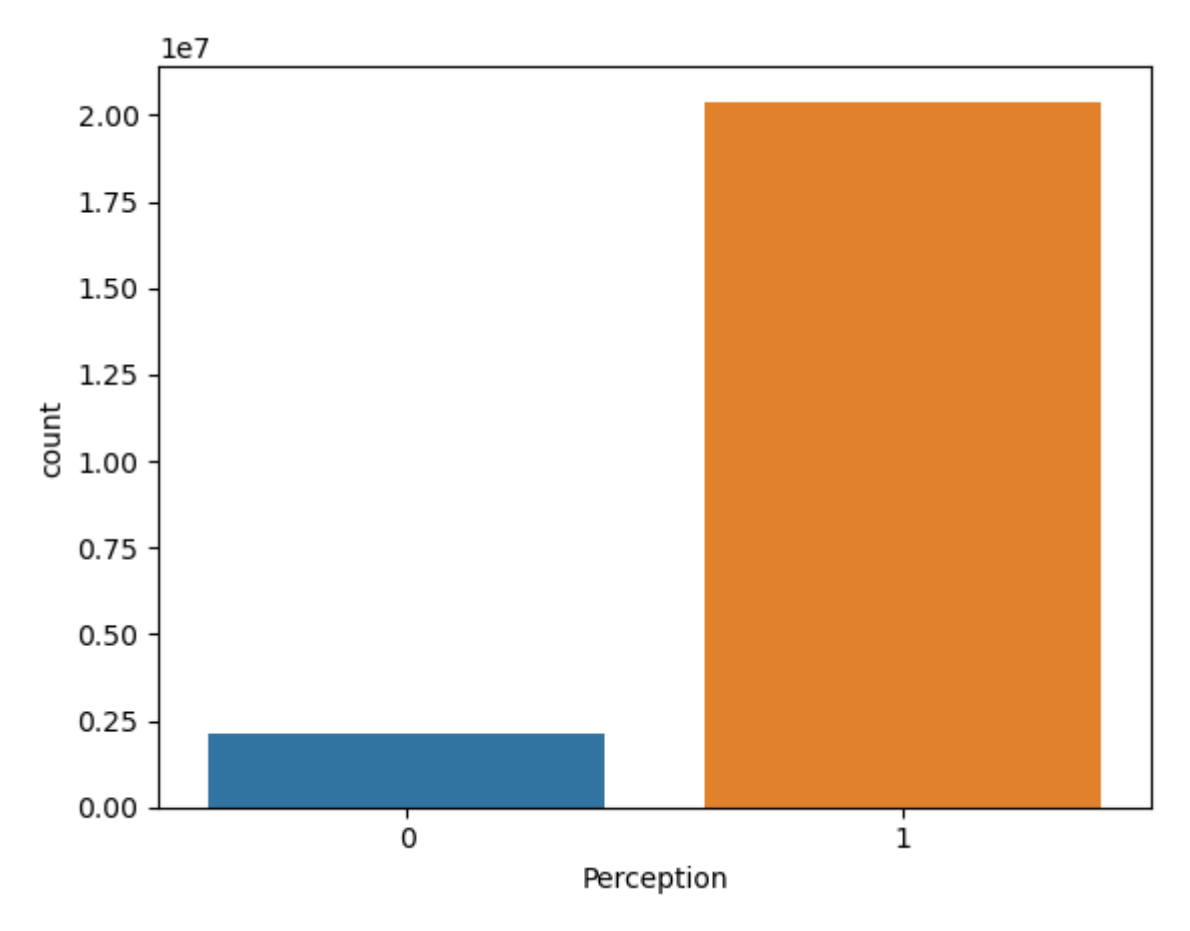
Out[17]:

```
0
```

In [18]:
```
sns.countplot(x='Perception', data=no_null)
```

Out[18]:

```
<Axes: xlabel='Perception', ylabel='count'>
```

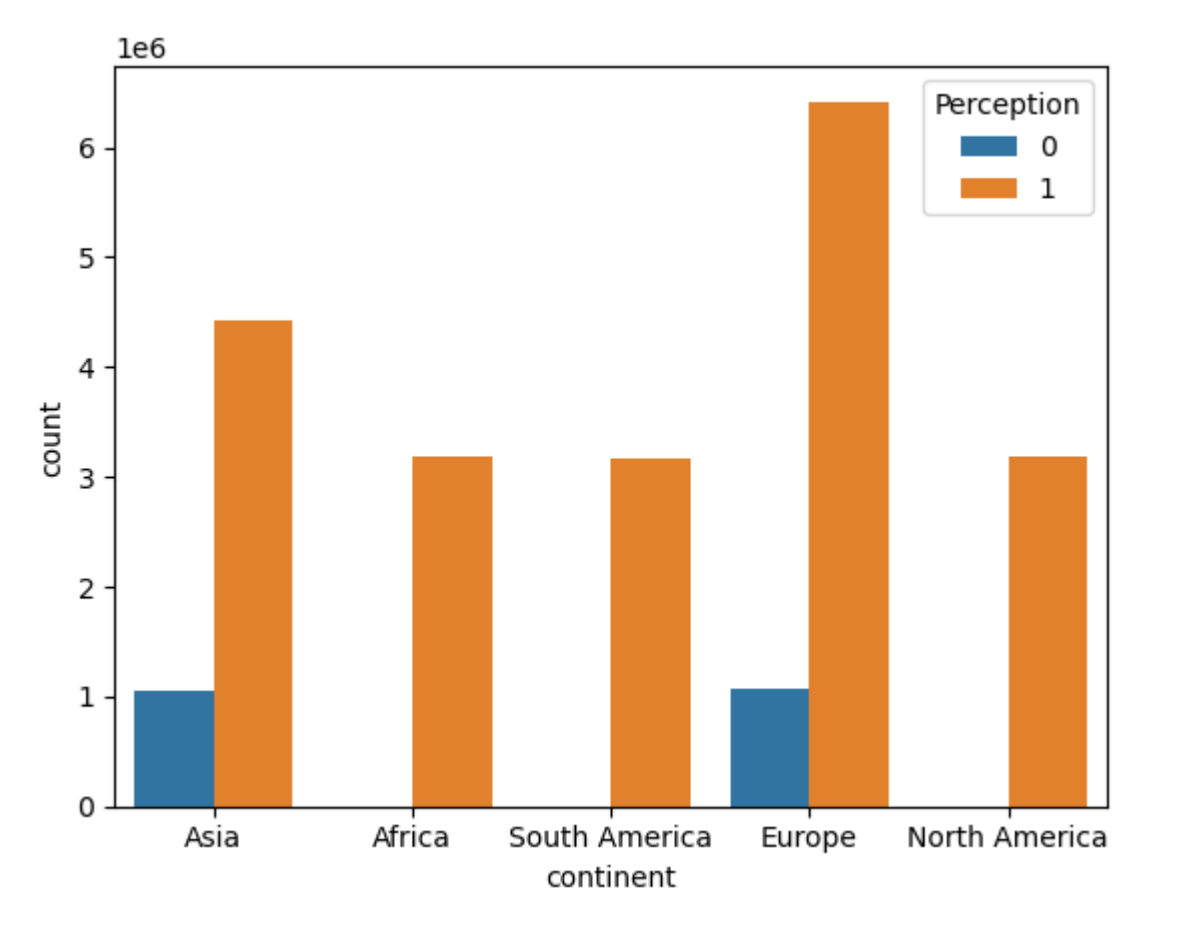


In [19]:
```
# As the above the graph depicts that the majority of people were in favour of TAKING THE VACCINES
# and very less population were against the idea.
```

In [20]:
```
sns.countplot(x='continent',hue='Perception',data=no_null)
```

Out[20]:

```
<Axes: xlabel='continent', ylabel='count'>
```

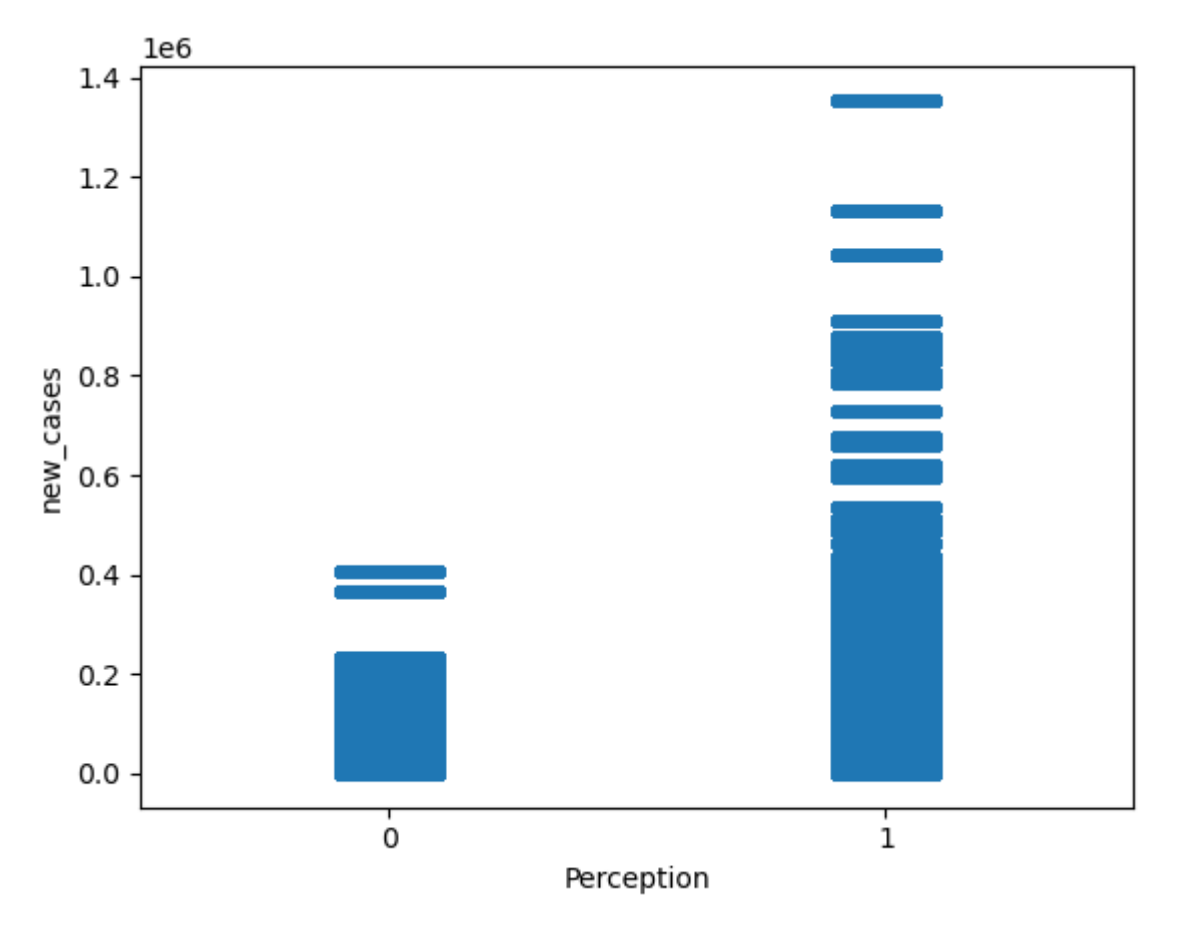


In [21]:
```
# The above graph shows that mostly the group of population not in favour of taking the vaccines is from
# Asia and Europe.
```

In [22]:
```
sns.stripplot(x='Perception',y='new_cases',data=no_null)
```

Out[22]:

```
<Axes: xlabel='Perception', ylabel='new_cases'>
```



In [23]:
```
# With the above graph, it is clear that the people who weren't willing to take vaccines were also leading to the increase
## in newly prevailed cases of Covid. Thereby increasing the cases count
```

In [24]:

DATA MODELLING

In [25]:
```
no_null['Perception'].value_counts()
```

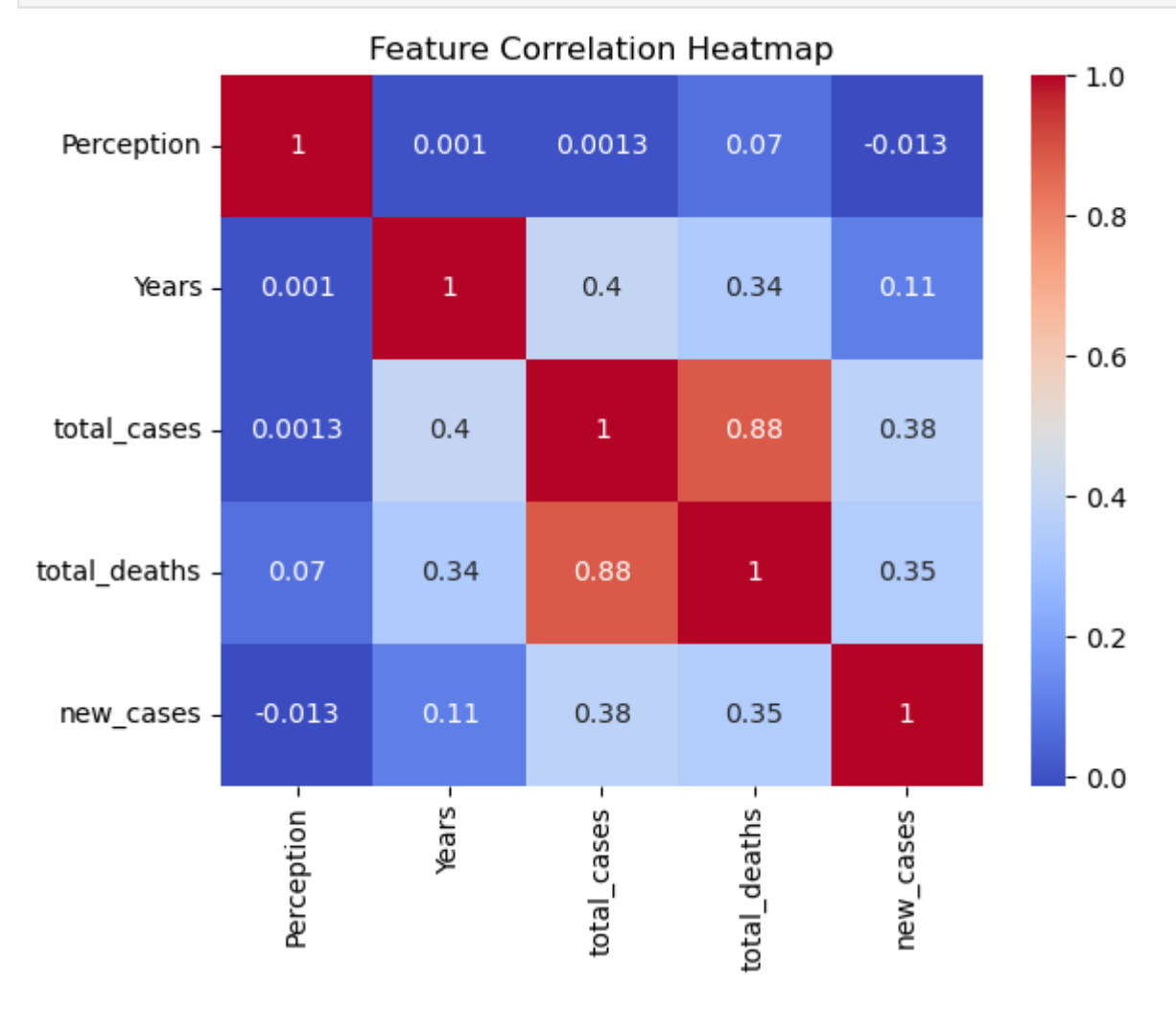
Out[25]:

```
1    20388174
0     2136000
Name: Perception, dtype: int64
```

In [26]:
```
df=no_null
```

In [59]:
```
 # Calculating the correlation amongs the features
    # We can get corelation between features based on values ranging (-1, 1). Near to -1 or 1 shows strong corelation
    # in OPPOSITE directions.
    # Near to zero means lesser corelation.
correlation = df.corr()
```

In [28]:
```
# Visualization to check the features corelation and multicollinearity
sns.heatmap(correlation, annot=True, cmap='coolwarm')
plt.title('Feature Correlation Heatmap')
plt.show()
```



In [55]:
```
## DATA BALANCING

# Our data needs to be balanced because against the hughe values fo 1's, we have less values of 0's.
# We would do the Over Sampling technique. The reason behind this is because Under Sampling technique reduces the
# size of the sampled data as per the lower value.
```

I did both the sampling technique. UNDER & OVER.

I'm only showing the output with Undersampling because doing that also gets us lot of data to utilize.

The oversampling also lead to approximately the same accuracy of the model.

In [31]:
```
X=df #Independent features
y=df['Perception'] # Dependent Features
```

In [32]:
```
# Under Sampling technique for balancing the Data.

from imblearn.under_sampling import RandomUnderSampler
rus = RandomUnderSampler(random_state=0)
X_resampled, y_resampled = rus.fit_resample(X,y)
print(sorted(Counter(y_resampled).items()),y_resampled.shape)
```

```
[(0, 2136000), (1, 2136000)] (4272000,)
```

In [33]:
```
A=X_resampled
```

In [34]:
```
A.drop(['Country','continent','date','Years'], axis=1, inplace=True)
A
```

Out[34]:

	Perception	total_cases	total_deaths	new_cases
0	0	47.0	1.0	29.0
1	0	98.0	1.0	51.0
2	0	192.0	3.0	94.0
3	0	359.0	4.0	167.0
4	0	670.0	9.0	311.0
...	...	...	...	...
4271995	1	27564.0	628.0	454.0
4271996	1	3377792.0	78977.0	19715.0
4271997	1	151881.0	19694.0	1094.0
4271998	1	1768186.0	163970.0	6611.0
4271999	1	228.0	1.0	63.0

4272000 rows × 4 columns

In [35]:
```
B=y_resampled
B
```

Out[35]:

```
0          0
1          0
2          0
3          0
4          0
          ..
4271995    1
4271996    1
4271997    1
4271998    1
4271999    1
Name: Perception, Length: 4272000, dtype: int64
```

In [56]:
```
#  We would apply LOGISTIC REGRESSION because our Depedent Feature is Categorical.
#  We would divide our dataset into 70:30 Ratio.
#   70% - Train Data
#   30%-Test Data
#   Random State = 1 means that whenever the code is run, our data would be split into small but same datasets.
```

In [37]:
```
from sklearn.linear_model import LogisticRegression
classifier=LogisticRegression()
```

In [38]:
```
# Splitting the data into test and train.

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(A, B, test_size=0.3, random_state=1)
```

In [39]:
```
classifier.fit(X_train,y_train)
```

Out[39]:

```
▾ LogisticRegression
LogisticRegression()
```

In [40]:
```
predictions=classifier.predict(X_test)
```

In [41]:
```
from sklearn.metrics import classification_report
classification_report(y_test,predictions)
```

Out[41]:

```
'              precision    recall  f1-score   support\n\n           0       0.66      0.58      0.62    640245\n           1       0.62      0.71      0.66    641355\n\n    accurac
y                           0.64   1281600\n   macro avg       0.64      0.64      0.64   1281600\nweighted avg       0.64      0.64      0.64   1281600\n'
```

In [45]:
```
class_report = classification_report(y_test,predictions)
print("Classification Report:")
print(class_report)
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.66      0.58      0.62    640245
           1       0.62      0.71      0.66    641355

    accuracy                           0.64   1281600
   macro avg       0.64      0.64      0.64   1281600
weighted avg       0.64      0.64      0.64   1281600
```

In [42]:
```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
```

In [43]:
```
#Calculating precision,recall and f1 score
```

In [44]:
```
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
```

In [49]:
```
# ACCURACY   # PRECISION   # RECALL   # F1-SCORE

accuracy = accuracy_score(y_test,predictions)

print('Precision:', 100* precision_score(y_test,predictions))
print('Recall:', 100* recall_score(y_test,predictions))
print('f1-score:', f1_score(y_test,predictions))

print(f'Accuracy: {round(accuracy*100,2)}%')
```

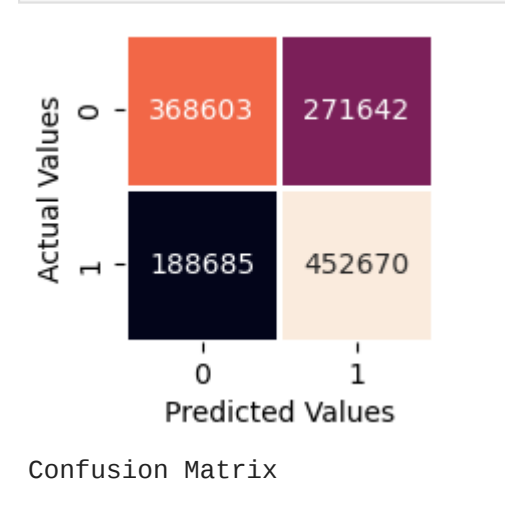
```
Precision: 62.4965484487348
Recall: 70.5802558645368
f1-score: 0.6629288106104929
Accuracy: 64.08%
```

In [50]:
```
from sklearn import metrics
cm = metrics.confusion_matrix(y_test,predictions)
cm
```

Out[50]:

```
array([[368603, 271642],
       [188685, 452670]], dtype=int64)
```

In [51]:
```
plt.figure(figsize=(2,2))
sns.heatmap(cm, annot=True, fmt = '1.0f', linewidth = 1.0,
          square = True, cbar= False)
plt.ylabel('Actual Values')
plt.xlabel('Predicted Values')
plt.show()
print('Confusion Matrix')
```




```
Confusion Matrix
```

Result

In [52]:
```
# We hereby conclude that the accuracy of the model is 64%. This accuracy is not that bad, yet not so good.
# As our data is purely based on the available data.
# The Sampling technique has been used here because the accuracy without using Random Sampler was 86%.
# Which shows that our model was overfitting. Thus to avoid the situation, we took the path of Sampling.
# Further, the values of No's was much less then values of Yes's, which lead to incorrect assumption.
```

CONCLUSION

In [1]:
```
## ANSWER 1
## Continents Asia and Europe were the ones with people who weren't willing to take the vaccines, as per the data.

## ANSWER 2
## the majority of people were in favour of vaccines and very less were against.

## BONUS,
## The group of population who weren not willing to take vaccines, were not much, but they were also playing a role
## in increasing the number of cases worldwide.
```

Pros

In [53]:
```
# We have a simple data wherein with the use of above approach of Logistic Regression we can come to conclusion that
# the depedent feature provides useful insights and help us understand how the perception impacted the cases.
```

Cons

In [54]:
```
## Our dataset is huge with many missing values, which hamper the predictions at the first place.
## Also there is so much imbalance in our predicting feature.
## By doing Undersampling and Oversampling, we got Accuracy of 64.08% and 64.10%. This shows that our data isn't perfect,
## we wouldn't get better predictions if we try either method.
```

In [ ]: