In [1]:
```
#import all necessary library
import pandas as pd
import string
from nltk.corpus import stopwords
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import cross_val_score,StratifiedKFold,cross_val_
from sklearn.metrics import confusion_matrix,classification_report
from sklearn.naive_bayes import MultinomialNB, BernoulliNB
import seaborn as sns
import plotly.express as px
from nltk.stem import WordNetLemmatizer
from sklearn.feature_extraction.text import TfidfVectorizer
from imblearn.over_sampling import RandomOverSampler
import warnings as w
from sklearn.linear_model import LogisticRegression
w.filterwarnings('ignore')
```

In [2]:
```
#import the dataset
```

In [3]:
```
url = 'https://raw.githubusercontent.com/rashakil-ds/Public-Datasets/main/amaz
```

In [4]:
```
df = pd.read_csv(url)
```

In [5]:
```
df.head()
```

Out[5]:

	reviewText	Positive
0	This is a one of the best apps acording to a b...	1
1	This is a pretty good version of the game for ...	1
2	this is a really cool game. there are a bunch ...	1
3	This is a silly game and can be frustrating, b...	1
4	This is a terrific game on any pad. Hrs of fun...	1

In [6]:
```
df.shape
```

Out[6]:
```
(20000, 2)
```

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20000 entries, 0 to 19999
Data columns (total 2 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   reviewText  20000 non-null  object
 1   Positive    20000 non-null  int64 
dtypes: int64(1), object(1)
memory usage: 312.6+ KB
```

In [8]:
```
df.rename(columns={'Positive':'target' , 'reviewText': 'text'},inplace=True)
```

In [9]:
```
df.head()
```

Out[9]:

	text	target
0	This is a one of the best apps acording to a b...	1
1	This is a pretty good version of the game for ...	1
2	this is a really cool game. there are a bunch ...	1
3	This is a silly game and can be frustrating, b...	1
4	This is a terrific game on any pad. Hrs of fun...	1

Visualisation

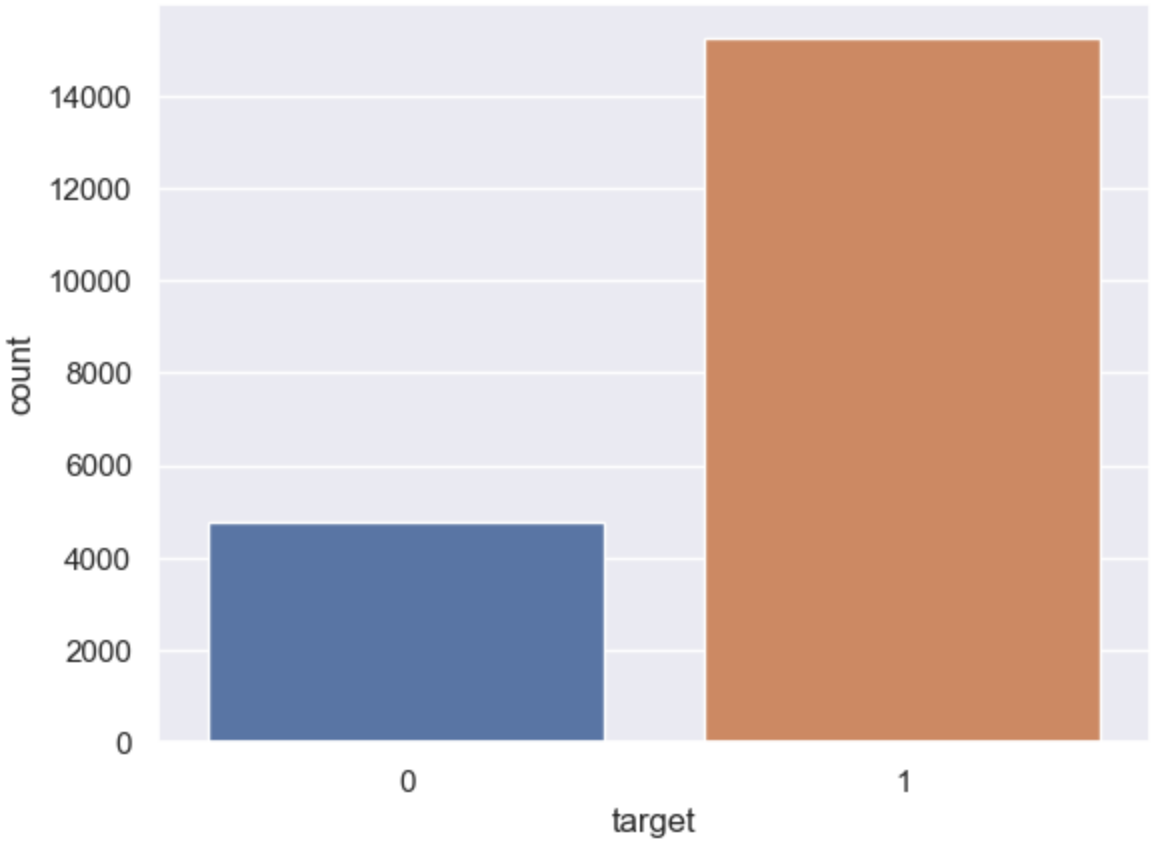
In [10]:
```
df.columns
```

Out[10]:
```
Index(['text', 'target'], dtype='object')
```

In [11]:
```
sns.set()
```

In [12]:
```
sns.countplot(x=df['target'])
```

Out[12]:
```
<AxesSubplot: xlabel='target', ylabel='count'>
```



In [13]:
```
ham_spam = df.target.value_counts()
```

In [14]:
```
value = ham_spam.values
index = ham_spam.index
```

In [15]:
```
ham_spam
```

Out[15]:
```
1    15233
0     4767
Name: target, dtype: int64
```

In [16]:
```
px.pie(df,
       values = value,
       names = index,
       hole = .7)
```

In [17]:
```
df.isnull().sum()
```

Out[17]:
```
text      0
target    0
dtype: int64
```

In [18]:
```
df.duplicated().sum()
```

Out[18]:
```
0
```

In [19]:
```
df.shape
```

Out[19]:
```
(20000, 2)
```

In [20]:
```
tar = df.target.value_counts()
tar
```

Out[20]:
```
1    15233
0     4767
Name: target, dtype: int64
```

In [21]:
```
string.punctuation
```

Out[21]:
```
'!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'
```

In [22]:
```
def text_preprocessing(text):

    remove_punctuation = [word for word in text if word not in string.punctuat
    join_word = ''.join(remove_punctuation)
    split_word = join_word.split()

    stop_word= [ word for word in split_word if word.lower() not in stopwords.
    join_ = ' '.join(stop_word)

    lemmatize_text =WordNetLemmatizer().lemmatize(join_)
    return lemmatize_text
```

In [23]:
```
df['text'] = df['text'].apply(text_preprocessing)
```

In [24]:
```
df.head()
```

Out[24]:

	text	target
0	one best apps acording bunch people agree bomb...	1
1	pretty good version game free LOTS different l...	1
2	really cool game bunch levels find golden eggs...	1
3	silly game frustrating lots fun definitely rec...	1
4	terrific game pad Hrs fun grandkids love Great...	1

In [25]:
```
df['text'][50]
```

Out[25]:
```
'Well Im would call gamer game found addicting Makes nuts enough try fail'
```

In [26]:
```
import numpy as np
```

In [27]:
```
x =TfidfVectorizer().fit_transform(df['text']).toarray()
y=df['target']
```

```
---------------------------------------------------------------------------
MemoryError                               Traceback (most recent call last)
Cell In[27], line 1
----> 1 x =TfidfVectorizer().fit_transform(df['text']).toarray()
      2 y=df['target']

File ~\anaconda3\Lib\site-packages\scipy\sparse\_compressed.py:1051, in _cs_matrix.toarray(self, order, out)
   1049 if out is None and order is None:
   1050     order = self._swap('cf')[0]
-> 1051 out = self._process_toarray_args(order, out)
   1052 if not (out.flags.c_contiguous or out.flags.f_contiguous):
   1053     raise ValueError('Output array must be C or F contiguous')

File ~\anaconda3\Lib\site-packages\scipy\sparse\_base.py:1298, in spmatrix._process_toarray_args(self, order, out)
   1296     return out
   1297 else:
-> 1298     return np.zeros(self.shape, dtype=self.dtype, order=order)

MemoryError: Unable to allocate 3.57 GiB for an array with shape (20000, 23969) and data type float64
```

In [ ]:
```
new_x , new_y =RandomOverSampler(random_state=100).fit_resample(x,y)
```

In [35]:
```
new_x
```

Out[35]:
```
array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]])
```

In [36]:
```
def result(model,new_x,new_y):
    mull = model(alpha=1.0 , fit_prior=True )
    mod = mull.fit(new_x,new_y)
    st = StratifiedKFold(n_splits=6)
    cro = cross_val_score(mod , new_x,new_y , cv = st)
    return cro
```

In [37]:
```
result(MultinomialNB,new_x,new_y)
```

Out[37]:
```
array([0.86983064, 0.8773139 , 0.86963371, 0.89877905, 0.89009257,
       0.85897183])
```

In [42]:
```
new_x
```

Out[42]:
```
array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]])
```

In [43]:
```
new_y
```

Out[43]:
```
0        1
1        1
2        1
3        1
4        1
        ..
30461    0
30462    0
30463    0
30464    0
30465    0
Name: target, Length: 30466, dtype: int64
```

In [52]:
```
from sklearn.model_selection import train_test_split
```

In [53]:
```
xtrain,xtest,ytrain,ytest = train_test_split(new_x,new_y,test_size=.15)
```

In [54]:
```
MultinomialNB = MultinomialNB().fit(xtrain,ytrain)
```

In [56]:
```
MultinomialNB = MultinomialNB.score(xtest , ytest)
```

In [57]:
```
MultinomialNB
```

Out[57]:
```
0.8982494529540481
```

In [62]:
```
LogisticRegression = LogisticRegression().fit(xtrain,ytrain)
```

In [63]:
```
LogisticRegression = LogisticRegression.score(xtest , ytest)
```

In [66]:
```
LogisticRegression
```

Out[66]:
```
0.9072210065645514
```

In [ ]:
```
#LogisticRegression given better score
```