AMAZON REVIEWS (NLP)

Read Csv

```
import pandas as pd
import numpy as np
import seaborn as sns

df = pd.read_csv('Amazon_Unlocked_Mobile.csv')

df = df.sample(frac=0.1, random_state=10)

df.head()
```

```
df.tail()
```

	Product Name	Brand Name	Price	Rating
30001	Apple iPhone 5c 32GB (Blue) - AT&T	Apple	274.95	5

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 41384 entries, 394349 to 109303
Data columns (total 6 columns):
Product Name    41384 non-null object
Brand Name      34846 non-null object
Price           40762 non-null float64
Rating          41384 non-null int64
Reviews         41374 non-null object
Review Votes    40194 non-null float64
dtypes: float64(2), int64(1), object(3)
memory usage: 2.2+ MB
```

```
df.describe()
```

	Price	Rating	Review Votes
count	40762.000000	41384.000000	40194.000000

Add new column

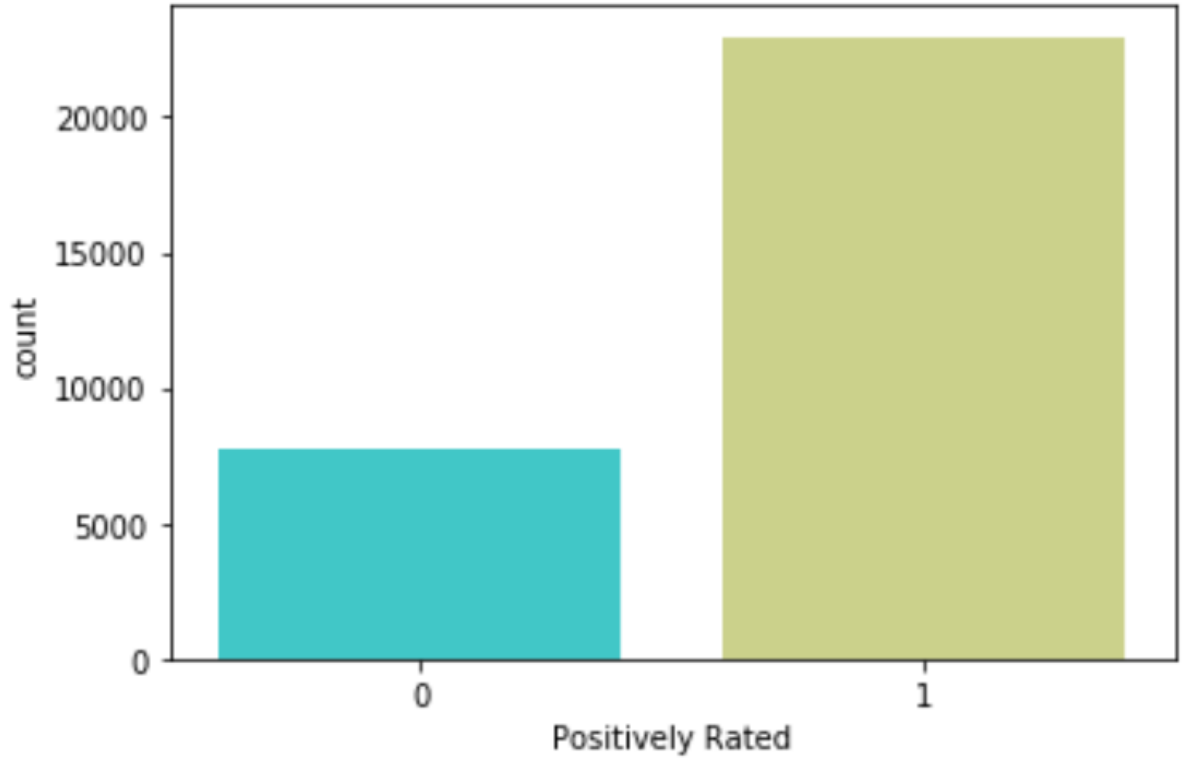
```
df.dropna(inplace=True)
df = df[df['Rating'] != 3]

df['Positively Rated'] = np.where(df['Rating'] > 3, 1, 0)
df.head(10)
```

```
df['Positively Rated'].mean()
```

Visualization

```
sns.countplot(x="Positively Rated", data=df, palette="rainbow")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x2553ed8c8c8>
```



Visualization

```
import matplotlib.pyplot as plt
df['Rating'].plot(kind='hist', figsize=(8, 5))

plt.title('Histogram of Rating') # add a title to the histogram
plt.ylabel('Number of') # add y-label
plt.xlabel('Rating') # add x-label

plt.show()
```

```
df['Review Votes'].plot(kind='hist', figsize=(4,5))

plt.title('Histogram of Review') # add a title to the histogram
plt.ylabel('Number of') # add y-label
plt.xlabel('Review Votes') # add x-label
plt.axis([0,200,None,None])

plt.show()
```

Pre-processing

```
df["Reviews"]=df["Reviews"].str.lower()
```

```
df.dtypes
```

```
Product Name          object
Brand Name            object
Price                float64
Rating                 int64
Reviews               object
Review Votes         float64
Positively Rated       int32
dtype: object
```

```
df["Reviews"]=df["Reviews"].astype('str')
```

Pre-processing

```
from nltk.stem import WordNetLemmatizer
from nltk.corpus import stopwords
import string

lemmatizer = WordNetLemmatizer()

def text_process(mess):

    # Check characters to see if they are in punctuation
    nopunc = [char for char in mess if char not in string.punctuation]

    # Join the characters again to form the string.
    nopunc = ''.join(nopunc)

    # Now just remove any stopwords
    words= [word for word in nopunc.split() if word.lower() not in stopwords.words('english')]

    return [lemmatizer.lemmatize(word) for word in words]
```

Pre-processing

```
df['Reviews'].apply(text_process)
```

```
34377          [phone, needed, sim, card, would, nice, know]
248521    [3, month, away, upgrade, stratosphere, kept, ...
167661                            [experience, want, forget]
73287          [great, phone, work, according, expectation]
277158    [fell, love, phone, everything, suppose, 3g, n...
                                ...
30001     [upgrade, compared, iphone, 4, going, love, ph...
313198    [liked, first, starting, lag, already, also, a...
138219                                                [nice]
66571     [new, one, tagboard, box, changed, checked, se...
109303    [phone, truly, terrible, purchase, 4gb, intern...
Name: Reviews, Length: 30737, dtype: object
```

```
X= df["Reviews"]
y=df['Positively Rated']
```

Count Vectorizer

```
from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer(min_df=5)
vect = cv.fit(X)
X_vect = vect.transform(X)
print(X_vect.shape)
```

```
(30737, 6360)
```

```
df1= pd.DataFrame(X_vect.toarray(), columns= cv.get_feature_names())
df1
```

	00	000	01	02	04	06	07	09	10	100	...	zenfone	zenfone2	zero	zippy	zone	zoom	zooming
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0

Count Vectorizer

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X_vect, y, test_size=0.2, random_state=0)
```

```
X_train
```

```
<24589x6360 sparse matrix of type '<class 'numpy.int64'>'
        with 634211 stored elements in Compressed Sparse Row format>
```

Multinomial Naïve Bayes

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import roc_auc_score,accuracy_score

classifier= MultinomialNB()
classifier.fit(X_train,y_train)

predictions = classifier.predict(X_test)

print('AUC: ', roc_auc_score(y_test, predictions)) # Area under curve score
print('Accuracy Score ', accuracy_score(y_test, predictions))
```

```
AUC:  0.8804468980377277
Accuracy Score  0.9154196486662329
```

Logistic Regression

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_auc_score,accuracy_score

classifier1= LogisticRegression()

classifier1.fit(X_train,y_train)
predictions1 = classifier1.predict((X_test))

print('AUC: ', roc_auc_score(y_test, predictions1))
print('Accuracy: ', accuracy_score(y_test, predictions1))
```

Prediction & Sort Coefficient

```
# These reviews predicted
print(classifier1.predict(vect.transform(['not an issue, phone is working',
                                          'an issue, phone is not working'])))
```

```
[0 0]
```

```
sorted_coef_index = classifier1.coef_[0].argsort()
feature_names = np.array(vect.get_feature_names())

print('Smallest Coefs:\n{}\n'.format(feature_names[sorted_coef_index[:10]]))
print('Largest Coefs: \n{}'.format(feature_names[sorted_coef_index[:-11:-1]]))
```

```
Smallest Coefs:
['worst' 'terrible' 'slow' 'junk' 'garbage' 'horrible' 'sucks' 'waste'
 'poor' 'useless']

Largest Coefs:
['excelent' 'excelente' 'excellent' 'perfectly' 'love' 'perfect' 'exactly'
 'great' 'awesome' 'loves']
```

Tfidf Vectorizer

```
from sklearn.feature_extraction.text import TfidfVectorizer
tf = TfidfVectorizer(min_df=5)
vect1 = tf.fit(X)
X_vect1 = vect1.transform(X)
print(X_vect1.shape)
```

```
(30737, 6360)
```

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X_vect1, y, test_size=0.2, random_state=0)
```

Multinomial Naïve Bayes

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import roc_auc_score,accuracy_score

classifier= MultinomialNB()
classifier.fit(X_train,y_train)

predictions = classifier.predict(X_test)

print('AUC: ', roc_auc_score(y_test, predictions))
print('Accuracy Score ', accuracy_score(y_test, predictions))
```

Logistic Regression

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_auc_score,accuracy_score

classifier1= LogisticRegression()
classifier1.fit(X_train,y_train)

predictions = classifier1.predict(X_test)

print('AUC: ', roc_auc_score(y_test, predictions))
print('Accuracy Score ', accuracy_score(y_test, predictions))
```

Prediction & Sort Coefficient

```
# These reviews predicted
print(classifier1.predict(vect1.transform(['not an issue, phone is working',
                                           'an issue, phone is not working'])))
```

```
[0 0]
```

```
feature_names = np.array(vect1.get_feature_names())
sorted_coef_index = classifier1.coef_[0].argsort()

print('Smallest Coefs:\n{}\n'.format(feature_names[sorted_coef_index[:10]]))
print('Largest Coefs: \n{}'.format(feature_names[sorted_coef_index[:-11:-1]]))
```

```
Smallest Coefs:
['not' 'slow' 'disappointed' 'terrible' 'worst' 'never' 'return' 'doesn'
 'waste' 'horrible']

Largest Coefs:
['great' 'love' 'excellent' 'good' 'best' 'perfect' 'price' 'awesome'
 'far' 'perfectly']
```

Sort tfidf values

```
sorted_tfidf_index = X_vect1.max(0).toarray()[0].argsort()

print('Smallest tfidf:\n{}\n'.format(feature_names[sorted_tfidf_index[:10]]))
print('Largest tfidf: \n{}'.format(feature_names[sorted_tfidf_index[:-11:-1]]))
```

```
Smallest tfidf:
['disabling' 'ft' '61' 'additions' 'combining' 'printer' 'circumference'
 '5v' 'adjustment' 'realistic']

Largest tfidf:
['handy' 'marvelous' 'brilliant' 'too' 'medium' 'kool' 'me' 'bad' 'top'
 'tops']
```

n - grams & tfidf

```
from sklearn.feature_extraction.text import TfidfVectorizer
vect2 = TfidfVectorizer(min_df=10, ngram_range=(1,3)).fit(X)
X_vect2 = vect2.transform(X)

print(X_vect2.shape)
```

```
(30737, 26513)
```

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X_vect2, y, test_size=0.2, random_state=0)
classifier1= LogisticRegression()
classifier1.fit(X_train,y_train)
```

Correct Prediction & Sort Coefficients

```
# These reviews are now correctly identified
print(classifier1.predict(vect2.transform(['not an issue, phone is working',
                                           'an issue, phone is not working'])))
```

```
[1 0]
```

```
print(classifier1.predict(vect2.transform(['phone is working smoothly , performance is good',
                                           'no issue, phone is working'])))
```

```
[1 1]
```

```
feature_names = np.array(vect2.get_feature_names())
sorted_coef_index = classifier1.coef_[0].argsort()
print('Smallest Coef: \n{}\n'.format(feature_names[sorted_coef_index][:10]))
print('Largest Coef: \n{}\n'.format(feature_names[sorted_coef_index][:-11:-1]))
```

```
Smallest Coef:
['not' 'slow' 'disappointed' 'never' 'bad' 'doesn' 'terrible' 'horrible'
 'worst' 'return']

Largest Coef:
['great' 'love' 'good' 'excellent' 'perfect' 'best' 'awesome' 'excelente'
```

n – gram & Count Vectorizer

```
vect3 = CountVectorizer(min_df=5, ngram_range=(1,3)).fit(X)

X_vect3 = vect3.transform(X)

print(X_vect3.shape)
```

```
(30737, 59480)
```

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X_vect3, y, test_size=0.2, random_state=0)
classifier1= LogisticRegression()
classifier1.fit(X_train,y_train)
```

Correct Prediction & Sort Coefficients

```
#These reviews are now correctly identified
print(classifier1.predict(vect3.transform(['not an issue, phone is working',
                                           'an issue, phone is not working'])))
```

```
[1 0]
```

```
print(classifier1.predict(vect3.transform(['phone is working smoothly , performance is good',
                                           'no issue, phone is working','phone is not good'])))
```

```
[1 1 0]
```

```
feature_names = np.array(vect3.get_feature_names())
sorted_coef_index = classifier1.coef_[0].argsort()
print('Smallest Coef: \n{}\n'.format(feature_names[sorted_coef_index][:10]))
print('Largest Coef: \n{}\n'.format(feature_names[sorted_coef_index][:-11:-1]))
```

```
Smallest Coef:
['no good' 'junk' 'poor' 'not good' 'slow' 'broken' 'worst' 'terrible'
 'defective' 'horrible']

Largest Coef:
['excellent' 'excelente' 'great' 'perfect' 'excelent' 'love' 'awesome'
```


सी डैक
CDAC
ज्ञानादेव तु केवलम्
ADVANCED COMPUTING FOR HUMAN ADVANCEMENT

THANK YOU