Karanjot Singh

In [52]:

```
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()
```

Out[52]:

	Product Name	Brand Name	Price	Rating	Reviews	Review Votes
394349	Sony XPERIA Z2 D6503 FACTORY UNLOCKED Internat...	NaN	244.95	5	Very good one! Better than Samsung S and iphon...	0.0
34377	Apple iPhone 5c 8GB (Pink) - Verizon Wireless	Apple	194.99	1	The phone needed a SIM card, would have been n...	1.0
248521	Motorola Droid RAZR MAXX XT912 M Verizon Smart...	Motorola	174.99	5	I was 3 months away from my upgrade and my Str...	3.0
167661	CNPGD [U.S. Office Extended Warranty] Smartwat...	CNPGD	49.99	1	an experience i want to forget	0.0
73287	Apple iPhone 7 Unlocked Phone 256 GB - US Vers...	Apple	922.00	5	GREAT PHONE WORK ACCORDING MY EXPECTATIONS.	1.0

In [53]:

```
df.tail(5)
```

Out[53]:

	Product Name	Brand Name	Price	Rating	Reviews	Review Votes
30001	Apple iPhone 5c 32GB (Blue) - AT&T	Apple	274.95	5	What an upgrade compared to the iPhone 4. Goin...	7.0
313198	Samsung Galaxy Grand Prime DUOS G531H/DS - Gra...	Samsung	179.99	4	I liked it at first but is starting to lag alr...	0.0
138219	BLU Studio 5.0 C HD Unlocked Cellphone, White	BLU	2000.00	4	very nice	0.0
66571	Apple iPhone 6s 64 GB International Warranty U...	Apple	689.95	1	It is not a new one. The tagboard on the box w...	0.0
109303	BLU Dash J Unlocked Phone - Retail Packaging -...	BLU	39.99	1	This phone was truly a terrible purchase!! It ...	1.0

In [54]:

```
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
```

In [55]:

```
df.describe()
```

Out[55]:

	Price	Rating	Review Votes
count	40762.000000	41384.000000	40194.000000
mean	227.626005	3.815170	1.498109
std	276.992862	1.551391	8.452564
min	1.730000	1.000000	0.000000
25%	79.950000	3.000000	0.000000
50%	140.000000	5.000000	0.000000
75%	269.990000	5.000000	1.000000
max	2408.730000	5.000000	524.000000

In [56]:

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

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

Out[56]:

	Product Name	Brand Name	Price	Rating	Reviews	Review Votes	Positively Rated
34377	Apple iPhone 5c 8GB (Pink) - Verizon Wireless	Apple	194.99	1	The phone needed a SIM card, would have been n...	1.0	0
248521	Motorola Droid RAZR MAXX XT912 M Verizon Smart...	Motorola	174.99	5	I was 3 months away from my upgrade and my Str...	3.0	1
167661	CNPGD [U.S. Office Extended Warranty] Smartwat...	CNPGD	49.99	1	an experience i want to forget	0.0	0
73287	Apple iPhone 7 Unlocked Phone 256 GB - US Vers...	Apple	922.00	5	GREAT PHONE WORK ACCORDING MY EXPECTATIONS.	1.0	1
277158	Nokia N8 Unlocked GSM Touch Screen Phone Featu...	Nokia	95.00	5	I fell in love with this phone because it did ...	0.0	1
100311	Blackberry Torch 2 9810 Unlocked Phone with 1....	BlackBerry	77.49	5	I am pleased with this Blackberry phone! The p...	0.0	1
251669	Motorola Moto E (1st Generation) - Black - 4 G...	Motorola	89.99	5	Great product, best value for money smartphone...	0.0	1
279878	OtterBox 77-29864 Defender Series Hybrid Case ...	OtterBox	9.99	5	I've bought 3 no problems. Fast delivery.	0.0	1
406017	Verizon HTC Rezound 4G Android Smarphone - 8MP...	HTC	74.99	4	Great phone for the price...	0.0	1
302567	RCA M1 Unlocked Cell Phone, Dual Sim, 5Mp Came...	RCA	159.99	5	My mom is not good with new technoloy but this...	4.0	1

In [57]:

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

Out[57]:

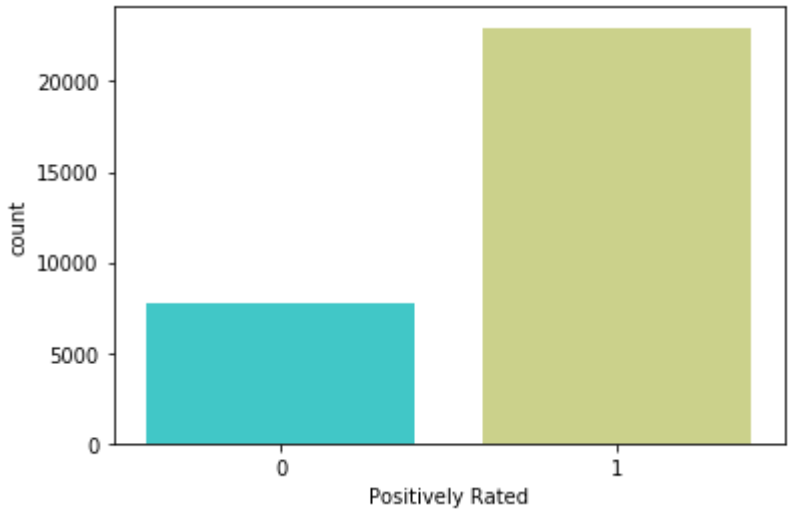
```
0.7471776686078667
```

In [58]:

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

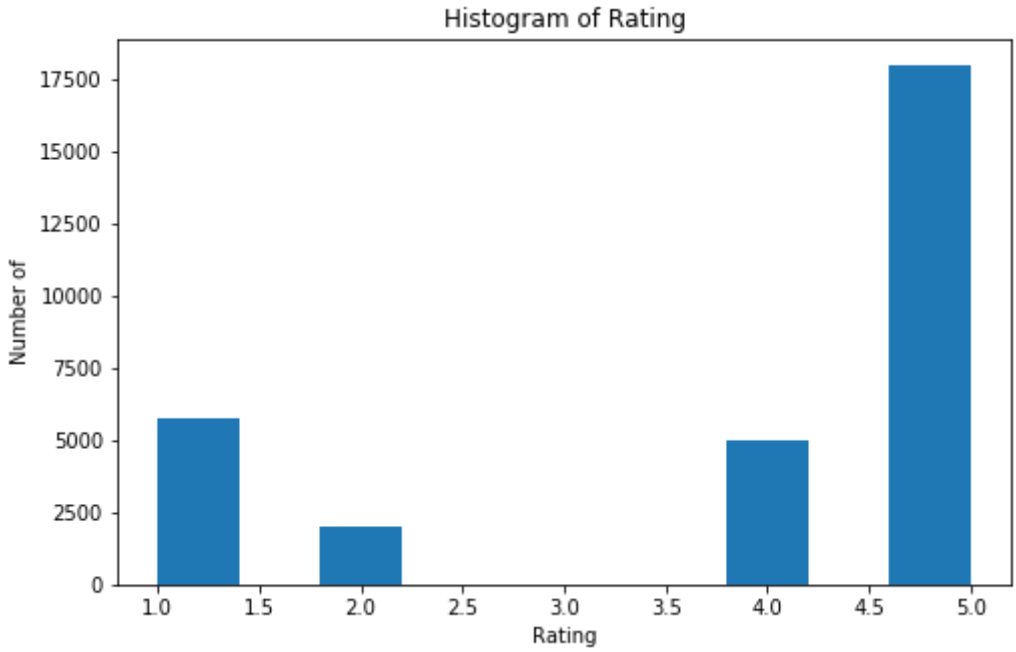
Out[58]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x20bc1135288>
```



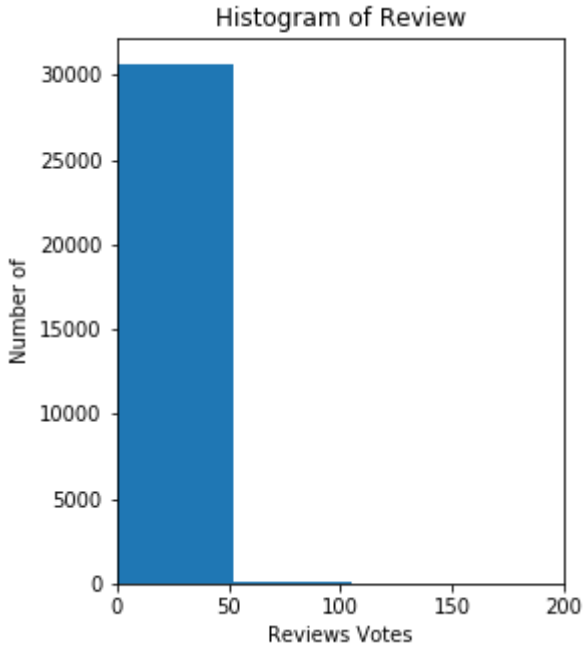
In [59]:

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

plt.title('Histogram of Rating')
plt.ylabel('Number of')
plt.xlabel('Rating')

plt.show()
```



In [60]:

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

plt.title('Histogram of Review')
plt.ylabel('Number of')
plt.xlabel('Reviews Votes')
plt.axis([0,200,None,None])
plt.show()
```



In [61]:

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

Out[61]:

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

In [62]:

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

In [63]:

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

lemmatizer = WordNetLemmatizer()

def text_process(mess):
    nopunc = [char for char in mess if char in string.punctuation]
    nopunc = ''.join(nopunc)
    words = [word for word in nopunc.split() if word.lower() not in stopwords.words('en
glish')]
    return [lemmatizer.lemmatize(word) for word in words]
```

In [64]:

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

Out[64]:

```
34377                                          [,.]
248521                             [.'.'..'!!'..''!!]
167661                                           []
73287                                           [.]
277158                                  [.,.....,.]
                           ...
30001                                        [...]
313198                                      [.,+.]
138219                                         []
66571                                   [...,...'?]
109303    [!!!!,,,!!,!!'!!!!!!!!'!!!!'?!?!''!!!!!!!!]
Name: Reviews, Length: 30737, dtype: object
```

In [65]:

```
x = df["Reviews"]
y = df["Positively Rated"]
```

In [66]:

```
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)
```

In [67]:

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

Out[67]:

	00	000	01	02	04	06	07	09	10	100	...	zenfone	zenfone2	zero	zippy	zone
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
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
30732	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
30733	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
30734	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
30735	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
30736	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0

30737 rows × 6360 columns

In [68]:

```
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)
```

In [69]:

```
x_train
```

Out[69]:

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

In [70]:

```
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))
```

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

In [72]:

```
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))
```

```
C:\Users\Karan Singh\Anaconda3\lib\site-packages\sklearn\linear_model\logi
stic.py:432: FutureWarning: Default solver will be changed to 'lbfgs' in
0.22. Specify a solver to silence this warning.
  FutureWarning)

AUC:  0.8968248590633131
Accuracy:  0.9311971372804164
```

In [73]:

```
print(classifier1.predict(vect.transform(['not an issue, phone is working','an issue, p
hone is not working'])))
```

```
[0 0]
```

In [74]:

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

print('Smallest Coefs:\n{}\n'.format(feature_name[sorted_coef_index[:10]]))
print('Largest Coefs:\n{}'.format(feature_name[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']
```

In [75]:

```
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)
```

In [76]:

```
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)
```

In [77]:

```
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))
```

```
AUC:  0.8291111615531946
Accuracy Score:  0.9035458685751464
```

In [78]:

```
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))
```

```
C:\Users\Karan Singh\Anaconda3\lib\site-packages\sklearn\linear_model\logi
stic.py:432: FutureWarning: Default solver will be changed to 'lbfgs' in
0.22. Specify a solver to silence this warning.
  FutureWarning)

AUC:  0.8899866629625613
Accuracy:  0.9303838646714379
```

In [79]:

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

```
[0 0]
```

In [80]:

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

print('Smallest Coefs:\n{}\n'.format(feature_name[sorted_coef_index[:10]]))
print('Largest Coefs:\n{}'.format(feature_name[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']
```

In [81]:

```
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']
```

In [84]:

```
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)
```

In [85]:

```
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)
```

In [87]:

```
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)

print(classifier1.predict(vect2.transform(['not an issue, phone is working','an issue,
 phone is not working'])))
```

```
[1 0]
```

In [88]:

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

```
[1 1]
```

In [90]:

```
feature_names = np.array(vect2.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' 'never' 'bad' 'doesn' 'terrible' 'horrible'
 'worst' 'return']

Largest Coefs:
['great' 'love' 'good' 'excellent' 'perfect' 'best' 'awesome' 'excelente'
 'price' 'nice']
```

In [92]:

```
vect3 = CountVectorizer(min_df=5 , ngram_range=(1,3)).fit(x)
x_vect3 = vect3.transform(x)

print(x_vect3.shape)
```

```
(30737, 59480)
```

In [96]:

```
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)

print(classifier1.predict(vect3.transform(['not an issue, phone is working','an issue,
 phone is not working'])))

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

```
[1 0]
[1 1 0]
```

In [97]:

```
feature_names = np.array(vect3.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:
['no good' 'junk' 'poor' 'not good' 'slow' 'broken' 'worst' 'terrible'
 'defective' 'horrible']

Largest Coefs:
['excellent' 'excelente' 'great' 'perfect' 'excelent' 'love' 'awesome'
 'no problems' 'good' 'not bad']
```

In [ ]:

```

```