<http://localhost:8888/notebooks/final.ipynb>

# iPhone X Reviews: Sentiment Analysis

## Opinion Mining

```python

from lxml import html

import requests

import re

page = requests.get(url)

tree = html.fromstring(page.content)

comments\_section = tree.xpath('///text()')

part1=re.split('text":"',comments\_section[20])

comments\_list=[]

for i in range(3,len(part1),2):

part2=re.split('"},"ranges"',part1[i])

comment=str(part2[0])

comment= comment.replace("\\u2019","'")

comment= comment.replace("\\u00a3","£")

comment= comment.replace("u0025","%")

comment= comment.replace("u0040","@")

comment= comment.replace("\nn"," ")

comment= comment.replace("\n"," ")

comment= comment.replace("\t"," ")

comment= comment.replace('\\', '')

comment= comment.replace('.t', '.')

comment= comment.replace('nn', '')

comment= comment.replace('n-n-', '')

comment= comment.replace('-n-n', '')

comment= comment.replace('. n', ' ')

comment= comment.replace('. n', ' ')

comment= comment.replace('.n', ' ')

comment= comment.replace('?n', ' ')

comment= comment.replace('\'',"’")

comments\_list.append(comment)

```

```python

# if len(comments\_list)==0, use comments\_section[21] in the definition of part1

len(comments\_list)

```

```python

# Three sample comments

comments\_list[:3]

## Data Prep for Sentiment Analysis

```python

import pandas as pd

import numpy as np

# Read in the data

df1 = pd.read\_csv('https://raw.githubusercontent.com/ehsansade/Sentiment-Analysis/master/Amazon\_Unlocked\_Mobile\_Reviews1.csv')

df2 = pd.read\_csv('https://raw.githubusercontent.com/ehsansade/Sentiment-Analysis/master/Amazon\_Unlocked\_Mobile\_Reviews2.csv')

df=pd.concat([df1,df2],ignore\_index=True)

df.head()

<div>

<style>

.dataframe thead tr:only-child th {

text-align: right;

}

.dataframe thead th {

text-align: left;

}

.dataframe tbody tr th {

vertical-align: top;

}

</style>

<table border="1" class="dataframe">

<thead>

<tr style="text-align: right;">

<th></th>

<th>Product Name</th>

<th>Brand Name</th>

<th>Price</th>

<th>Rating</th>

<th>Reviews</th>

<th>Review Votes</th>

</tr>

</thead>

<tbody>

<tr>

<th>0</th>

<td>Apple iPhone SE Unlocked Phone - 64 GB Retail ...</td>

<td>Apple</td>

<td>514.98</td>

<td>5</td>

<td>Exactly what I wanted, fast shipping.</td>

<td>0.0</td>

</tr>

<tr>

<th>1</th>

<td>Apple iPhone SE Unlocked Phone - 64 GB Retail ...</td>

<td>Apple</td>

<td>514.98</td>

<td>5</td>

<td>I was a little skeptical about buying electron...</td>

<td>1.0</td>

</tr>

<tr>

<th>2</th>

<td>Apple iPhone SE Unlocked Phone - 64 GB Retail ...</td>

<td>Apple</td>

<td>514.98</td>

<td>1</td>

<td>Phone was not what I ordered, very disappointe...</td>

<td>1.0</td>

</tr>

<tr>

<th>3</th>

<td>Apple iPhone SE Unlocked Phone - 64 GB Retail ...</td>

<td>Apple</td>

<td>514.98</td>

<td>5</td>

<td>The sexy body of a 5s and the guts of a 6........</td>

<td>0.0</td>

</tr>

<tr>

<th>4</th>

<td>Apple iPhone SE Unlocked Phone - 64 GB Retail ...</td>

<td>Apple</td>

<td>514.98</td>

<td>5</td>

<td>Love it!</td>

<td>0.0</td>

</tr>

</tbody>

</table>

</div>

```python

# Drop missing values

df.dropna(inplace=True)

# Remove any 'neutral' ratings equal to 3

df = df[df['Rating'] != 3]

# Encode 4s and 5s as 1 (rated positively)

# Encode 1s and 2s as 0 (rated poorly)

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

df.head()

<div>

<style>

.dataframe thead tr:only-child th {

text-align: right;

}

.dataframe thead th {

text-align: left;

}

.dataframe tbody tr th {

vertical-align: top;

}

</style>

<table border="1" class="dataframe">

<thead>

<tr style="text-align: right;">

<th></th>

<th>Product Name</th>

<th>Brand Name</th>

<th>Price</th>

<th>Rating</th>

<th>Reviews</th>

<th>Review Votes</th>

<th>Positively Rated</th>

</tr>

</thead>

<tbody>

<tr>

<th>0</th>

<td>Apple iPhone SE Unlocked Phone - 64 GB Retail ...</td>

<td>Apple</td>

<td>514.98</td>

<td>5</td>

<td>Exactly what I wanted, fast shipping.</td>

<td>0.0</td>

<td>1</td>

</tr>

<tr>

<th>1</th>

<td>Apple iPhone SE Unlocked Phone - 64 GB Retail ...</td>

<td>Apple</td>

<td>514.98</td>

<td>5</td>

<td>I was a little skeptical about buying electron...</td>

<td>1.0</td>

<td>1</td>

</tr>

<tr>

<th>2</th>

<td>Apple iPhone SE Unlocked Phone - 64 GB Retail ...</td>

<td>Apple</td>

<td>514.98</td>

<td>1</td>

<td>Phone was not what I ordered, very disappointe...</td>

<td>1.0</td>

<td>0</td>

</tr>

<tr>

<th>3</th>

<td>Apple iPhone SE Unlocked Phone - 64 GB Retail ...</td>

<td>Apple</td>

<td>514.98</td>

<td>5</td>

<td>The sexy body of a 5s and the guts of a 6........</td>

<td>0.0</td>

<td>1</td>

</tr>

<tr>

<th>4</th>

<td>Apple iPhone SE Unlocked Phone - 64 GB Retail ...</td>

<td>Apple</td>

<td>514.98</td>

<td>5</td>

<td>Love it!</td>

<td>0.0</td>

<td>1</td>

</tr>

</tbody>

</table>

</div>

```python

from sklearn.model\_selection import train\_test\_split

# Split data into training and test sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(df['Reviews'],

df['Positively Rated'],random\_state=3)

```

## CountVectorizer, n-grams

```python

from sklearn.feature\_extraction.text import CountVectorizer

vect = CountVectorizer(min\_df=5, ngram\_range=(1,2)).fit(X\_train)

X\_train\_vectorized = vect.transform(X\_train)

len(vect.get\_feature\_names())

103120

```python

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import roc\_auc\_score

model = LogisticRegression()

model.fit(X\_train\_vectorized, y\_train)

predictions = model.predict(vect.transform(X\_test))

print('ROC\_AUC: ', roc\_auc\_score(y\_test, predictions))

```

ROC\_AUC: 0.969177058414

```python

feature\_names = np.array(vect.get\_feature\_names())

sorted\_coef\_index = model.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' 'worst' 'poor' 'looks ok' 'not good' 'horrible'

'terrible' 'not happy' 'not very']

Largest Coefs:

['excellent' 'excelent' 'excelente' 'not bad' 'no problems' 'perfect'

'awesome' 'exelente' 'great' 'amazing']

```python

# check the model to see if it correctly identifies a positive review

print(model.predict(vect.transform(['not an issue, phone is working',

'an issue, phone is not working'])))

```

[1 0]

# Sentiment Analysis for iPhone X

```python

comments\_positivity=model.predict(vect.transform(comments\_list))

print(comments\_positivity)

```

[1 1 1 1 0 1 1 0 1 1 0 1 1 1 0 0 0 0 0 0 0 1 1 1 0 1 0 0 0 0 1 1 0 0 0 1 1

1 0 0 1 1 0 0 1 1 0 0 1 1 1 1 1 1 0 0 1 0 1 1 1 1 0 0 1 1 0 1 1 1 1 1 1 0

0 1 0 0 0 0 0 0 1 0 1 0 1 0 1 1 1 1 1 1 1 1 0 1 1 1 1 0 0 0 1 0 1 1 1 1 1

0 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 0]

```python

comments\_positivity.mean()

```

0.609375