Predict Sentiments of Amazon Customers using Python

Title of the project:

Predict Sentiments of Amazon Customers using Python.

Description:

Hello everyone!

In this tutorial, we are going to predict the sentiments of Amazon customers using Python. We mainly use NumPy, pandas, seaborn and scikit-learn(sklearn) libraries for this purpose. We apply a Logistic Regression machine learning algorithm on our data.

It calculates the top 20 positive and negative words. Also, it gives testing accuracy, confusion matrix and model accuracy.

Prerequisites:

1) Dataset file of reviews with a .csv extension.
2) Install Jupyter Notebook or any similar working environment with the latest version of Python installed.
3) Python language.
4) Knowledge of Python libraries like NumPy, pandas, scikit-learn(sklearn) , seaborn.

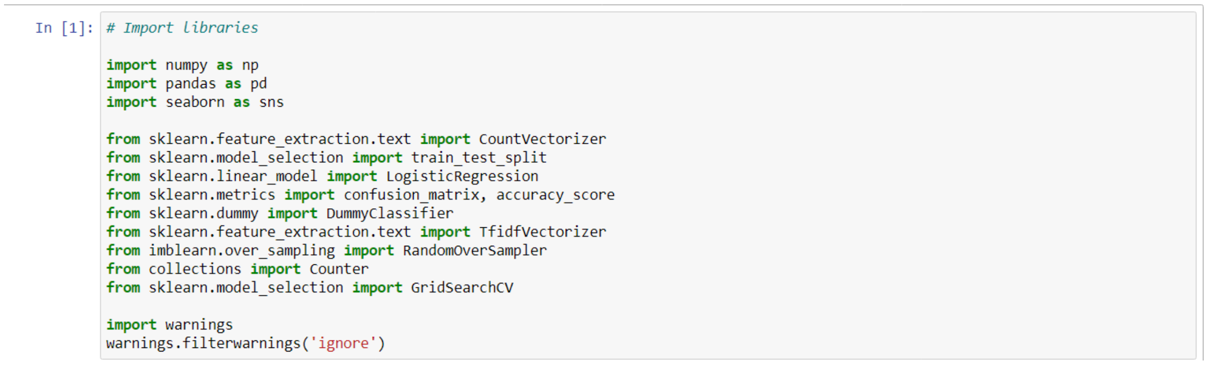
Datasets:

It contains the dataset of reviews(568454, 10).

Link : Reviews.csv

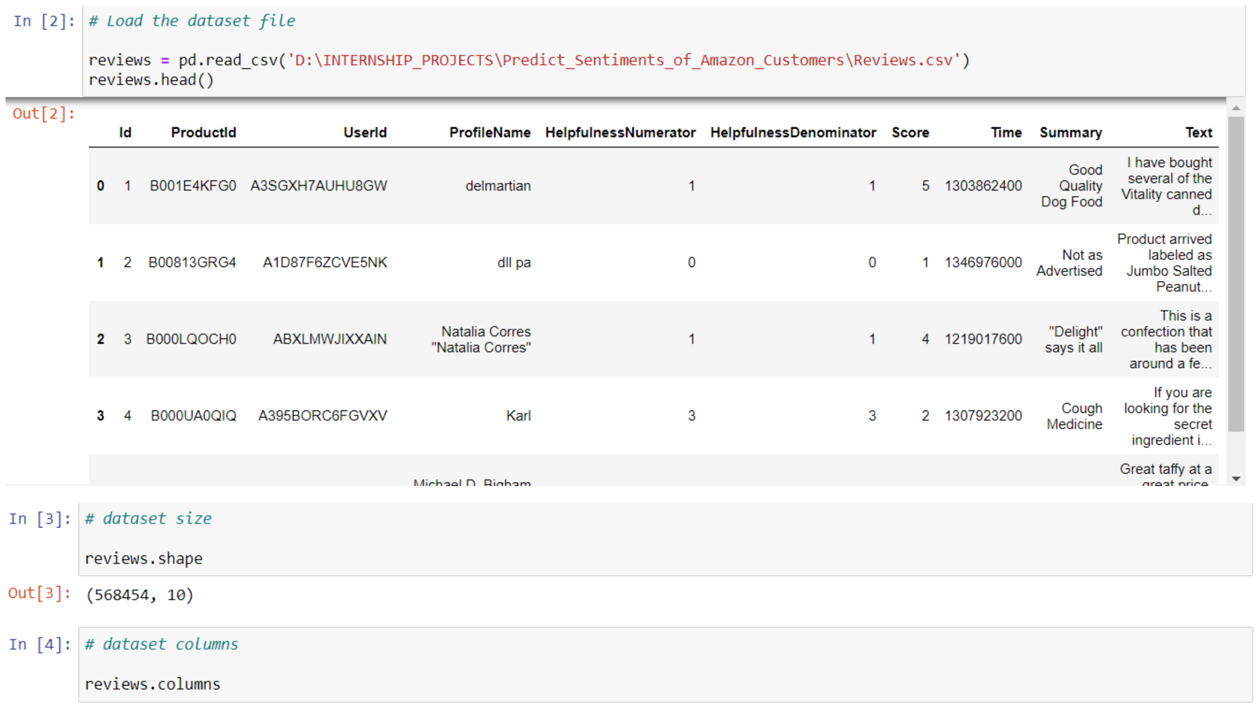
Implementation:

1) Import the required Python libraries.



```
In [1]: # Import libraries

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

        from sklearn.feature_extraction.text import CountVectorizer
        from sklearn.model_selection import train_test_split
        from sklearn.linear_model import LogisticRegression
        from sklearn.metrics import confusion_matrix, accuracy_score
        from sklearn.dummy import DummyClassifier
        from sklearn.feature_extraction.text import TfidfVectorizer
        from imblearn.over_sampling import RandomOverSampler
        from collections import Counter
        from sklearn.model_selection import GridSearchCV

        import warnings
        warnings.filterwarnings('ignore')
```

2) Reading the dataset. It contains a dataset of reviews. This dataset is present in the .csv extension file.



```
In [2]: # Load the dataset file

        reviews = pd.read_csv('D:\INTERNSHIP_PROJECTS\Predict_Sentiments_of_Amazon_Customers\Reviews.csv')
        reviews.head()
```

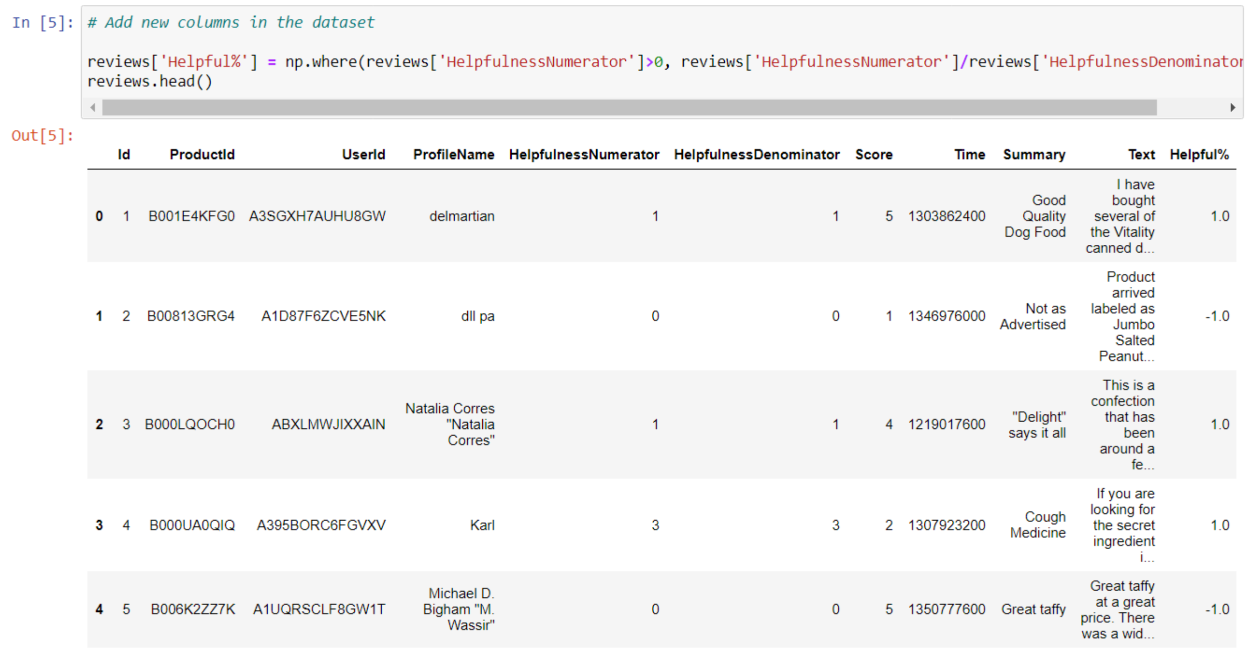
Out[2]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time	Summary	Text
0	1	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	1	5	1303862400	Good Quality Dog Food	I have bought several of the Vitality canned d...
1	2	B00813GRG4	A1D87F6ZCVE5NK	dll pa	0	0	1	1346976000	Not as Advertised	Product arrived labeled as Jumbo Salted Peanut...
2	3	B000LQOCH0	ABXLMWJIXXAIN	Natalia Corres "Natalia Corres"	1	1	4	1219017600	"Delight" says it all	This is a confection that has been around a fe...
3	4	B000UA0QIQ	A395BORC6FGVXV	Karl	3	3	2	1307923200	Cough Medicine	If you are looking for the secret ingredient i...
				Michael D. Bigham						Great taffy at a great price

```
In [3]: # dataset size

        reviews.shape

Out[3]: (568454, 10)
```

```
In [4]: # dataset columns

        reviews.columns
```

3) First, we add a new column of helpful% to our review dataset.

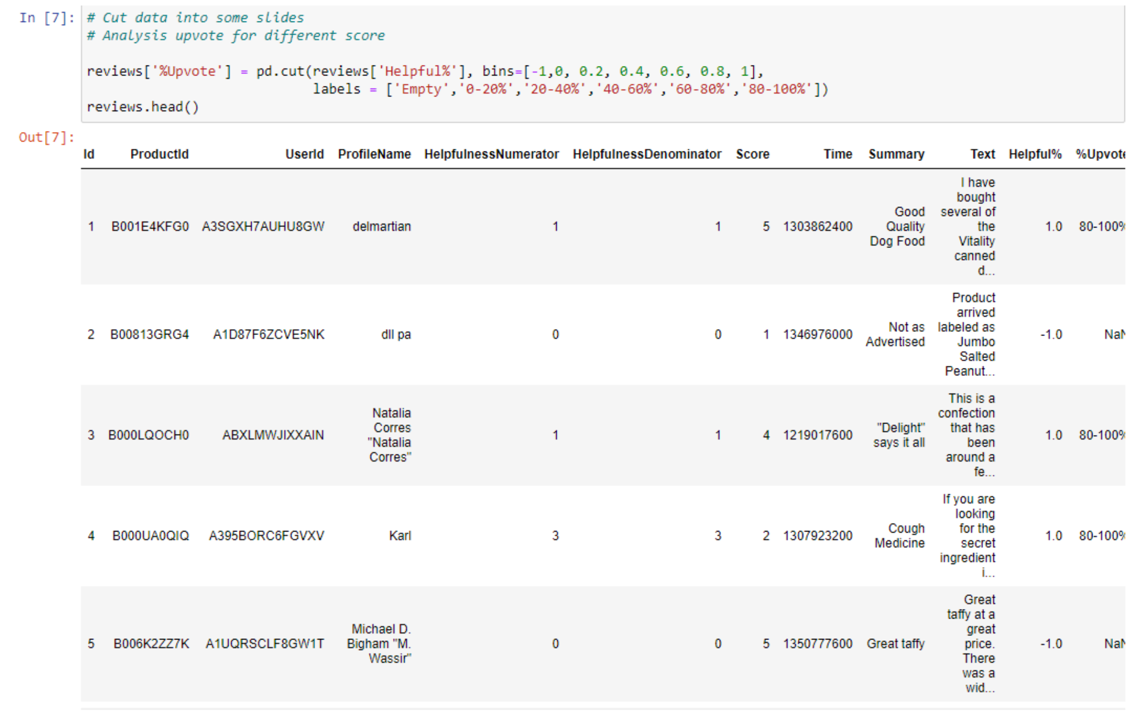


```
In [5]: # Add new columns in the dataset

        reviews['Helpful%'] = np.where(reviews['HelpfulnessNumerator']>0, reviews['HelpfulnessNumerator']/reviews['HelpfulnessDenominator
        reviews.head()
```

Out[5]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time	Summary	Text	Helpful%
0	1	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	1	5	1303862400	Good Quality Dog Food	I have bought several of the Vitality canned d...	1.0
1	2	B00813GRG4	A1D87F6ZCVE5NK	dll pa	0	0	1	1346976000	Not as Advertised	Product arrived labeled as Jumbo Salted Peanut...	-1.0
2	3	B000LQOCH0	ABXLMWJIXXAIN	Natalia Corres "Natalia Corres"	1	1	4	1219017600	"Delight" says it all	This is a confection that has been around a fe...	1.0
3	4	B000UA0QIQ	A395BORC6FGVXV	Karl	3	3	2	1307923200	Cough Medicine	If you are looking for the secret ingredient i...	1.0
4	5	B006K2ZZ7K	A1UQRSCLF8GW1T	Michael D. Bigham "M. Wassir"	0	0	5	1350777600	Great taffy	Great taffy at a great price. There was a wid...	-1.0

4) After that, cut the data into some slides and then analyze upvotes for different scores.

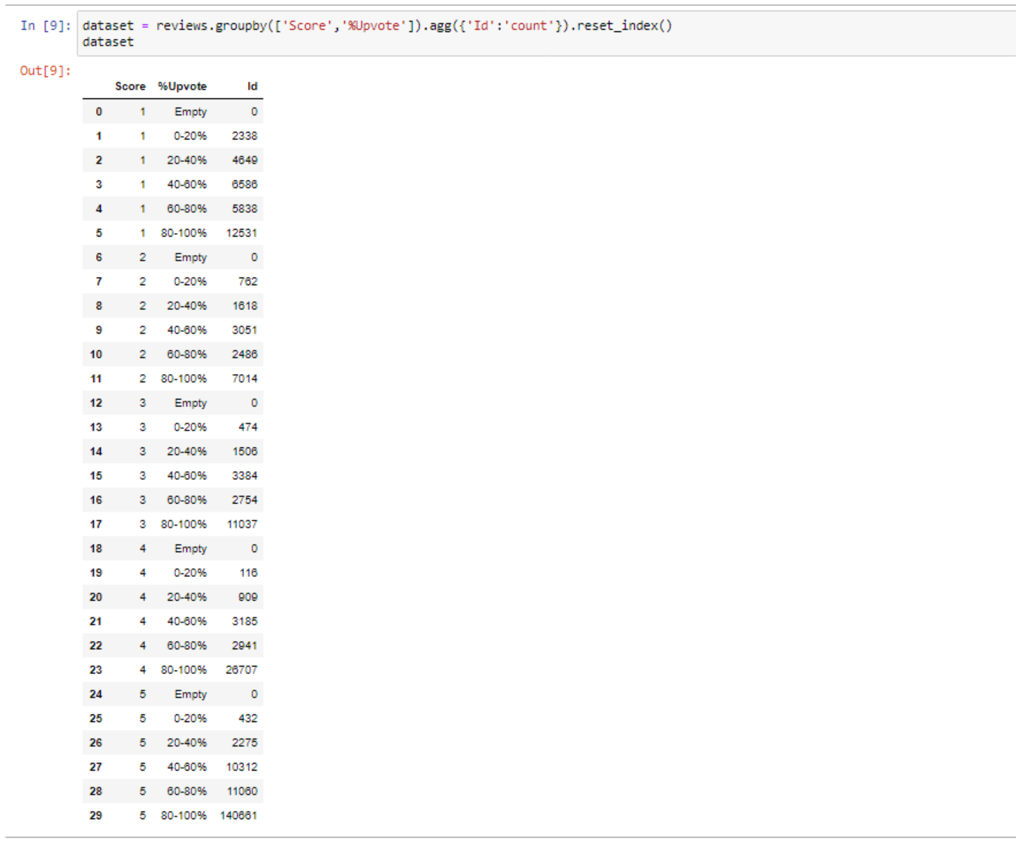


```
In [7]: # Cut data into some slides
        # Analysis upvote for different score

        reviews['%Upvote'] = pd.cut(reviews['Helpful%'], bins=[-1,0, 0.2, 0.4, 0.6, 0.8, 1],
                                    labels = ['Empty','0-20%','20-40%','40-60%','60-80%','80-100%'])
        reviews.head()
```

Out[7]:

Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time	Summary	Text	Helpful%	%Upvot
1	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	1	5	1303862400	Good Quality Dog Food	I have bought several of the Vitality canned d...	1.0	80-100
2	B00813GRG4	A1D87F6ZCVE5NK	dll pa	0	0	1	1346976000	Not as Advertised	Product arrived labeled as Jumbo Salted Peanut...	-1.0	Na
3	B000LQOCH0	ABXLMWJIXXAIN	Natalia Corres "Natalia Corres"	1	1	4	1219017600	"Delight" says it all	This is a confection that has been around a fe...	1.0	80-100
4	B000UA0QIQ	A395BORC6FGVXV	Karl	3	3	2	1307923200	Cough Medicine	If you are looking for the secret ingredient i...	1.0	80-100
5	B006K2ZZ7K	A1UQRSCLF8GW1T	Michael D. Bigham "M. Wassir"	0	0	5	1350777600	Great taffy	Great taffy at a great price. There was a wid...	-1.0	Na

5) Next, we prepare a dataset containing scores and upvotes along with Id.



```
In [9]: dataset = reviews.groupby(['Score','%Upvote']).agg({'Id':'count'}).reset_index()
        dataset
```

Out[9]:

	Score	%Upvote	Id
0	1	Empty	0
1	1	0-20%	2338
2	1	20-40%	4649
3	1	40-60%	6586
4	1	60-80%	5838
5	1	80-100%	12531
6	2	Empty	0
7	2	0-20%	762
8	2	20-40%	1618
9	2	40-60%	3051
10	2	60-80%	2486
11	2	80-100%	7014
12	3	Empty	0
13	3	0-20%	474
14	3	20-40%	1506
15	3	40-60%	3384
16	3	60-80%	2754
17	3	80-100%	11037
18	4	Empty	0
19	4	0-20%	116
20	4	20-40%	909
21	4	40-60%	3185
22	4	60-80%	2941
23	4	80-100%	26707
24	5	Empty	0
25	5	0-20%	432
26	5	20-40%	2275
27	5	40-60%	10312
28	5	60-80%	11060
29	5	80-100%	140661

6) Create a pivot table for the dataset and plot a heatmap of this pivot table.

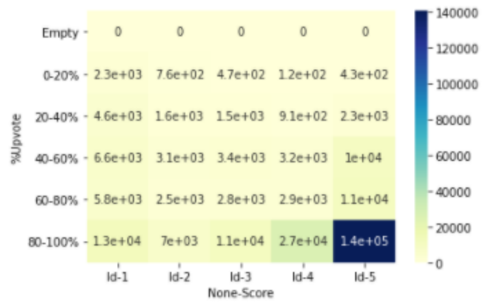
```
In [10]: # Create pivot table

         pivot = dataset.pivot(index = '%Upvote', columns='Score')
         pivot
```

Out[10]:

	Id				
Score	**1**	**2**	**3**	**4**	**5**
%Upvote					
Empty	0	0	0	0	0
0-20%	2338	762	474	116	432
20-40%	4649	1618	1506	909	2275
40-60%	6586	3051	3384	3185	10312
60-80%	5838	2486	2754	2941	11060
80-100%	12531	7014	11037	26707	140661

```
In [11]: # Heatmap

         sns.heatmap(pivot, annot=True, cmap='YlGnBu')
```

```
Out[11]: <AxesSubplot:xlabel='None-Score', ylabel='%Upvote'>
```



7) Now, start the calculation for prediction.

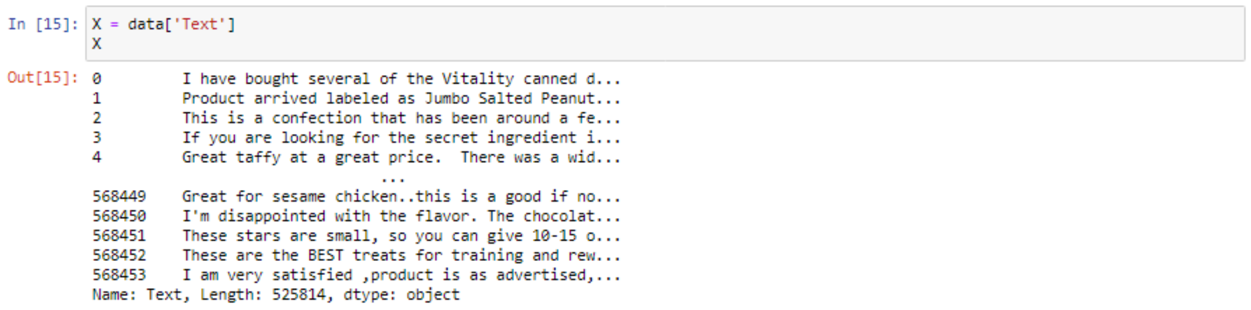
8) Remove reviews having a score value of 3, as it represents a neutral score.

```
In [13]: # remove all 3 as it represent a neutral score

         data = reviews[reviews['Score']!=3]
         data.head()
```

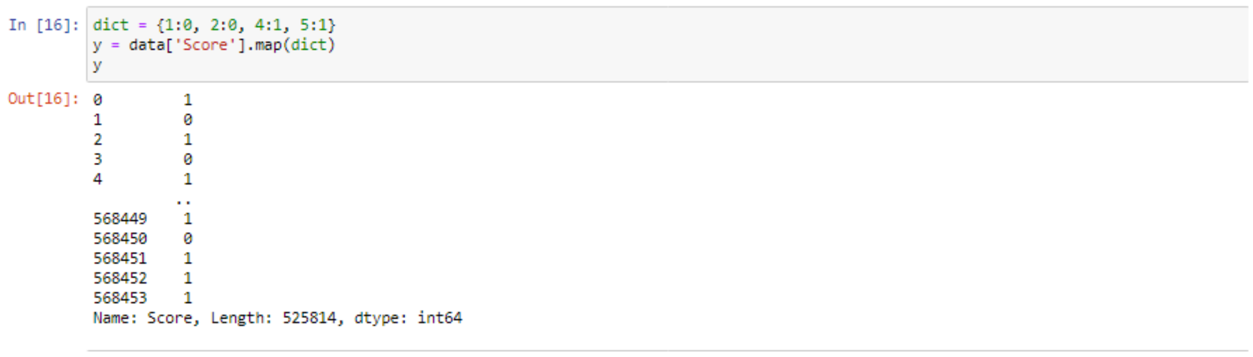
Out[13]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time	Summary	Text	Helpful%	%U
0	1	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	1	5	1303862400	Good Quality Dog Food	I have bought several of the Vitality canned d...	1.0	80
1	2	B00813GRG4	A1D87F6ZCVE5NK	dll pa	0	0	1	1346976000	Not as Advertised	Product arrived labeled as Jumbo Salted Peanut...	-1.0	
2	3	B000LQOCH0	ABXLMWJIXXAIN	Natalia Corres "Natalia Corres"	1	1	4	1219017600	"Delight" says it all	This is a confection that has been around a fe...	1.0	80
3	4	B000UA0QIQ	A395BORC6FGVXV	Karl	3	3	2	1307923200	Cough Medicine	If you are looking for the secret ingredient i...	1.0	80
4	5	B006K2ZZ7K	A1UQRSCLF8GW1T	Michael D. Bigham "M. Wassir"	0	0	5	1350777600	Great taffy	Great taffy at a great price. There was a wid...	-1.0	

9) Prepare ‘X’ and ‘y’ variables. ‘X’ represents our text data and ‘y’ represents the score.

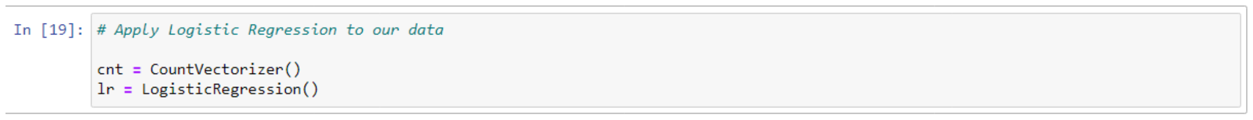


```
In [15]: X = data['Text']
         X

Out[15]: 0         I have bought several of the Vitality canned d...
         1         Product arrived labeled as Jumbo Salted Peanut...
         2         This is a confection that has been around a fe...
         3         If you are looking for the secret ingredient i...
         4         Great taffy at a great price.  There was a wid...
                                         ...
         568449    Great for sesame chicken..this is a good if no...
         568450    I'm disappointed with the flavor. The chocolat...
         568451    These stars are small, so you can give 10-15 o...
         568452    These are the BEST treats for training and rew...
         568453    I am very satisfied ,product is as advertised,...
         Name: Text, Length: 525814, dtype: object
```



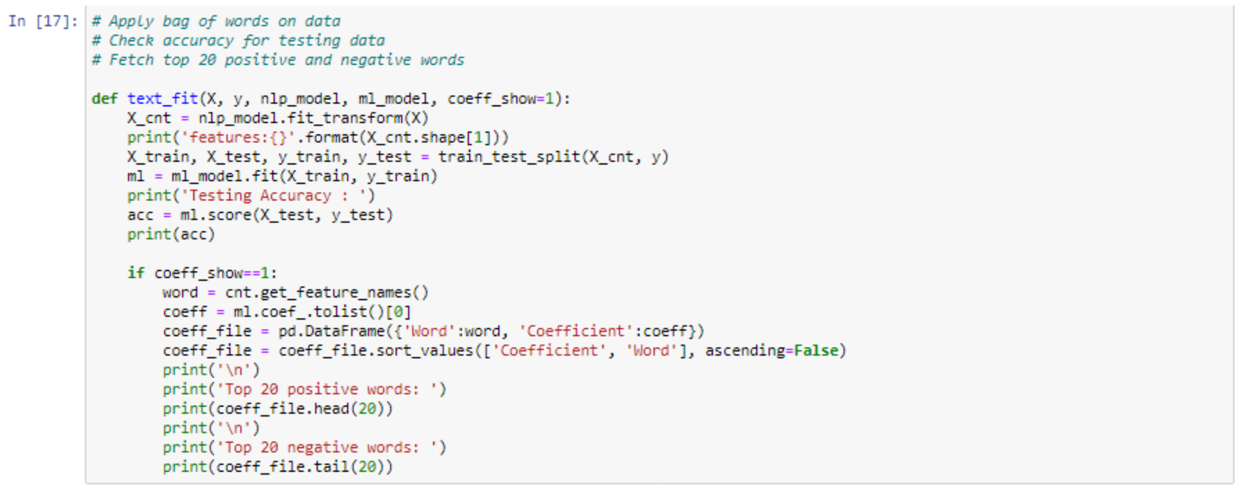
```
In [16]: dict = {1:0, 2:0, 4:1, 5:1}
         y = data['Score'].map(dict)
         y

Out[16]: 0         1
         1         0
         2         1
         3         0
         4         1
                  ..
         568449    1
         568450    0
         568451    1
         568452    1
         568453    1
         Name: Score, Length: 525814, dtype: int64
```

10) To predict sentiments we apply logistic regression machine learning algorithms on data.



```
In [19]: # Apply Logistic Regression to our data

         cnt = CountVectorizer()
         lr = LogisticRegression()
```

11) Apply Bag of words on data. Calculate the test accuracy and print the top 20 positive and negative words.



```
In [17]: # Apply bag of words on data
         # Check accuracy for testing data
         # Fetch top 20 positive and negative words

         def text_fit(X, y, nlp_model, ml_model, coeff_show=1):
             X_cnt = nlp_model.fit_transform(X)
             print('features:{}'.format(X_cnt.shape[1]))
             X_train, X_test, y_train, y_test = train_test_split(X_cnt, y)
             ml = ml_model.fit(X_train, y_train)
             print('Testing Accuracy : ')
             acc = ml.score(X_test, y_test)
             print(acc)

             if coeff_show==1:
                 word = cnt.get_feature_names()
                 coeff = ml.coef_.tolist()[0]
                 coeff_file = pd.DataFrame({'Word':word, 'Coefficient':coeff})
                 coeff_file = coeff_file.sort_values(['Coefficient', 'Word'], ascending=False)
                 print('\n')
                 print('Top 20 positive words: ')
                 print(coeff_file.head(20))
                 print('\n')
                 print('Top 20 negative words: ')
                 print(coeff_file.tail(20))
```

```
In [20]: text_fit(X,y,cnt, lr)
```

```
features:115282
Testing Accuracy :
0.9382369498075372


Top 20 positive words:
               Word  Coefficient
55155        hooked     2.447969
80801    pleasantly     2.390047
94888     skeptical     2.170149
35706     delicious     2.127111
19523          beat     2.074250
113443      worried     2.010215
86940    refreshing     1.910093
79105     perfectly     1.887336
44845     favorites     1.825904
11025       awesome     1.696881
114741        yummy     1.693897
114673          yum     1.688697
91121     satisfied     1.683484
44544     fantastic     1.659558
5867       addicted     1.650786
79089       perfect     1.643734
80810       pleased     1.615207
43310     excellent     1.611712
76497   outstanding     1.599568
103091     terrific     1.598428


Top 20 negative words:
                  Word  Coefficient
88559        returning    -1.667014
30805          concept    -1.741261
46180       flavorless    -1.760915
11033            awful    -1.787659
25006        cancelled    -1.789247
37909       disgusting    -1.793610
57596         inedible    -1.800359
107654      undrinkable    -1.810907
90069           ruined    -1.831228
93250            shame    -1.863703
94968             skip    -1.868341
89135              rip    -1.898396
103079        terrible    -1.936823
102272       tasteless    -2.010035
55204            hopes    -2.125152
96731          sounded    -2.134086
114633            yuck    -2.582982
37630   disappointment    -2.719006
37627    disappointing    -2.974698
113470           worst    -3.025679
```

12) Calculate the accuracy of the model along with the confusion matrix.

```
In [18]: # Predictions

         def predict(X,y,nlp_model, ml_model):
             X_cnt = nlp_model.fit_transform(X)
             X_train, X_test, y_train, y_test = train_test_split(X_cnt, y)
             ml = ml_model.fit(X_train, y_train)
             predictions = ml.predict(X_test)
             cm = confusion_matrix(predictions, y_test)
             print('Confusion Matrix: ')
             print(cm)
             print('\n')
             acc = accuracy_score(predictions, y_test)
             print('Accuracy of the model: ')
             print(acc)
```

```
In [21]: predict(X,y,cnt,lr)
```

```
Confusion Matrix:
[[ 15106   2777]
 [  5540 108031]]


Accuracy of the model:
0.9367307194912289
```