Amezon Alexa data set using NLP 1

May 10, 2021

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stat
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
#Automatic EDA using DTale Laibrary
import dtale
```

```
C:\Users\Chandra Sekhar\Anaconda3\lib\site-
packages\statsmodels\tools\_testing.py:19: FutureWarning: pandas.util.testing is
deprecated. Use the functions in the public API at pandas.testing instead.
  import pandas.util.testing as tm
```

[4]:
```
df=pd.read_csv('amazon_alexa.csv')
```

[3]:
```
df.head()
```

[3]:

	rating	date	variation	verified_reviews	feedback
0	5	31-Jul-18	Charcoal Fabric	Love my Echo!	1
1	5	31-Jul-18	Charcoal Fabric	Loved it!	1
2	4	31-Jul-18	Walnut Finish	Sometimes while playing a game, you can answer...	1
3	5	31-Jul-18	Charcoal Fabric	I have had a lot of fun with this thing. My 4 ...	1
4	5	31-Jul-18	Charcoal Fabric	Music	1

[5]:
```
df['variation'].value_counts()
```

[5]:
```
Black  Dot                   516
Charcoal Fabric              430
```

```
Configuration: Fire TV Stick    350
Black  Plus                      270
Black  Show                      265
Black                            261
Black  Spot                      241
White  Dot                       184
Heather Gray Fabric              157
White  Spot                      109
White                             91
Sandstone Fabric                  90
White  Show                       85
White  Plus                       78
Oak Finish                        14
Walnut Finish                      9
Name: variation, dtype: int64
```

[16]:
```
dtale.show(df)
```

```
<IPython.lib.display.IFrame at 0x1e2fa7e1b00>
```

[16]:

[ ]:

[6]:
```
df.drop('date',axis=1,inplace=True)
```

[7]:
```
x=df.drop('feedback',axis=1)
```

[8]:
```
y=df['feedback']
```

[9]:
```
import nltk
```

[10]:
```
from nltk.stem.porter import PorterStemmer
from nltk.corpus import stopwords
import re
```

[12]:
```
ps=PorterStemmer()
corpus=[]

for i in range(0, len(x)):
    review=re.sub('[^A-Za-z^]', ' ',x['verified_reviews'][i])
    review=review.lower()
    review=review.split()

    review=[ps.stem(word) for word in review if not word in stopwords.
 ↪words('english')]
    review=' '.join(review)
    corpus.append(review)
```

[13]:
```
corpus[3]
```

[13]:
```
'lot fun thing yr old learn dinosaur control light play game like categori nice
sound play music well'
```

[14]:
```
from sklearn.feature_extraction.text import␣
↪CountVectorizer,TfidfVectorizer,HashingVectorizer
```

[15]:
```
tfidf=TfidfVectorizer(max_features=1500,ngram_range=(1,3))
```

[16]:
```
X=tfidf.fit_transform(corpus).toarray()
```

[57]:
```
X
```

[57]:
```
array([[0.        , 0.        , 0.        , ..., 0.        , 0.        ,
        0.        ],
       [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
        0.        ],
       [0.        , 0.18274227, 0.        , ..., 0.        , 0.        ,
        0.        ],
       ...,
       [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
        0.        ],
       [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
        0.        ],
       [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
        0.        ]])
```

[42]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=0)
```

[43]:
```
x_train.shape,x_test.shape
```

[43]:
```
((2205, 1500), (945, 1500))
```

[19]:
```
X.shape
```

[19]:
```
(3150, 1500)
```

[20]:
```
tfidf.get_feature_names()[:20]
```

[20]:
```
['abil',
 'abl',
 'abl listen',
 'abl play',
 'abl see',
 'abl set',
 'abl tell',
 'abl use',
 'abl watch',
 'absolut',
 'absolut love',
 'accent',
 'access',
 'account',
```

```
 'across',
 'across room',
 'act',
 'activ',
 'actual',
 'ad']
```

[21]:
```
tfidf.get_params()
```

[21]:
```
{'analyzer': 'word',
 'binary': False,
 'decode_error': 'strict',
 'dtype': numpy.float64,
 'encoding': 'utf-8',
 'input': 'content',
 'lowercase': True,
 'max_df': 1.0,
 'max_features': 1500,
 'min_df': 1,
 'ngram_range': (1, 3),
 'norm': 'l2',
 'preprocessor': None,
 'smooth_idf': True,
 'stop_words': None,
 'strip_accents': None,
 'sublinear_tf': False,
 'token_pattern': '(?u)\\b\\w\\w+\\b',
 'tokenizer': None,
 'use_idf': True,
 'vocabulary': None}
```

[22]:
```
count_df=pd.DataFrame(x_train,columns=tfidf.get_feature_names())
```

[23]:
```
count_df
```

[23]:
```
      abil  abl  abl listen  abl play  abl see  abl set  abl tell  abl use  \
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
2      0.0  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      0.0
4      0.0  0.0         0.0       0.0      0.0      0.0       0.0      0.0
...    ...  ...         ...       ...      ...      ...       ...      ...
2200   0.0  0.0         0.0       0.0      0.0      0.0       0.0      0.0
2201   0.0  0.0         0.0       0.0      0.0      0.0       0.0      0.0
2202   0.0  0.0         0.0       0.0      0.0      0.0       0.0      0.0
2203   0.0  0.0         0.0       0.0      0.0      0.0       0.0      0.0
2204   0.0  0.0         0.0       0.0      0.0      0.0       0.0      0.0

      abl watch  absolut  ...  wrong  yard   ye  year  year old  \
```

```
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
2          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
4          0.0      0.0  ...    0.0   0.0  0.0   0.0       0.0
...        ...      ...  ...    ...   ...  ...   ...       ...
2200       0.0      0.0  ...    0.0   0.0  0.0   0.0       0.0
2201       0.0      0.0  ...    0.0   0.0  0.0   0.0       0.0
2202       0.0      0.0  ...    0.0   0.0  0.0   0.0       0.0
2203       0.0      0.0  ...    0.0   0.0  0.0   0.0       0.0
2204       0.0      0.0  ...    0.0   0.0  0.0   0.0       0.0

      year old love  yell  yet  youtub  youtub video
0               0.0   0.0  0.0     0.0           0.0
1               0.0   0.0  0.0     0.0           0.0
2               0.0   0.0  0.0     0.0           0.0
3               0.0   0.0  0.0     0.0           0.0
4               0.0   0.0  0.0     0.0           0.0
...             ...   ...  ...     ...           ...
2200            0.0   0.0  0.0     0.0           0.0
2201            0.0   0.0  0.0     0.0           0.0
2202            0.0   0.0  0.0     0.0           0.0
2203            0.0   0.0  0.0     0.0           0.0
2204            0.0   0.0  0.0     0.0           0.0

[2205 rows x 1500 columns]
```

[28]:
```
#output variable immbalence we need to balence the data
import imblearn
from imblearn.over_sampling import SMOTE
```

[47]:
```
from sklearn.naive_bayes import MultinomialNB
nm=MultinomialNB()
```

[41]:
```
nm.fit(x_train,y_train)
```

[41]:
```
MultinomialNB()
```

[29]:
```
pred=nm.predict(x_test)
```

[30]:
```
from sklearn.metrics import␣
 ↪classification_report,confusion_matrix,accuracy_score
```

[31]:
```
print(confusion_matrix(pred,y_test))
```

```
[[  3   0]
 [ 72 870]]
```

[32]:
```
print(classification_report(pred,y_test))
```

```
              precision    recall  f1-score   support

           0       0.04      1.00      0.08         3
           1       1.00      0.92      0.96       942

    accuracy                           0.92       945
   macro avg       0.52      0.96      0.52       945
weighted avg       1.00      0.92      0.96       945
```

[33]:
```
print(accuracy_score(pred,y_test))
```

```
0.9238095238095239
```

[34]:
```
#apply the Hyperperameter tunning to get the best param ang good accuracy
import numpy as np
n_estimators=[int(x) for x in np.linspace(start=100,stop=2000,num=20)]
max_features=['auto','sqrt','log2']
max_depth=[int(x) for x in np.linspace(10,1000,10)]
min_smaple_split=[1,3,5,7,9]
min_sample_leaf=[1,2,3,4,5,6]
random_grid={
    'n_estimators': n_estimators,
    'max_depth': max_depth,
    'max_fetures': max_features,
    'min_smaple_split':min_smaple_split,
    'min_sample_leaf': min_sample_leaf,
    'criterion': ['gini','entropy']
}
```

[ ]:
```
random grid
```

[36]:
```
from sklearn.model_selection import RandomizedSearchCV
```

[37]:
```
rcv=RandomizedSearchCV
```

[38]:
```
raj=rcv(estimator=MultinomialNB,param_distributions=random_grid,random_state=0.
↪3,cv=3,verbose=2,n_iter=-1)
```

[ ]:
```
rcv=RandomizedSearchCV(estimator=nm,param_distributions=random_grid,cv=3,n_iter=100,n_jobs=-1,
```

[52]:
```

```

[ ]:
```

```

[ ]:
```

```