|  |
| --- |
| Python coding for Sentiment Analysis |
| import pandas as pd  import numpy as np  from sklearn.linear\_model import LinearRegression  from sklearn.model\_selection import train\_test\_split  from sklearn.feature\_extraction.text import CountVectorizer  from sklearn.metrics import accuracy\_score  df = pd.read\_csv('overallProduct.csv')  df1 = df[['name','reviewstext','reviewsrating']]  df1.groupby('name').count()  df1.groupby('name').mean().sort\_values(["reviewsrating"],ascending=False)  #convert text to string  for i in range(0,len(df1)-1):      if type(df1.iloc[i]['reviewstext']) != str:          df1.iloc[i]['reviewstext'] = str(df1.iloc[i]['reviewstext'])  #Drop rating = 3  amazon = df1[df1['reviewsrating'] != 3]  #Assign 1 and 0 for rating >= 4 and <= 2  def sentiment(n):      if n >= 4:          return 1      else: return 0  amazon['sentiment'] = amazon.loc[:,'reviewsrating'].apply(sentiment)  amazon.head()  def alltogeter (row):      return row['name'] + ' '+ row['reviewstext']  amazon['name\_text'] = amazon.apply(alltogeter, axis=1)  amazon.head()  X = amazon['name\_text']  y = amazon['sentiment']  #75% Train and 25%test  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, random\_state=0)  workcount = CountVectorizer()  numTFTR = workcount.fit\_transform(X\_train)  numTFtest = workcount.transform(X\_test)  model = LogisticRegression()  model.fit(numTF, y\_train)  predict = model.predict(numTFtest)  for i in predict:      print(i, end=" ")  accuracy\_score(y\_test, predict) |

|  |
| --- |
| Python coding for Word Count |
| import pandas as pd  import numpy as np  from sklearn.feature\_extraction.text import CountVectorizer  df = pd.read\_csv('overallProduct.csv')  df2 = df[['name','reviewstext','reviewsrating']]  df2  df2[df2['name']=='Amazon Echo Show Alexa-enabled Bluetooth Speaker with 7" Screen']  df2.dtypes  df2[['reviewsrating']] = df2[['reviewsrating']].astype(str)  df2.dtypes  array = ['4','5']  df2 = df2.loc[(df2['name'] == 'Amazon Echo Show Alexa-enabled Bluetooth Speaker with 7" Screen') & df2['reviewsrating'].isin(array)]  df2['reviewstext']= df2['reviewstext'].str.lower()  df2  df2['reviewstext']= df2['reviewstext'].str.replace('\W+', ' ',regex =True)  df3 = df2[['reviewstext']]  df3['reviewstext']= df3['reviewstext'].str.replace('\d+', '',regex =True)  dicto = {'äôt':'','äôve':'','äù':'','äú':'','äúalexa':'','äúbest':'','äúdropping':'','äúshow':'','äústar':'','äúthings':'','äôm':'','äôs':'',           'äî':'','äîbadly':'','äîespecially':'','äò':'','äôd':'','äôll':'','äôre':''}  df3['reviewstext']= df3['reviewstext'].replace(dicto,regex = True)  df3  vectorizer = CountVectorizer()  matrix = vectorizer.fit\_transform(df3['reviewstext'])  matrix  matrix.toarray()  print(vectorizer.get\_feature\_names())  counts = pd.DataFrame(matrix.toarray(),                        columns=vectorizer.get\_feature\_names())  s = counts.sum(axis = 0)  s = s.sort\_values(ascending=False)  wordscount = s.head(30)  wordscount.to\_csv('wordcount.csv',index = True) |