Decision Tree AFFR

March 28, 2020

[1]:
```
import pyforest
lazy_imports()
```

[1]:
```
['import spacy',
 'import numpy as np',
 'from pyspark import SparkContext',
 'import nltk',
 'import sklearn',
 'import plotly.express as px',
 'import glob',
 'import bokeh',
 'import seaborn as sns',
 'import plotly as py',
 'import datetime as dt',
 'from openpyxl import load_workbook',
 'from sklearn import svm',
 'import dash',
 'import pandas as pd',
 'import os',
 'from sklearn.ensemble import GradientBoostingClassifier',
 'import altair as alt',
 'from sklearn.feature_extraction.text import TfidfVectorizer',
 'from pathlib import Path',
 'from sklearn.preprocessing import OneHotEncoder',
 'import tqdm',
 'from sklearn.model_selection import train_test_split',
 'import keras',
 'import statistics',
 'import tensorflow as tf',
 'from sklearn.ensemble import RandomForestRegressor',
 'import re',
 'import matplotlib as mpl',
 'from dask import dataframe as dd',
 'import matplotlib.pyplot as plt',
 'import gensim',
 'from sklearn.ensemble import GradientBoostingRegressor',
 'from sklearn.ensemble import RandomForestClassifier',
```

```
 'import pydot',
 'import sys',
 'from sklearn.manifold import TSNE',
 'import plotly.graph_objs as go',
 'import pickle']
```

[2]:
```
import sqlite3
```

[3]:
```
con = sqlite3.connect("Downloads/Datasets/amazon-fine-food-reviews/database.
↪sqlite")
```

[4]:
```
con
```

[4]:
```
<sqlite3.Connection at 0x7fbbdd2c28f0>
```

[5]:
```
database=pd.read_sql_query("select * from reviews where Score<>3 limit␣
↪10005",con)
```

```
<IPython.core.display.Javascript object>
```

[6]:
```
def partitions(x):
    if x>3:
        return 1
    return 0
```

[7]:
```
database
```

[7]:

	Id	ProductId	UserId	ProfileName	\
0	1	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	
1	2	B00813GRG4	A1D87F6ZCVE5NK	dll pa	
2	3	B000LQOCH0	ABXLMWJIXXAIN	Natalia Corres "Natalia Corres"	
3	4	B000UA0QIQ	A395BORC6FGVXV	Karl	
4	5	B006K2ZZ7K	A1UQRSCLF8GW1T	Michael D. Bigham "M. Wassir"	
…	…	…	…	…	
10000	10927	B000EM6PC6	A39UIJ7NSFP3RG	Rick the Camera Man	
10001	10928	B000EM6PC6	AFNJIJ3ZODAA4	Shelley P	
10002	10929	B000EM6PC6	AXFC46EC13J4Z	squaredancer	
10003	10930	B000EM6PC6	A18TAR586WZNLC	kitkat	
10004	10931	B000EM6PC6	A1J205ZK25TZ6W	kez panel	

	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time	\
0	1	1	5	1303862400	
1	0	0	1	1346976000	
2	1	1	4	1219017600	
3	3	3	2	1307923200	
4	0	0	5	1350777600	

```
…                       …                       …    …          …
10000                   0                       0    5  1348876800
10001                   0                       0    5  1348790400
10002                   0                       0    5  1348185600
10003                   0                       0    5  1348099200
10004                   0                       0    5  1335657600

                                            Summary  \
0                                Good Quality Dog Food
1                                    Not as Advertised
2                                "Delight" says it all
3                                       Cough Medicine
4                                          Great taffy
…                                                  …
10000              Loose Tea is cheaper than Tea Bags
10001  Top quality product for hard-to-find loose tea
10002                                        loose tea
10003                         English tradition goes on
10004      a few dry grams makes 2 liters of hot tea!

                                                    Text
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…
…                                                    …
10000  As I said, loose Tea is cheaper than Tea Bags,…
10001  My family makes a lot of ice tea during the su…
10002  This is the only way to make good good iced te…
10003  Being married to a person from England, as mos…
10004  I use a little plastic shot glass to brew 2 li…

[10005 rows x 10 columns]
```

[8]:
```
database.Score=database.Score.map(partitions)
```

[9]:
```
database.Score.value_counts()
```

[9]:
```
1    8326
0    1679
Name: Score, dtype: int64
```

[10]:
```
database[database.HelpfulnessNumerator<=database.HelpfulnessDenominator]
```

[10]:
```
          Id   ProductId          UserId                      ProfileName  \
0          1  B001E4KFG0  A3SGXH7AUHU8GW                       delmartian
```

```
1          2  B00813GRG4  A1D87F6ZCVE5NK                           dll pa
2          3  B000LQOCH0   ABXLMWJIXXAIN  Natalia Corres "Natalia Corres"
3          4  B000UA0QIQ  A395BORC6FGVXV                             Karl
4          5  B006K2ZZ7K  A1UQRSCLF8GW1T    Michael D. Bigham "M. Wassir"
…        …          …             …                                    …
10000  10927  B000EM6PC6  A39UIJ7NSFP3RG             Rick the Camera  Man
10001  10928  B000EM6PC6   AFNJIJ3ZODAA4                         Shelley P
10002  10929  B000EM6PC6   AXFC46EC13J4Z                     squaredancer
10003  10930  B000EM6PC6  A18TAR586WZNLC                           kitkat
10004  10931  B000EM6PC6  A1J205ZK25TZ6W                         kez panel

       HelpfulnessNumerator  HelpfulnessDenominator  Score        Time  \
0                         1                       1      1  1303862400
1                         0                       0      0  1346976000
2                         1                       1      1  1219017600
3                         3                       3      0  1307923200
4                         0                       0      1  1350777600
…                       …                       …    …          …
10000                     0                       0      1  1348876800
10001                     0                       0      1  1348790400
10002                     0                       0      1  1348185600
10003                     0                       0      1  1348099200
10004                     0                       0      1  1335657600

                                        Summary  \
0                          Good Quality Dog Food
1                              Not as Advertised
2                          "Delight" says it all
3                                 Cough Medicine
4                                    Great taffy
…                                              …
10000              Loose Tea is cheaper than Tea Bags
10001  Top quality product for hard-to-find loose tea
10002                                  loose tea
10003                     English tradition goes on
10004      a few dry grams makes 2 liters of hot tea!

                                                    Text
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…
…                                                    …
10000  As I said, loose Tea is cheaper than Tea Bags,…
10001  My family makes a lot of ice tea during the su…
10002  This is the only way to make good good iced te…
```

```
10003  Being married to a person from England, as mos…
10004  I use a little plastic shot glass to brew 2 li…

[10005 rows x 10 columns]
```

[11]:
```
sorted_database=database.sort_values("ProductId")
```

[12]:
```
drop_duplicates_database=sorted_database.
↪drop_duplicates(["ProductId","UserId","Time","Text"])
```

[13]:
```
drop_duplicates_database=drop_duplicates_database.sort_values("Time")
```

[18]:
```
drop_duplicates_database
```

[18]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time	Summary
1146	1245	B00002Z754	A29Z5PI9BW2PU3	Robbie	7	7	1	961718400	Great Product
1145	1244	B00002Z754	A3B8RCEI0FXFI6	B G Chase	10	10	1	962236800	WOW Make your own 'slickers' !
7427	8111	B0000EIE2Z	A3M174IC0VXOS2	Gail Cooke	3	3	1	1075420800	BEST BLUEBERRIES
3481	3783	B00016UX0K	AF1PV3DIC0XM7	Robert Ashton	1	2	1	1081555200	Classic Condiment
6790	7432	B0001E1IME	A2IKCTD1I73PLW	Adeba	2	8	1	1083456000	amazon monopoly/ripoff
…	…	…	…	…	…	…	…	…	…
8254	9031	B006N3IG4K	A3GFZIL1E0Z5V8	bloomen1	0	0	1	1351209600	
7620	8322	B003VXFK44	A3GFZIL1E0Z5V8	bloomen1	0	0	1	1351209600	
7451	8135	B0019GVYR2	ACSO5EDO1UMZ5	SeekingBodhi	0	0	0	1351209600	
1005	1089	B004FD13RW	A1BPLP0BKERV	Paul	0	0	1	1351209600	
5259	5703	B009WSNWC4	AMP7K1O84DH1T	ESTY	0	0	1	1351209600	

```
8254  Rodeo Drive is Crazy Good Coffee!
7620  Rodeo Drive is Crazy Good Coffee!
7451                            NEWSFLASH
1005                      It is awesome.
5259                            DELICIOUS

                                                  Text
1146  This was a really good idea and the final prod…
1145  I just received my shipment and could hardly w…
7427  In the winter when fresh blueberries exceed ou…
3481  Mae Ploy Sweet Chili Sauce is becoming a stand…
6790  love the snack.  wanted to buy a bunch.<p>ship…
…                                                   …
8254  Rodeo Drive is my absolute favorite and I'm re…
7620  Rodeo Drive is my absolute favorite and I'm re…
7451  I just called Bob's Red Mill customer service …
1005  My partner is very happy with the tea, and is …
5259  Purchased this product at a local store in NY …

[9995 rows x 10 columns]
```

[14]:
```python
def preprocess_text(sentence):
    def remove_html(sentence):
        html_tag_re_obj = re.compile('<.*>?')
        return re.sub(html_tag_re_obj, ' ', sentence)

    def remove_punctuations(sentence):
        cleaned_sentence = re.sub(r'[^a-zA-Z]', r' ', sentence)
        return cleaned_sentence

    def decontracted(phrase):
        # specific
        phrase = re.sub(r"won\'t", "will not", phrase)
        phrase = re.sub(r"can\'t", "can not", phrase)

        # general
        phrase = re.sub(r"n\'t", " not", phrase)
        phrase = re.sub(r"\'re", " are", phrase)
        phrase = re.sub(r"\'s", " is", phrase)
        phrase = re.sub(r"\'d", " would", phrase)
        phrase = re.sub(r"\'ll", " will", phrase)
        phrase = re.sub(r"\'t", " not", phrase)
        phrase = re.sub(r"\'ve", " have", phrase)
        phrase = re.sub(r"\'m", " am", phrase)
        return phrase

    from bs4 import BeautifulSoup
```

```
    import re
    #from nltk.corpus import stopwords
    import nltk
    from tqdm import tqdm
    from nltk.stem import SnowballStemmer
    from nltk.corpus import stopwords
    stopwords = stopwords.words('english')
    stemmer = SnowballStemmer('english')
    stopwords = set(stopwords)
    stopwords.remove('not')

    cleaned_corpus = []
    for doc in sentence:
        cleaned_doc_1 = remove_html(doc)
        cleaned_doc_2 = remove_punctuations(doc)
        cleaned_doc_2 = decontracted(cleaned_doc_2)
        cleaned_corpus.append(cleaned_doc_2)
    count = 0

    filtered_corpus = list(map(lambda doc: ' '.join(list(filter(lambda word:␣
↪True if word not in stopwords else False\
                                                                , doc.
↪split())))),cleaned_corpus))
    process_text = list(map(lambda doc: ' '.join(list(map(stemmer.stem, doc.
↪split())))),filtered_corpus))

    return process_text
```

[15]:
```
%%time
process_text=preprocess_text(drop_duplicates_database.Text)
```

```
CPU times: user 8.17 s, sys: 235 ms, total: 8.4 s
Wall time: 14 s
```

[16]:
```
process_text[1]
```

[16]:
```
'i receiv shipment could hard wait tri product we love quot slicker quot call
instead sticker remov easili my daughter design sign print revers use car window
they print beauti the print shop program i go lot fun product window everywher
surfac like tv screen comput monitor'
```

[19]:
```
X=process_text
Y=drop_duplicates_database.Score
```

[21]:
```
x_train,x_test,y_train,y_test=train_test_split(X,Y,test_size=0.3)
print(len(x_train))
print(len(x_test))
```

```
print(len(y_train))
print(len(y_test))
```

```
<IPython.core.display.Javascript object>


6996
2999
6996
2999
```

[22]:
```
from sklearn.feature_extraction.text import CountVectorizer
```

[23]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
```

[26]:
```
count_vectorizer=CountVectorizer(min_df=10)
```

[27]:
```
count_vectorizer.fit(x_train)
```

[27]:
```
CountVectorizer(analyzer='word', binary=False, decode_error='strict',
                dtype=<class 'numpy.int64'>, encoding='utf-8', input='content',
                lowercase=True, max_df=1.0, max_features=None, min_df=10,
                ngram_range=(1, 1), preprocessor=None, stop_words=None,
                strip_accents=None, token_pattern='(?u)\\b\\w\\w+\\b',
                tokenizer=None, vocabulary=None)
```

[29]:
```
bow_features=count_vectorizer.get_feature_names()
print(bow_features)
```

```
['abil', 'abl', 'about', 'absolut', 'absorb', 'accept', 'accid', 'accord',
'account', 'acerola', 'ach', 'acid', 'acquir', 'across', 'activ', 'actual',
'ad', 'add', 'addict', 'addit', 'address', 'adjust', 'admit', 'adopt', 'ador',
'adult', 'advanc', 'advantag', 'advertis', 'advic', 'affect', 'afford',
'afraid', 'after', 'afternoon', 'aftertast', 'afterward', 'again', 'agav',
'age', 'ago', 'agre', 'ahead', 'aid', 'air', 'airtight', 'aisl', 'alcohol',
'all', 'allerg', 'allergen', 'allergi', 'allow', 'almond', 'almost', 'alon',
'along', 'alot', 'alreadi', 'also', 'alter', 'altern', 'although', 'aluminum',
'alway', 'am', 'amaz', 'amazon', 'amber', 'america', 'american', 'among',
'amount', 'an', 'and', 'ani', 'anim', 'anni', 'annoy', 'anoth', 'answer',
'anti', 'anticip', 'antioxid', 'anymor', 'anyon', 'anyth', 'anytim', 'anyway',
'anywher', 'apart', 'appar', 'appeal', 'appear', 'appl', 'appli', 'appreci',
'approach', 'approxim', 'are', 'area', 'argentina', 'arm', 'aroma', 'aromat',
'around', 'arriv', 'artifici', 'as', 'asian', 'asid', 'ask', 'aspect', 'associ',
'assort', 'assum', 'assur', 'at', 'ate', 'attach', 'attempt', 'attent',
'attract', 'audio', 'authent', 'auto', 'automat', 'avail', 'averag', 'avoid',
'aw', 'awak', 'awar', 'away', 'awesom', 'awhil', 'babi', 'back', 'bacon', 'bad',
'bag', 'bake', 'baker', 'balanc', 'ball', 'banana', 'bar', 'barbecu', 'barbequ',
'bare', 'bargain', 'barley', 'base', 'basi', 'basic', 'basket', 'batch',
```

'bathroom', 'batter', 'bbq', 'be', 'bean', 'bear', 'beat', 'beauti', 'becam',
'becaus', 'becom', 'bed', 'beef', 'been', 'beer', 'befor', 'beg', 'began',
'begin', 'behind', 'belgian', 'believ', 'belli', 'benefit', 'berri', 'besid',
'best', 'bet', 'better', 'betti', 'beverag', 'bewar', 'beyond', 'big', 'bigger',
'biggest', 'bill', 'birthday', 'biscotti', 'biscuit', 'bisquick', 'bit', 'bite',
'bitter', 'black', 'blackberri', 'bland', 'blend', 'blender', 'blood', 'blow',
'blue', 'blueberri', 'bob', 'bodi', 'boil', 'bold', 'bolder', 'bone', 'bonus',
'book', 'boost', 'bore', 'born', 'both', 'bother', 'bottl', 'bottom', 'bought',
'bowl', 'box', 'boy', 'boyfriend', 'bpa', 'br', 'brand', 'bread', 'break',
'breakfast', 'breast', 'breath', 'breed', 'brew', 'brewer', 'bright', 'bring',
'broccoli', 'broke', 'broken', 'broth', 'brother', 'brought', 'brown', 'browni',
'brush', 'btw', 'bubbl', 'buck', 'bud', 'budget', 'buffalo', 'bug', 'build',
'bulk', 'bump', 'bunch', 'burger', 'burn', 'burnt', 'busi', 'but', 'butter',
'butteri', 'buy', 'buyer', 'by', 'cafe', 'caffein', 'cajun', 'cake', 'cal',
'calcium', 'california', 'call', 'calm', 'calori', 'came', 'camp', 'campbel',
'can', 'cancel', 'candi', 'cane', 'canist', 'cannot', 'canola', 'cant', 'cap',
'car', 'caramel', 'carb', 'carbohydr', 'carbon', 'card', 'cardboard', 'care',
'caribou', 'carri', 'carrot', 'carton', 'case', 'cashew', 'cat', 'catch',
'caught', 'caus', 'caution', 'cayenn', 'celeri', 'celiac', 'cent', 'center',
'cereal', 'certain', 'certifi', 'chain', 'challeng', 'chamomil', 'chanc',
'chang', 'charact', 'charg', 'cheap', 'cheaper', 'cheapest', 'check', 'cheddar',
'chees', 'chef', 'chemic', 'cherri', 'chew', 'chewi', 'chia', 'chicken',
'child', 'childhood', 'children', 'chili', 'chill', 'china', 'chines', 'chip',
'chocol', 'chocolatey', 'chocolati', 'choic', 'choke', 'cholesterol', 'choos',
'chop', 'chose', 'chowder', 'christma', 'chunk', 'cider', 'cilantro',
'cinnamon', 'citi', 'citrus', 'claim', 'clam', 'class', 'classic', 'clean',
'clear', 'click', 'close', 'closer', 'closest', 'club', 'clump', 'co', 'coast',
'coat', 'coco', 'cocoa', 'coconut', 'coff', 'coffe', 'coke', 'cola', 'colada',
'cold', 'collect', 'colleg', 'colombian', 'color', 'columbian', 'com', 'combin',
'combo', 'come', 'comfort', 'comment', 'commerci', 'common', 'communiti',
'compani', 'compar', 'comparison', 'competit', 'complain', 'complaint',
'complet', 'complex', 'compliment', 'compromis', 'comput', 'con', 'concentr',
'concern', 'condens', 'condit', 'confid', 'confirm', 'confus', 'connect',
'connoisseur', 'conscious', 'consid', 'consider', 'consist', 'constant',
'constip', 'consum', 'consumpt', 'contact', 'contain', 'content', 'continu',
'control', 'conveni', 'convent', 'convert', 'convinc', 'cook', 'cookbook',
'cooki', 'cool', 'cooler', 'corn', 'corner', 'correct', 'cost', 'costa',
'costco', 'cough', 'could', 'count', 'counter', 'countri', 'coupl', 'coupon',
'cours', 'cover', 'cow', 'crack', 'cracker', 'cramp', 'cranberri', 'crap',
'crave', 'crazi', 'cream', 'creamer', 'creami', 'creat', 'credit', 'creme',
'cri', 'crisp', 'crispi', 'critic', 'crocker', 'crumb', 'crumbl', 'crunch',
'crunchi', 'crush', 'crust', 'crystal', 'cube', 'cup', 'cupboard', 'cure',
'curious', 'curl', 'current', 'curri', 'custom', 'cut', 'cute', 'dad', 'daili',
'dairi', 'damag', 'dark', 'darker', 'darn', 'date', 'daughter', 'day', 'de',
'dead', 'deal', 'decad', 'decaf', 'decaff', 'decaffein', 'decent', 'decid',
'decis', 'decor', 'deep', 'defin', 'definit', 'degre', 'dehydr', 'delic',
'delici', 'delight', 'deliv', 'deliveri', 'dens', 'dent', 'depend', 'describ',
'descript', 'desert', 'deserv', 'design', 'desir', 'desk', 'desper', 'despit',

'dessert', 'detail', 'detect', 'determin', 'develop', 'devic', 'devour',
'diabet', 'diagnos', 'diamond', 'diarrhea', 'dice', 'did', 'didn', 'didnt',
'die', 'diet', 'dietari', 'differ', 'difficult', 'digest', 'dijon', 'dilut',
'dinner', 'dip', 'direct', 'dirt', 'dirti', 'disappear', 'disappoint',
'discontinu', 'discount', 'discov', 'diseas', 'disgust', 'dish', 'dislik',
'dispos', 'disposakup', 'dissolv', 'distinct', 'distribut', 'do', 'doctor',
'doe', 'doesnt', 'dog', 'dollar', 'don', 'done', 'dont', 'donut', 'door',
'dose', 'doubl', 'doubt', 'dough', 'downsid', 'dozen', 'dr', 'drain', 'drank',
'draw', 'drawback', 'dress', 'dri', 'drink', 'drinkabl', 'drinker', 'drip',
'drive', 'driver', 'drop', 'due', 'dump', 'dunkin', 'durabl', 'dust', 'each',
'eager', 'ear', 'earl', 'earli', 'earlier', 'earth', 'eas', 'easi', 'easier',
'easili', 'eat', 'eaten', 'eater', 'econom', 'edg', 'edibl', 'edit', 'effect',
'effort', 'egg', 'eight', 'either', 'electrolyt', 'elimin', 'els', 'elsewher',
'em', 'email', 'emerg', 'emeril', 'empti', 'end', 'energi', 'england',
'english', 'enhanc', 'enjoy', 'enough', 'ensur', 'entir', 'equal', 'equival',
'error', 'escap', 'especi', 'espresso', 'essenti', 'etc', 'europ', 'european',
'even', 'event', 'eventu', 'ever', 'everi', 'everybodi', 'everyday', 'everyon',
'everyth', 'everywher', 'evo', 'exact', 'exampl', 'exceed', 'excel', 'except',
'excess', 'excit', 'exclus', 'exist', 'exot', 'expand', 'expect', 'expens',
'experi', 'experienc', 'expert', 'expir', 'explain', 'expos', 'extra',
'extract', 'extrem', 'eye', 'fabul', 'face', 'fact', 'factor', 'factori',
'fail', 'fair', 'fake', 'fall', 'fals', 'famili', 'familiar', 'famous', 'fan',
'fanci', 'fantast', 'far', 'farm', 'fashion', 'fast', 'faster', 'fat', 'father',
'fatten', 'fatti', 'favor', 'favorit', 'featur', 'fed', 'feed', 'feel', 'feet',
'fell', 'felt', 'fewer', 'fiber', 'fight', 'figur', 'fill', 'filler', 'filter',
'final', 'find', 'fine', 'finger', 'finicki', 'finish', 'fire', 'firm', 'first',
'fish', 'fit', 'five', 'fix', 'fizz', 'fizzi', 'flake', 'flat', 'flavor',
'flavorless', 'flavour', 'flax', 'flaxse', 'fli', 'floor', 'flour', 'flower',
'fluffi', 'foil', 'fold', 'folger', 'folk', 'follow', 'fond', 'food', 'fool',
'for', 'forc', 'forev', 'forget', 'forgot', 'fork', 'form', 'formula', 'fortun',
'forward', 'found', 'four', 'fraction', 'fragil', 'fragranc', 'fragrant',
'frank', 'free', 'freez', 'freezer', 'french', 'frequent', 'fresh', 'fresher',
'fri', 'friday', 'fridg', 'friend', 'from', 'front', 'frost', 'frozen',
'fructos', 'fruit', 'fruiti', 'frustrat', 'fuel', 'full', 'fuller', 'fulli',
'fun', 'function', 'funni', 'fur', 'fussi', 'futur', 'gag', 'gain', 'gallon',
'game', 'garbag', 'garden', 'garlic', 'gas', 'gave', 'gel', 'general',
'generous', 'gentl', 'gerber', 'german', 'get', 'gevalia', 'gf', 'giant',
'gift', 'ginger', 'gingerbread', 'girl', 'girlfriend', 'give', 'given', 'glad',
'glass', 'glaze', 'gluten', 'glycem', 'gmo', 'go', 'goat', 'gobbl', 'god',
'goe', 'gold', 'golden', 'gone', 'gonna', 'good', 'goodi', 'googl', 'got',
'gotta', 'gotten', 'gourmet', 'gp', 'grab', 'grade', 'gradual', 'grain',
'graini', 'gram', 'grandmoth', 'grandpa', 'grandson', 'granola', 'grant',
'granul', 'grape', 'grapefruit', 'grass', 'grate', 'gravi', 'greas', 'greasi',
'great', 'greatest', 'green', 'grew', 'grey', 'grill', 'grind', 'grinder',
'gritti', 'grocer', 'groceri', 'gross', 'ground', 'group', 'grove', 'grow',
'grown', 'guarante', 'guess', 'guest', 'guilt', 'guilti', 'gum', 'gummi', 'guy',
'habanero', 'habit', 'had', 'hair', 'half', 'hamburg', 'hand', 'handi', 'handl',
'hang', 'happen', 'happi', 'happier', 'happili', 'hard', 'harder', 'harmoni',

'harsh', 'has', 'hassl', 'hate', 'have', 'haven', 'hawaiian', 'hazelnut', 'he',
'head', 'headach', 'health', 'healthi', 'healthier', 'hear', 'heard', 'heart',
'hearti', 'heat', 'heaven', 'heavi', 'heavier', 'heavili', 'held', 'help',
'hemp', 'henc', 'her', 'herb', 'herbal', 'here', 'hershey', 'hesit', 'hey',
'hidden', 'hide', 'high', 'higher', 'highest', 'hint', 'hit', 'hodgson', 'hold',
'holder', 'hole', 'holiday', 'holist', 'home', 'homemad', 'honest', 'honey',
'hook', 'hope', 'horribl', 'hors', 'hot', 'hour', 'hous', 'household', 'how',
'howev', 'href', 'http', 'hubbi', 'huge', 'hull', 'human', 'hundr', 'hungri',
'hunt', 'hurri', 'hurt', 'husband', 'hydrat', 'hydrogen', 'iam', 'ice', 'icicl',
'idea', 'ideal', 'ident', 'if', 'ill', 'im', 'imagin', 'immedi', 'import',
'imposs', 'impress', 'improv', 'in', 'inch', 'includ', 'increas', 'incred',
'inde', 'indian', 'indic', 'individu', 'indulg', 'industri', 'inexpens',
'infant', 'infect', 'info', 'inform', 'infus', 'ingredi', 'initi', 'insan',
'insid', 'instant', 'instead', 'instruct', 'intact', 'intak', 'intend',
'intens', 'interest', 'intern', 'internet', 'intoler', 'introduc', 'involv',
'irish', 'iron', 'irrit', 'is', 'island', 'issu', 'it', 'itali', 'italian',
'item', 'ive', 'izz', 'jack', 'jalapeno', 'jam', 'jamaica', 'jamaican',
'japanes', 'jar', 'jelli', 'jerki', 'jet', 'job', 'joe', 'joke', 'joy', 'judg',
'juic', 'juici', 'jump', 'junk', 'just', 'justifi', 'kcup', 'keep', 'kept',
'kernel', 'ketchup', 'kettl', 'keurig', 'key', 'kibbl', 'kick', 'kid', 'kill',
'kind', 'kinda', 'kit', 'kitchen', 'kitti', 'kiwi', 'knew', 'knock', 'know',
'known', 'kona', 'kosher', 'la', 'lab', 'label', 'lack', 'lactos', 'ladi',
'lamb', 'larg', 'larger', 'last', 'late', 'later', 'latest', 'latt', 'laugh',
'lavazza', 'law', 'lay', 'layer', 'lb', 'lbs', 'lead', 'leaf', 'leak', 'learn',
'least', 'leav', 'lech', 'left', 'leftov', 'leg', 'lemon', 'lemonad', 'length',
'less', 'let', 'level', 'lick', 'licoric', 'lid', 'lie', 'life', 'lift',
'light', 'lighter', 'like', 'lime', 'limit', 'line', 'linger', 'link', 'lipton',
'liquid', 'liquor', 'list', 'liter', 'littl', 'live', 'liver', 'load', 'loaf',
'lobster', 'local', 'locat', 'lock', 'lol', 'lollipop', 'long', 'longer',
'look', 'loos', 'lose', 'loss', 'lost', 'lot', 'lousi', 'love', 'lover', 'low',
'lower', 'lowest', 'lowrey', 'luck', 'lucki', 'luckili', 'lunch', 'mac',
'machin', 'made', 'magic', 'mahogani', 'mail', 'main', 'maintain', 'major',
'make', 'maker', 'malt', 'maltodextrin', 'man', 'manag', 'mango', 'mani',
'manner', 'manufactur', 'mapl', 'marinad', 'mark', 'market', 'mart', 'marzano',
'match', 'mate', 'materi', 'matter', 'maxwel', 'may', 'mayb', 'mberri', 'me',
'meal', 'mean', 'meant', 'measur', 'meat', 'medic', 'medium', 'meet', 'melitta',
'mellow', 'melt', 'member', 'memori', 'mention', 'mess', 'messi', 'met',
'metal', 'method', 'mexican', 'mg', 'mic', 'microphon', 'microwav', 'mid',
'middl', 'might', 'mild', 'milder', 'mile', 'milk', 'mill', 'min', 'mind',
'mine', 'miner', 'mini', 'minim', 'minimum', 'mint', 'minus', 'minut', 'miracl',
'mislead', 'miss', 'mistak', 'mix', 'mixtur', 'mocha', 'moder', 'modifi',
'moist', 'moistur', 'molass', 'mold', 'moldi', 'mom', 'moment', 'money',
'monitor', 'month', 'mood', 'more', 'morn', 'most', 'mother', 'mountain',
'mouth', 'move', 'movi', 'mrs', 'msg', 'much', 'muffin', 'mug', 'multi',
'multipl', 'munch', 'mushi', 'mushroom', 'must', 'mustard', 'my', 'namast',
'name', 'nana', 'nasti', 'natur', 'nd', 'near', 'nearbi', 'necessari', 'need',
'needless', 'negat', 'neighbor', 'neighborhood', 'neither', 'nervous', 'net',
'never', 'new', 'newman', 'news', 'next', 'nice', 'night', 'no', 'nois', 'non',

'none', 'noodl', 'normal', 'nose', 'not', 'note', 'noth', 'notic', 'now',
'nowher', 'number', 'numer', 'nut', 'nutrient', 'nutrit', 'nutriti', 'nutti',
'ny', 'oat', 'oatmeal', 'obtain', 'obvious', 'occas', 'occasion', 'occur',
'odd', 'of', 'offer', 'offic', 'often', 'oh', 'oil', 'oili', 'ok', 'okay',
'old', 'older', 'oldest', 'ole', 'oliv', 'omaha', 'omega', 'omg', 'on', 'onc',
'one', 'onion', 'onli', 'onlin', 'onto', 'oolong', 'open', 'opinion',
'opportun', 'oppos', 'option', 'or', 'orang', 'orangina', 'order', 'oreo',
'organ', 'origin', 'other', 'otherwis', 'ounc', 'our', 'out', 'outrag',
'outsid', 'outstand', 'oven', 'over', 'overal', 'overnight', 'overpow',
'overpr', 'overweight', 'overwhelm', 'own', 'owner', 'oz', 'pack', 'packag',
'packet', 'page', 'paid', 'pain', 'pair', 'palat', 'pamela', 'pan', 'pancak',
'pantri', 'paper', 'parent', 'part', 'parti', 'partial', 'particular', 'pass',
'past', 'pasta', 'patient', 'patti', 'pay', 'pb', 'pea', 'peac', 'peach',
'peanut', 'pear', 'pecan', 'peel', 'peet', 'penni', 'peopl', 'pepper',
'peppermint', 'per', 'perfect', 'perform', 'perhap', 'period', 'person',
'pesticid', 'pet', 'phone', 'photo', 'physic', 'pick', 'picki', 'pickl',
'pictur', 'pie', 'piec', 'pill', 'pina', 'pinch', 'pineappl', 'pink', 'pitcher',
'pizza', 'place', 'plain', 'plan', 'planet', 'plant', 'plastic', 'plate',
'play', 'pleas', 'pleasant', 'pleasur', 'plenti', 'plug', 'plum', 'plump',
'plus', 'po', 'pocket', 'pod', 'point', 'pomegran', 'poop', 'poor', 'pop',
'popchip', 'popcorn', 'popper', 'popular', 'pork', 'portabl', 'portion',
'posit', 'possibl', 'post', 'pot', 'potassium', 'potato', 'potenti', 'pouch',
'pound', 'pour', 'powder', 'power', 'practic', 'pre', 'prefer', 'pregnant',
'premium', 'prepar', 'present', 'preserv', 'press', 'pressur', 'pretti',
'pretzel', 'prevent', 'previous', 'price', 'pricey', 'prime', 'princ', 'print',
'prior', 'pro', 'probabl', 'problem', 'process', 'produc', 'product',
'profession', 'profil', 'profit', 'program', 'promis', 'promot', 'prompt',
'pronounc', 'proper', 'pros', 'protect', 'protein', 'provid', 'puck', 'pud',
'puff', 'pull', 'pumpkin', 'punch', 'punctur', 'pup', 'puppi', 'purchas',
'pure', 'purina', 'purpos', 'purs', 'put', 'qualiti', 'quantiti', 'quart',
'quench', 'question', 'quick', 'quinoa', 'quit', 'rabbit', 'rais', 'raisin',
'ramen', 'ran', 'ranch', 'rancid', 'rang', 'rank', 'rare', 'raspberri', 'rate',
'rather', 'ratio', 'rave', 'raw', 'rd', 'reach', 'reaction', 'read', 'readi',
'readili', 'real', 'realiz', 'realli', 'reason', 'recal', 'receiv', 'recent',
'reciev', 'recip', 'recogn', 'recommend', 'reconstitut', 'record', 'recycl',
'red', 'reduc', 'refer', 'refil', 'refin', 'reflux', 'refresh', 'refriger',
'refund', 'refus', 'regard', 'regardless', 'regret', 'regul', 'regular',
'relat', 'relax', 'reliabl', 'reliev', 'remain', 'rememb', 'remind', 'remov',
'reorder', 'repeat', 'replac', 'repli', 'report', 'request', 'requir', 'reseal',
'research', 'resembl', 'reserv', 'residu', 'resist', 'resolv', 'respect',
'respond', 'respons', 'rest', 'restaur', 'restrict', 'result', 'retail',
'return', 'reus', 'review', 'reward', 'rica', 'rice', 'rich', 'richer', 'rid',
'ride', 'ridicul', 'right', 'rind', 'rins', 'rip', 'rise', 'risk', 'riviera',
'road', 'roast', 'robust', 'rock', 'rodeo', 'roll', 'room', 'root', 'rose',
'rotat', 'rough', 'round', 'routin', 'royal', 'rub', 'ruin', 'rum', 'run',
'runni', 'rush', 'rye', 'sad', 'safe', 'said', 'salad', 'sale', 'salmon',
'salsa', 'salt', 'salti', 'sam', 'same', 'sampl', 'sampler', 'san', 'sandwich',
'sardin', 'sat', 'satisfact', 'satisfi', 'satur', 'sauc', 'sausag', 'saut',

'save', 'saver', 'savor', 'savori', 'saw', 'say', 'scale', 'scare', 'scent',
'schedul', 'school', 'scienc', 'scissor', 'scoop', 'scratch', 'sea', 'seafood',
'seal', 'search', 'season', 'seattl', 'second', 'secret', 'section', 'secur',
'see', 'seed', 'seem', 'seen', 'select', 'self', 'sell', 'seller', 'seltzer',
'semi', 'send', 'senior', 'sens', 'senseo', 'sensit', 'sent', 'separ',
'serious', 'serv', 'servic', 'sesam', 'set', 'settl', 'seven', 'sever', 'shake',
'shame', 'shape', 'share', 'sharp', 'she', 'shed', 'sheet', 'shelf', 'shell',
'shelter', 'shelv', 'shine', 'shini', 'ship', 'shipment', 'shock', 'shop',
'short', 'shot', 'should', 'show', 'shower', 'shown', 'shred', 'shrimp', 'sick',
'side', 'sign', 'signific', 'similac', 'similar', 'simpl', 'simpli', 'sinc',
'singl', 'sinus', 'sip', 'sister', 'sit', 'site', 'situat', 'six', 'size',
'skeptic', 'skin', 'skinni', 'skip', 'sleep', 'slice', 'slight', 'slow',
'slowli', 'small', 'smaller', 'smallest', 'smell', 'smile', 'smoke', 'smokey',
'smoki', 'smooth', 'smoother', 'smoothi', 'snack', 'snap', 'snob', 'snow', 'so',
'soak', 'soda', 'sodium', 'soft', 'softer', 'sold', 'solid', 'solut', 'solv',
'some', 'somehow', 'someon', 'someth', 'sometim', 'somewhat', 'somewher', 'son',
'soon', 'sooo', 'soooo', 'sooth', 'sore', 'sorri', 'sort', 'sound', 'soup',
'sour', 'sourc', 'south', 'southern', 'soy', 'soybean', 'space', 'sparkl',
'speak', 'spearmint', 'special', 'specialti', 'specif', 'speed', 'spend',
'spent', 'spice', 'spici', 'spike', 'spill', 'spinach', 'spit', 'splash',
'splenda', 'split', 'spoil', 'spoon', 'sport', 'spot', 'spray', 'spread',
'spring', 'sprinkl', 'sprout', 'squar', 'squeez', 'st', 'stabl', 'stack',
'stage', 'stale', 'stand', 'standard', 'stapl', 'star', 'starbuck', 'starch',
'start', 'starter', 'stash', 'state', 'station', 'stay', 'steak', 'steep',
'stem', 'step', 'stephen', 'stevia', 'stew', 'stick', 'sticki', 'still', 'stir',
'stock', 'stomach', 'stool', 'stop', 'storag', 'store', 'stori', 'stove',
'straight', 'strang', 'strawberri', 'strength', 'stress', 'strip', 'strong',
'stronger', 'struggl', 'stuck', 'studi', 'stuf', 'stuff', 'stumbl', 'sturdi',
'style', 'subject', 'subscrib', 'subscript', 'substanc', 'substanti',
'substitut', 'subtl', 'success', 'suck', 'sucralos', 'suffer', 'suffici',
'sugar', 'sugari', 'suggest', 'suit', 'sumatra', 'summer', 'sun', 'sunday',
'sunflow', 'sunset', 'super', 'superb', 'superior', 'supermarket', 'supplement',
'suppli', 'supplier', 'support', 'suppos', 'supris', 'sure', 'surpris',
'suspect', 'swallow', 'swear', 'sweet', 'sweeten', 'sweeter', 'sweetner',
'swiss', 'switch', 'symptom', 'syrup', 'system', 'tabl', 'tablespoon', 'tablet',
'tad', 'take', 'taken', 'talk', 'tang', 'tangerin', 'tangi', 'tapioca',
'target', 'tart', 'tassimo', 'tast', 'tastebud', 'tasteless', 'tasti',
'tastier', 'tastiest', 'tax', 'tea', 'teabag', 'tear', 'teaspoon', 'teeth',
'tell', 'temperatur', 'tempt', 'temptat', 'ten', 'tend', 'tender', 'term',
'terribl', 'terrier', 'terrif', 'test', 'textur', 'th', 'thai', 'thank', 'that',
'the', 'their', 'them', 'theme', 'then', 'there', 'therefor', 'these', 'they',
'thick', 'thicken', 'thicker', 'thin', 'thing', 'think', 'thinner', 'third',
'thirst', 'this', 'thorough', 'those', 'though', 'thought', 'three', 'threw',
'thrill', 'thrive', 'throat', 'throughout', 'throw', 'thrown', 'thru', 'thumb',
'thus', 'tiger', 'tight', 'till', 'time', 'timothi', 'tin', 'tini', 'tip',
'tire', 'titl', 'to', 'toast', 'today', 'toddler', 'toffe', 'tofu', 'togeth',
'told', 'toler', 'tomato', 'ton', 'tone', 'tongu', 'too', 'took', 'tooth',
'top', 'tortilla', 'toss', 'total', 'touch', 'tough', 'toward', 'town', 'toxic',

```
'toy', 'trade', 'trader', 'tradit', 'train', 'tran', 'transact', 'transfer',
'transit', 'trap', 'trash', 'travel', 'tray', 'treat', 'tree', 'tri', 'trick',
'trip', 'tropic', 'troubl', 'true', 'truffl', 'truli', 'trust', 'truth', 'tsp',
'tub', 'tube', 'tulli', 'tummi', 'tuna', 'turkey', 'turn', 'tv', 'twenti',
'twice', 'twin', 'twist', 'two', 'type', 'typic', 'ugh', 'ultim', 'un', 'unabl',
'unbeliev', 'understand', 'unfortun', 'unhealthi', 'uniqu', 'unit', 'unless',
'unlik', 'unpleas', 'unsalt', 'unsweeten', 'until', 'unusu', 'up', 'updat',
'upon', 'upset', 'us', 'usa', 'usb', 'use', 'user', 'usual', 'vacat', 'vacuum',
'valley', 'valu', 'van', 'vanilla', 'vari', 'varieti', 'various', 'vegan',
'veget', 'vegetarian', 'veggi', 'vend', 'vendor', 'veri', 'versatil', 'version',
'versus', 'vet', 'via', 'vine', 'vinegar', 'virtual', 'visit', 'vita',
'vitamin', 'vodka', 'volum', 'vomit', 'vs', 'waffl', 'wait', 'wake', 'wal',
'walk', 'walmart', 'walnut', 'want', 'warehous', 'warm', 'warn', 'was', 'wash',
'wast', 'watch', 'watcher', 'water', 'wateri', 'watermelon', 'way', 'we',
'weak', 'weaker', 'wean', 'wear', 'weather', 'web', 'websit', 'wed', 'week',
'weekend', 'weigh', 'weight', 'weird', 'well', 'went', 'west', 'wet', 'what',
'whatev', 'wheat', 'when', 'whenev', 'where', 'whether', 'whey', 'whi', 'which',
'while', 'whim', 'whip', 'white', 'who', 'whole', 'wholesom', 'whose', 'wide',
'wife', 'wild', 'will', 'win', 'window', 'wine', 'wing', 'winner', 'winter',
'wise', 'wish', 'with', 'within', 'without', 'wolf', 'wolfgang', 'women', 'won',
'wonder', 'wont', 'wood', 'word', 'work', 'worker', 'workout', 'world', 'worri',
'wors', 'worst', 'worth', 'would', 'wow', 'wrap', 'wrapper', 'write', 'written',
'wrong', 'wrote', 'www', 'xlr', 'yeah', 'year', 'yeast', 'yellow', 'yes',
'yesterday', 'yet', 'yogurt', 'york', 'you', 'young', 'younger', 'your', 'yr',
'yuban', 'yuck', 'yum', 'yummi', 'zero', 'zip', 'zuke']
```

[30]:
```
bow_x_train=count_vectorizer.transform(x_train)
bow_x_test=count_vectorizer.transform(x_test)
print(bow_x_train.shape)
print(bow_x_test.shape)
```

```
(6996, 2348)
(2999, 2348)
```

[32]:
```
print(bow_x_train)
print(bow_x_test)
```

```
  (0, 133)      1
  (0, 386)      1
  (0, 609)      1
  (0, 896)      1
  (0, 914)      1
  (0, 1432)     1
  (0, 1629)     1
  (0, 1778)     1
  (0, 1846)     1
  (0, 1947)     1
  (0, 2033)     1
```

```
(0, 2091)     1
(1, 386)      2
(1, 490)      1
(1, 869)      1
(1, 896)      1
(1, 1010)     1
(1, 1583)     1
(1, 1719)     1
(1, 1789)     1
(1, 2033)     1
(1, 2099)     1
(2, 66)       1
(2, 371)      1
(2, 629)      1
:     :
(6994, 1353)  2
(6994, 1355)  1
(6994, 1469)  2
(6994, 1515)  1
(6994, 1557)  3
(6994, 1589)  1
(6994, 1631)  1
(6994, 1745)  1
(6994, 2002)  1
(6994, 2067)  1
(6994, 2099)  1
(6994, 2113)  1
(6994, 2126)  1
(6994, 2196)  1
(6994, 2333)  1
(6995, 134)   1
(6995, 869)   1
(6995, 950)   1
(6995, 1195)  1
(6995, 1394)  1
(6995, 1432)  1
(6995, 1466)  1
(6995, 1621)  2
(6995, 1724)  1
(6995, 2033)  1
(0, 121)      1
(0, 200)      1
(0, 386)      1
(0, 643)      1
(0, 877)      1
(0, 996)      1
(0, 1068)     2
(0, 1162)     1
```

```
  (0, 1195)     2
  (0, 1353)     2
  (0, 1394)     1
  (0, 1469)     1
  (0, 1596)     1
  (0, 2283)     1
  (1, 79)       1
  (1, 198)      1
  (1, 225)      1
  (1, 227)      1
  (1, 231)      4
  (1, 276)      1
  (1, 445)      1
  (1, 458)      1
  (1, 461)      1
  (1, 574)      1
  (1, 604)      1
  :     :
  (2997, 2111)  1
  (2997, 2146)  1
  (2997, 2211)  1
  (2997, 2237)  1
  (2997, 2281)  1
  (2997, 2317)  1
  (2998, 183)   1
  (2998, 223)   1
  (2998, 609)   1
  (2998, 639)   1
  (2998, 654)   2
  (2998, 743)   1
  (2998, 925)   1
  (2998, 994)   1
  (2998, 1068)  1
  (2998, 1188)  1
  (2998, 1193)  1
  (2998, 1394)  1
  (2998, 1534)  1
  (2998, 1549)  1
  (2998, 1745)  1
  (2998, 1809)  2
  (2998, 2033)  1
  (2998, 2062)  1
  (2998, 2099)  1
```

[33]:
```
bow_x_train=bow_x_train.toarray()
bow_x_test=bow_x_test.toarray()
print(bow_x_train)
```

```
print(bow_x_test)
```

```
[[0 0 0 … 0 0 0]
 [0 0 0 … 0 0 0]
 [0 0 0 … 0 0 0]
 …
 [0 0 0 … 0 0 0]
 [0 0 0 … 0 0 0]
 [0 0 0 … 0 0 0]]
[[0 0 0 … 0 0 0]
 [0 0 0 … 0 0 0]
 [0 0 0 … 0 0 0]
 …
 [0 0 0 … 0 0 0]
 [0 0 0 … 0 0 0]
 [0 0 0 … 0 0 0]]
```

[34]:
```
from sklearn.tree import DecisionTreeClassifier
```

[35]:
```
from sklearn.model_selection import GridSearchCV
```

[38]:
```
from sklearn.metrics import␣
↪accuracy_score,classification_report,confusion_matrix,roc_auc_score,roc_curve
```

[50]:
```
d_tree=DecisionTreeClassifier(class_weight="balanced")
parameter={"max_depth":list(np.arange(20,100,10)),"min_samples_leaf":list(np.
↪arange(7,30,2))}
```

```
<IPython.core.display.Javascript object>


<IPython.core.display.Javascript object>
```

[51]:
```
%%time
clf=GridSearchCV(d_tree,param_grid=parameter,scoring="roc_auc",cv=4,return_train_score=True)
clf.fit(bow_x_train,y_train)
```

```
CPU times: user 12min 20s, sys: 13.5 s, total: 12min 34s
Wall time: 12min 35s
```

[51]:
```
GridSearchCV(cv=4, error_score=nan,
             estimator=DecisionTreeClassifier(ccp_alpha=0.0,
                                              class_weight='balanced',
                                              criterion='gini', max_depth=None,
                                              max_features=None,
                                              max_leaf_nodes=None,
                                              min_impurity_decrease=0.0,
```

```
                                       min_impurity_split=None,
                                       min_samples_leaf=1,
                                       min_samples_split=2,
                                       min_weight_fraction_leaf=0.0,
                                       presort='deprecated',
                                       random_state=None,
                                       splitter='best'),
             iid='deprecated', n_jobs=None,
             param_grid={'max_depth': [20, 30, 40, 50, 60, 70, 80, 90],
                         'min_samples_leaf': [7, 9, 11, 13, 15, 17, 19, 21, 23,
                                       25, 27, 29]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=True,
             scoring='roc_auc', verbose=0)
```

[52]:
```
print(clf.best_params_)
print(clf.best_score_)
```

```
{'max_depth': 40, 'min_samples_leaf': 13}
0.7941074690808891
```

[53]:
```
from sklearn import metrics
```

[55]:
```
y_tr_pred=clf.predict_proba(bow_x_train)
fpr_tr,tpr_tr,threshold_tr=roc_curve(y_train,y_tr_pred[:,1])
roc_auc_tr=metrics.auc(fpr_tr,tpr_tr)
y_test_pred=clf.predict_proba(bow_x_test)
fpr,tpr,threshold=roc_curve(y_test,y_test_pred[:,1])
roc_auc_test=metrics.auc(fpr,tpr)
```

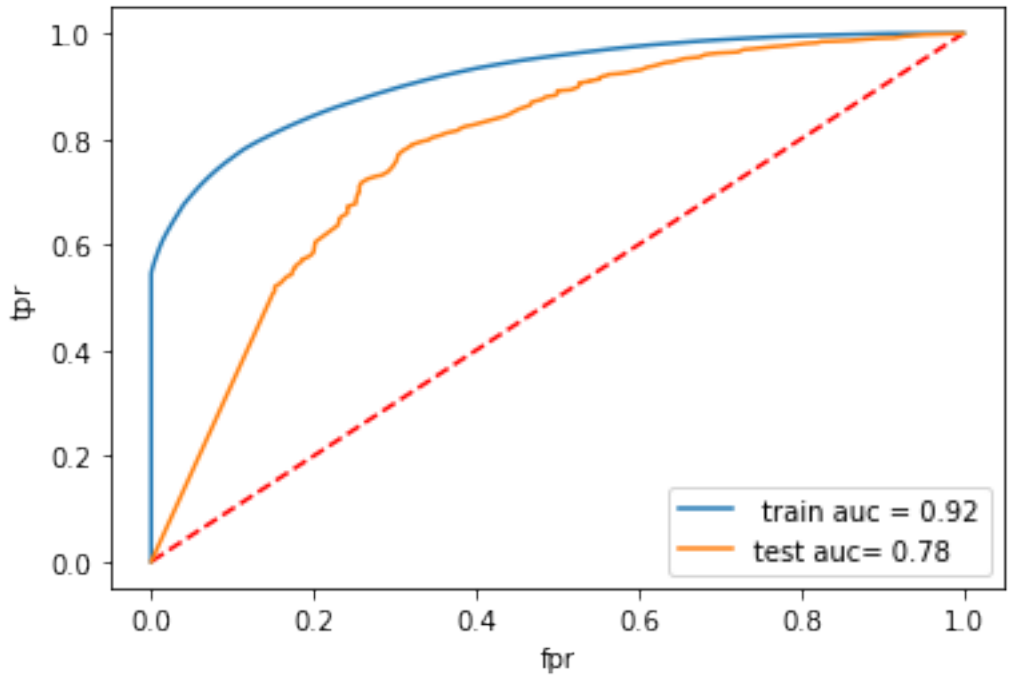
[57]:
```
plt.plot([0,1],[0,1],"r--")
plt.plot(fpr_tr,tpr_tr,label=" train auc = %0.2f"%roc_auc_tr)
plt.plot(fpr,tpr,label="test auc= %0.2f"%roc_auc_test)
plt.legend(loc="lower right")
plt.xlabel("fpr")
plt.ylabel("tpr")
plt.show()
```

```
<IPython.core.display.Javascript object>


<IPython.core.display.Javascript object>


<IPython.core.display.Javascript object>


<IPython.core.display.Javascript object>
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.Javascript object>
```



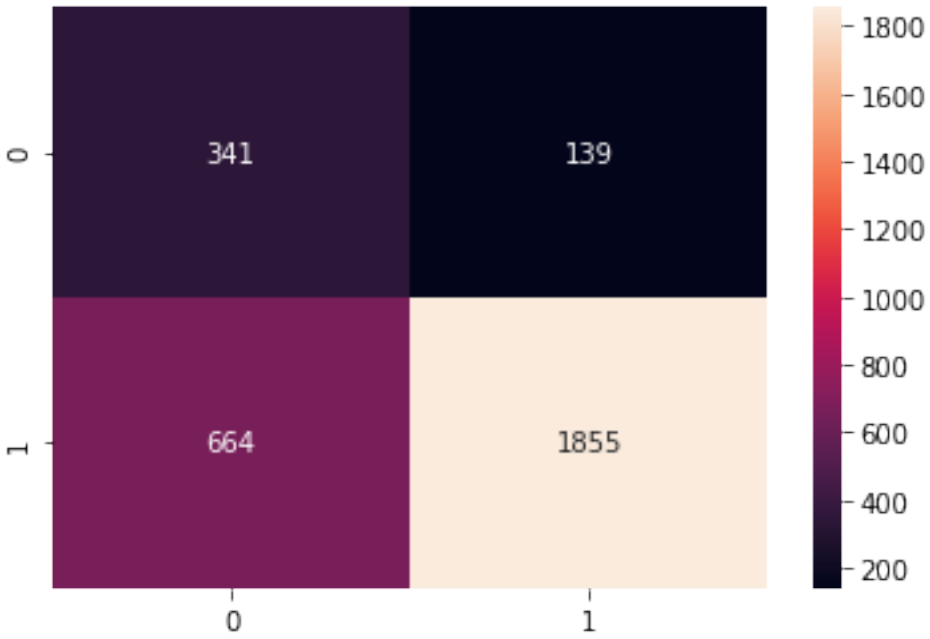
[63]:
```
y_pred=clf.predict(bow_x_test)
confusion_mat=confusion_matrix(y_test,y_pred)
print("confusion matrix:\n",confusion_mat)
class_report=classification_report(y_test,y_pred)
print("the classification reports:\n",class_report)
sns.heatmap(confusion_mat,annot=True,fmt="g")
plt.show()
```

```
confusion matrix:
 [[ 341  139]
 [ 664 1855]]
the classification reports:
              precision    recall  f1-score   support

           0       0.34      0.71      0.46       480
           1       0.93      0.74      0.82      2519

    accuracy                           0.73      2999
```

```
   macro avg       0.63      0.72      0.64      2999
weighted avg       0.84      0.73      0.76      2999
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.Javascript object>
```



[62]:
```
feature_importance=clf.best_estimator_.feature_importances_
print(feature_importance)
```

```
[0. 0. 0. … 0. 0. 0.]
```

[64]:
```
from IPython.display import Image
from sklearn.externals.six import StringIO
from sklearn.tree import export_graphviz
import pydot
import pydotplus
```

```
/home/sushil/anaconda3/lib/python3.7/site-packages/sklearn/externals/six.py:31:
FutureWarning: The module is deprecated in version 0.21 and will be removed in
version 0.23 since we've dropped support for Python 2.7. Please rely on the
official version of six (https://pypi.org/project/six/).
  "(https://pypi.org/project/six/).", FutureWarning)
```

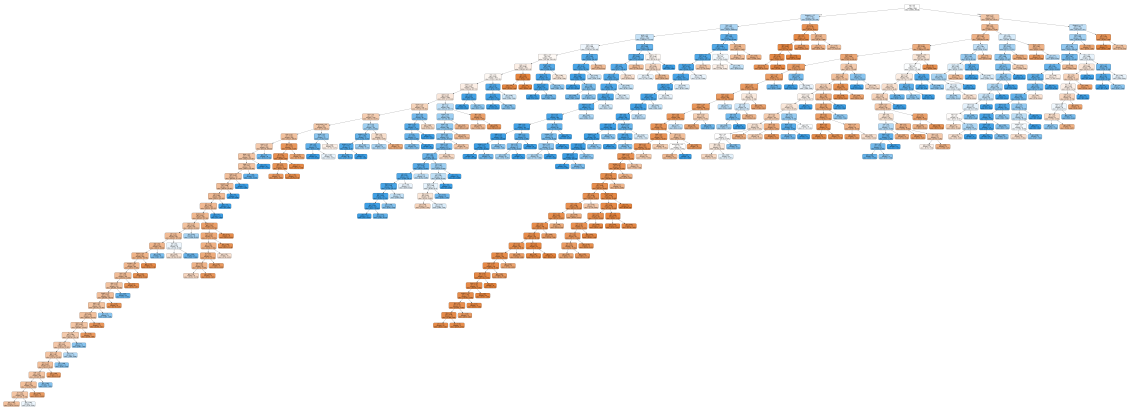
[65]:
```
from sklearn import tree
dotfile =StringIO()
tree.export_graphviz(clf.
 ↪best_estimator_,feature_names=bow_features,filled=True,rounded=True,out_file=dotfile)
graph=pydotplus.graph_from_dot_data(dotfile.getvalue())
graph.write_png("dtree.png")
```

[65]: True

[66]:
```
(graph,)=pydot.graph_from_dot_data(dotfile.getvalue())
```

[67]:
```
Image(graph.create_png())
```

[67]:



[ ]: