In [1]:
```
final1 = pd.read_csv("final.csv")
```

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

In [2]:
```
final1 = final1.drop(["Unnamed: 0"],axis=1)
final1.head(2)
```

Out[2]:

	questions_id	questions_author_id	answers_autho
0	332a511f1569444485cf7a7a556a5e54	8f6f374ffd834d258ab69d376dd998f5	36ff3b3666df400f956f8335cf53
1	332a511f1569444485cf7a7a556a5e54	8f6f374ffd834d258ab69d376dd998f5	36ff3b3666df400f956f8335cf53

In [3]:
```
x = final1['professionals_industry'].value_counts()
len(x)
```

Out[3]:
```
1200
```

In [4]:
```
final = final1[["questions_id","questions_body","questions_title","tags_tag_name"
```

In [5]:
```
final.head(2)
```

Out[5]:

	questions_id	questions_body	questions_title	tags_tag_name	professionals
0	332a511f1569444485cf7a7a556a5e54	What is a maths teacher? what is a ma...	Teacher career question	career	Mental H
1	332a511f1569444485cf7a7a556a5e54	What is a maths teacher? what is a ma...	Teacher career question	jobs	Mental H

In [6]:
```
final["professionals_industry"]=final["professionals_industry"].fillna("")
```

```
<ipython-input-6-0f05df7f9015>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  final["professionals_industry"]=final["professionals_industry"].fillna("")
```

In [7]:
```
final["tags_tag_name"]=final["tags_tag_name"].fillna("")
```

```
<ipython-input-7-e72a2c1d22ec>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  final["tags_tag_name"]=final["tags_tag_name"].fillna("")
```

In [8]:
```
# https://stackoverflow.com/a/47091490/4084039
import re

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
```

In [9]:
```
stopwords= set(['br', 'the', 'i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'our
             "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he',
             'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itse
             'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that
             'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has
             'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because
             'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'th
             'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off
             'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all
             'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than',
             's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've
             've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "di
             "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma',
             "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn'
             'won', "won't", 'wouldn', "wouldn't"])
```

In [10]:
```
# Combining all the above questions
from bs4 import BeautifulSoup
from tqdm import tqdm
preprocessed_questions_body = []
# tqdm is for printing the status bar
for sentance in tqdm(final['questions_body'].values):
    sentance = re.sub(r"http\S+", "", sentance)
    sentance = BeautifulSoup(sentance, 'lxml').get_text()
    sentance = decontracted(sentance)
    sentance = re.sub("\S*\d\S*", "", sentance).strip()
    sentance = re.sub('[^A-Za-z]+', ' ', sentance)
    # https://gist.github.com/sebleier/554280
    sentance = ' '.join(e.lower() for e in sentance.split() if e.lower() not in s
    preprocessed_questions_body.append(sentance.strip())
```

```
100%|██████████████████████████████████████████████████████████████████████████|
426927/426927 [03:13<00:00, 2204.44it/s]
```

In [11]:
```
preprocessed_questions_body[1500]
```

Out[11]:
```
'lot reform movements seem sweep public education movements varying quality sco
pe lot competing forces trying influence classroom keep becoming cynical losing
fire teaching teaching teacher education'
```

In [12]:
```
final['questions_body'] = preprocessed_questions_body
```

```
<ipython-input-12-311914f57263>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pyd
ata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
opy)
  final['questions_body'] = preprocessed_questions_body
```

In [13]:
```
final.head(2)
```

Out[13]:

	questions_id	questions_body	questions_title	tags_tag_name	professionals
0	332a511f1569444485cf7a7a556a5e54	maths teacher maths teacher useful college pro...	Teacher career question	career	Mental H
1	332a511f1569444485cf7a7a556a5e54	maths teacher maths teacher useful college pro...	Teacher career question	jobs	Mental H

In [14]:
```
# Combining all the above questions
from tqdm import tqdm
preprocessed_questions_title = []
# tqdm is for printing the status bar
for sentance in tqdm(final['questions_title'].values):
    sentance = re.sub(r"http\S+", "", sentance)
    sentance = BeautifulSoup(sentance, 'lxml').get_text()
    sentance = decontracted(sentance)
    sentance = re.sub("\S*\d\S*", "", sentance).strip()
    sentance = re.sub('[^A-Za-z]+', ' ', sentance)
    # https://gist.github.com/sebleier/554280
    sentance = ' '.join(e.lower() for e in sentance.split() if e.lower() not in s
    preprocessed_questions_title.append(sentance.strip())
```

```
100%|██████████████████████████████████████████████████████████████████████|
426927/426927 [03:19<00:00, 2134.86it/s]
```

In [15]:
```
preprocessed_questions_title[1500]
```

Out[15]:
```
'keep cynical teaching public schools'
```

In [16]:
```
final['questions_title'] = preprocessed_questions_title
```

```
<ipython-input-16-7cf2088d5da1>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pyd
ata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
opy)
  final['questions_title'] = preprocessed_questions_title
```

In [17]:
```
final.head(2)
```

Out[17]:

	questions_id	questions_body	questions_title	tags_tag_name	professionals
0	332a511f1569444485cf7a7a556a5e54	maths teacher maths teacher useful college pro...	teacher career question	career	Mental H
1	332a511f1569444485cf7a7a556a5e54	maths teacher maths teacher useful college pro...	teacher career question	jobs	Mental H

In [18]:
```
# Combining all the above questions
from tqdm import tqdm
preprocessed_tags = []
# tqdm is for printing the status bar
for sentance in tqdm(final['tags_tag_name'].values):
    sentance = re.sub(r"http\S+", "", sentance)
    sentance = BeautifulSoup(sentance, 'lxml').get_text()
    sentance = decontracted(sentance)
    sentance = re.sub("\S*\d\S*", "", sentance).strip()
    sentance = re.sub('[^A-Za-z]+', ' ', sentance)
    # https://gist.github.com/sebleier/554280
    sentance = ' '.join(e.lower() for e in sentance.split() if e.lower() not in s
    preprocessed_tags.append(sentance.strip())
```

```
  0%|
| 713/426927 [00:00<02:53, 2452.79it/s]C:\Users\sagar\anaconda3\lib\site-packag
es\bs4\__init__.py:332: MarkupResemblesLocatorWarning: "data" looks like a file
name, not markup. You should probably open this file and pass the filehandle in
to Beautiful Soup.
  warnings.warn(
  0%|▌
| 2059/426927 [00:00<02:36, 2709.48it/s]C:\Users\sagar\anaconda3\lib\site-packa
ges\bs4\__init__.py:332: MarkupResemblesLocatorWarning: "con" looks like a file
name, not markup. You should probably open this file and pass the filehandle in
to Beautiful Soup.
  warnings.warn(
100%|████████████████████████████████████████████████████████████████████████|
426927/426927 [02:56<00:00, 2415.33it/s]
```

In [19]:
```
# Combining all the above questions
from tqdm import tqdm
preprocessed_industry = []
# tqdm is for printing the status bar
for sentance in tqdm(final['professionals_industry'].values):
    sentance = re.sub(r"http\S+", "", sentance)
    sentance = BeautifulSoup(sentance, 'lxml').get_text()
    sentance = decontracted(sentance)
    sentance = re.sub("\S*\d\S*", "", sentance).strip()
    sentance = re.sub('[^A-Za-z]+', ' ', sentance)
    # https://gist.github.com/sebleier/554280
    sentance = ' '.join(e.lower() for e in sentance.split() if e.lower() not in s
    preprocessed_industry.append(sentance.strip())
```

```
100%|████████████████████████████████████████████████████████████████████████|
426927/426927 [02:49<00:00, 2521.47it/s]
```

In [20]:
```
final['tags_tag_name'] = preprocessed_tags
```

```
<ipython-input-20-bda626443de5>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  final['tags_tag_name'] = preprocessed_tags
```

In [21]:
```
final['professionals_industry'] = preprocessed_industry
```

```
<ipython-input-21-3a64261558fb>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  final['professionals_industry'] = preprocessed_industry
```

In [22]:
```
final.head(2)
```

Out[22]:

	questions_id	questions_body	questions_title	tags_tag_name	professionals
0	332a511f1569444485cf7a7a556a5e54	maths teacher maths teacher useful college pro...	teacher career question	career	mental
1	332a511f1569444485cf7a7a556a5e54	maths teacher maths teacher useful college pro...	teacher career question	jobs	mental

In [23]:
```
grp_qid_pindustry = final.groupby('questions_id')['professionals_industry'].apply
```

```
<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>

<IPython.core.display.Javascript object>

<IPython.core.display.Javascript object>

<IPython.core.display.Javascript object>

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

In [24]:
```
grp_qid_pindustry =pd.DataFrame(grp_qid_pindustry)
grp_qid_pindustry.columns
grp_qid_pindustry.head()
```

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

Out[24]:

questions_id	professionals_industry
0003e7bf48f24b5c985f8fce96e611f3	[career counseling, computer software]
0006609dd4da40dcaa5a83e0499aba14	[, insurance law, professional training]
000af224bc2f4e94a19f8b62ba279cc4	[research]
000b30fb534b41f7b716fa9ebf9c3f35	[professional trainer, writing editing]
0018752e44b44e26bb74a0a43232b4d6	[financial services education]

In [25]:
```
grp_qid_pindustry["professionals_industry"]= grp_qid_pindustry["professionals_inc

grp_qid_pindustry["professionals_industry"]= grp_qid_pindustry["professionals_inc
grp_qid_pindustry["professionals_industry"]= grp_qid_pindustry["professionals_inc
grp_qid_pindustry["professionals_industry"]= grp_qid_pindustry["professionals_inc
grp_qid_pindustry["professionals_industry"]= grp_qid_pindustry["professionals_inc
# grp_qid_pindustry["professionals_industry"]= grp_qid_pindustry["professionals_i
grp_qid_pindustry["professionals_industry"]= grp_qid_pindustry["professionals_inc
```

In [26]:
```
grp_qid_pindustry.head(2)
```

Out[26]:

questions_id	professionals_industry
0003e7bf48f24b5c985f8fce96e611f3	career counseling computer software
0006609dd4da40dcaa5a83e0499aba14	insurance law professional training

In [27]:
```
# grp_qid_pindustry["professionals_industry"]= grp_qid_pindustry["professionals_i
grp_qid_pindustry["professionals_industry"]= grp_qid_pindustry["professionals_inc
# mystring = re.sub(",", "", mystring)
grp_qid_pindustry.head(2)
```

Out[27]:

questions_id	professionals_industry
0003e7bf48f24b5c985f8fce96e611f3	career_counseling__computer_software
0006609dd4da40dcaa5a83e0499aba14	__insurance_law__professional_training

In [28]:
```
grp_qid_pindustry["professionals_industry"] = grp_qid_pindustry["professionals_ir

# Print the updated dataframe
# print(df_updated)
grp_qid_pindustry.head()
```

Out[28]:

questions_id	professionals_industry
0003e7bf48f24b5c985f8fce96e611f3	career_counseling,computer_software
0006609dd4da40dcaa5a83e0499aba14	,insurance_law,professional_training
000af224bc2f4e94a19f8b62ba279cc4	research
000b30fb534b41f7b716fa9ebf9c3f35	professional_trainer,writing_editing
0018752e44b44e26bb74a0a43232b4d6	financial_services_education

In [29]:
```
grp_qid_pindustry.head()
```

Out[29]:

questions_id	professionals_industry
0003e7bf48f24b5c985f8fce96e611f3	career_counseling,computer_software
0006609dd4da40dcaa5a83e0499aba14	,insurance_law,professional_training
000af224bc2f4e94a19f8b62ba279cc4	research
000b30fb534b41f7b716fa9ebf9c3f35	professional_trainer,writing_editing
0018752e44b44e26bb74a0a43232b4d6	financial_services_education

In [30]:
```
grp_qid_tags = final.groupby('questions_id')['tags_tag_name'].apply(lambda x: lis
grp_qid_tags.head()
```

```
<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>

<IPython.core.display.Javascript object>

<IPython.core.display.Javascript object>

<IPython.core.display.Javascript object>

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

In [31]:
```
grp_qid_tags =pd.DataFrame(grp_qid_tags)
grp_qid_tags.columns
grp_qid_tags.head()
```

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

Out[31]:

questions_id	tags_tag_name
0003e7bf48f24b5c985f8fce96e611f3	[cybersecurity, disciplined, none, tech, women...
0006609dd4da40dcaa5a83e0499aba14	[business, career, career path, college, colle...
000af224bc2f4e94a19f8b62ba279cc4	[brazil, children, coaching, distance educatio...
000b30fb534b41f7b716fa9ebf9c3f35	[editing, instructional design, professional t...
0018752e44b44e26bb74a0a43232b4d6	[education, educator, financial services, gene...

In [32]:
```
grp_qid_tags["tags_tag_name"]= grp_qid_tags["tags_tag_name"].astype(str)

grp_qid_tags["tags_tag_name"]= grp_qid_tags["tags_tag_name"].str.replace("[", "")
grp_qid_tags["tags_tag_name"]= grp_qid_tags["tags_tag_name"].str.replace("]", "")
grp_qid_tags["tags_tag_name"]= grp_qid_tags["tags_tag_name"].str.replace("'", "")
grp_qid_tags["tags_tag_name"]= grp_qid_tags["tags_tag_name"].str.replace("'", "")
grp_qid_tags.head(2)
```

Out[32]:

questions_id	tags_tag_name
0003e7bf48f24b5c985f8fce96e611f3	cybersecurity, disciplined, none, tech, women ...
0006609dd4da40dcaa5a83e0499aba14	business, career, career path, college, colleg...

In [33]:
```
grp_qid_tags["tags_tag_name"] =grp_qid_tags["tags_tag_name"].replace(to_replace =

# Print the updated dataframe
# print(df_updated)
grp_qid_tags.head()
```

Out[33]:

questions_id	tags_tag_name
0003e7bf48f24b5c985f8fce96e611f3	cybersecurity,disciplined,none,tech,womentech
0006609dd4da40dcaa5a83e0499aba14	business,career,careerpath,college,collegeadmi...
000af224bc2f4e94a19f8b62ba279cc4	brazil,children,coaching,distanceeducation,ngo...
000b30fb534b41f7b716fa9ebf9c3f35	editing,instructionaldesign,professionaltraini...
0018752e44b44e26bb74a0a43232b4d6	education,educator,financialservices,generalin...

In [34]:
```
grp_qid_qbody = final.groupby('questions_id')['questions_body'].apply(lambda x: ]
grp_qid_qbody.head()
```

```
<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>

<IPython.core.display.Javascript object>

<IPython.core.display.Javascript object>

<IPython.core.display.Javascript object>

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

In [35]:
```
grp_qid_qbody =pd.DataFrame(grp_qid_qbody)
grp_qid_qbody.columns
grp_qid_qbody.head()
```

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

Out[35]:

questions_id	questions_body
0003e7bf48f24b5c985f8fce96e611f3	[junior h right thinking double major tech aca...
0006609dd4da40dcaa5a83e0499aba14	[currently undergrad want go law school lawyer...
000af224bc2f4e94a19f8b62ba279cc4	[degree next biology marine]
000b30fb534b41f7b716fa9ebf9c3f35	[hi currently debating whether pursue career e...
0018752e44b44e26bb74a0a43232b4d6	[hi high school senior set becoming pure mathe...

In [36]:
```
grp_qid_qbody["questions_body"]= grp_qid_qbody["questions_body"].astype(str)

grp_qid_qbody["questions_body"]= grp_qid_qbody["questions_body"].str.replace("[",
grp_qid_qbody["questions_body"]= grp_qid_qbody["questions_body"].str.replace("]",
grp_qid_qbody["questions_body"]= grp_qid_qbody["questions_body"].str.replace("'",
grp_qid_qbody["questions_body"]= grp_qid_qbody["questions_body"].str.replace("'",
grp_qid_qbody.head(2)
```

Out[36]:

questions_id	questions_body
0003e7bf48f24b5c985f8fce96e611f3	junior h right thinking double major tech acad...
0006609dd4da40dcaa5a83e0499aba14	currently undergrad want go law school lawyer ...

In [37]:
```
grp_qid_qtitle = final.groupby('questions_id')['questions_title'].apply(lambda x:
grp_qid_qtitle.head()
```

```
<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>

<IPython.core.display.Javascript object>

<IPython.core.display.Javascript object>

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

In [38]:
```
grp_qid_qtitle =pd.DataFrame(grp_qid_qtitle)
grp_qid_qtitle.columns
grp_qid_qtitle.head()
```

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

Out[38]:

questions_id	questions_title
0003e7bf48f24b5c985f8fce96e611f3	[double major tech academy high school worth]
0006609dd4da40dcaa5a83e0499aba14	[declare minor undergrad want lawyer]
000af224bc2f4e94a19f8b62ba279cc4	[get job prefered field]
000b30fb534b41f7b716fa9ebf9c3f35	[demand gym teachers diminishing]
0018752e44b44e26bb74a0a43232b4d6	[aspiring mathematician stay motivated researc...

In [39]:
```
grp_qid_qtitle["questions_title"]= grp_qid_qtitle["questions_title"].astype(str)

grp_qid_qtitle["questions_title"]= grp_qid_qtitle["questions_title"].str.replace(
grp_qid_qtitle["questions_title"]= grp_qid_qtitle["questions_title"].str.replace(
grp_qid_qtitle["questions_title"]= grp_qid_qtitle["questions_title"].str.replace(
grp_qid_qtitle["questions_title"]= grp_qid_qtitle["questions_title"].str.replace(
grp_qid_qtitle.head(2)
```

Out[39]:

questions_id	questions_title
0003e7bf48f24b5c985f8fce96e611f3	double major tech academy high school worth
0006609dd4da40dcaa5a83e0499aba14	declare minor undergrad want lawyer

In [40]:
```
df1 = pd.merge(grp_qid_qtitle, grp_qid_qbody, on="questions_id")
```

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

In [41]:
```
df1.head(2)
```

Out[41]:

questions_id	questions_title	questions_body
0003e7bf48f24b5c985f8fce96e611f3	double major tech academy high school worth	junior h right thinking double major tech acad...
0006609dd4da40dcaa5a83e0499aba14	declare minor undergrad want lawyer	currently undergrad want go law school lawyer ...

In [42]:
```
df2 = pd.merge(grp_qid_tags,grp_qid_pindustry, on="questions_id")
```

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

In [43]:
```
df3 = pd.merge(df1,df2,on="questions_id")
```

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

In [44]:
```
df3.head(2)
```

Out[44]:

questions_id	questions_title	questions_body	ta
0003e7bf48f24b5c985f8fce96e611f3	double major tech academy high school worth	junior h right thinking double major tech acad...	cybersecurity,disciplined,none,te
0006609dd4da40dcaa5a83e0499aba14	declare minor undergrad want lawyer	currently undergrad want go law school lawyer ...	business,career,careerpath,college

In [45]:
```
df3 = pd.DataFrame(df3)
# Data = df3.to_csv("Data.csv")
```

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

In [11]:
```
Data = pd.read_csv("Data.csv")
```

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

In [12]:
```
Data.head(2)
```

Out[12]:

	questions_id	questions_title	questions_body	
0	0003e7bf48f24b5c985f8fce96e611f3	double major tech academy high school worth	junior h right thinking double major tech acad...	cybersecurity,disciplined,non
1	0006609dd4da40dcaa5a83e0499aba14	declare minor undergrad want lawyer	currently undergrad want go law school lawyer ...	business,career,careerpath,coll

In [13]:
```
Data = Data.fillna("")
```

In [14]:
```
print("number of data points in sample :", Data.shape)
print("number of dimensions :", Data.shape[1])
```

```
number of data points in sample : (23931, 5)
number of dimensions : 5
```

In [15]:
```
preprocessed_data = Data[["questions_body","professionals_industry"]]
```

In [16]:
```
preprocessed_data["questions_body"].shape
```

Out[16]:
```
(23931,)
```

In [17]:
```
print("number of data points in sample :", preprocessed_data.shape[0])
print("number of dimensions :", preprocessed_data.shape[1])
```

```
number of data points in sample : 23931
number of dimensions : 2
```

In [18]:
```
from sklearn.feature_extraction.text import CountVectorizer
# count_vect = CountVectorizer()
vectorizer = CountVectorizer(tokenizer = lambda x: x.split(), binary='true')
multilabel_y = vectorizer.fit_transform(preprocessed_data["professionals_industry
```

In [20]:
```
multilabel_y
```

Out[20]:
```
<23931x1065 sparse matrix of type '<class 'numpy.int64'>'
        with 39904 stored elements in Compressed Sparse Row format>
```

In [22]:
```
multilabel_y.shape
```

Out[22]:
```
(23931, 1065)
```

In [23]:
```
def tags_to_choose(n):
    t = multilabel_y.sum(axis=0).tolist()[0]
    sorted_tags_i = sorted(range(len(t)), key=lambda i: t[i], reverse=True)
    multilabel_yn=multilabel_y[:,sorted_tags_i[:n]]
    return multilabel_yn

def questions_explained_fn(n):
    multilabel_yn = tags_to_choose(n)
    x= multilabel_yn.sum(axis=1)
    return (np.count_nonzero(x==0))
```

In [24]:
```
questions_explained = []
total_tags=multilabel_y.shape[1]
total_qs=preprocessed_data.shape[0]
for i in range(1, total_tags, 23931):
    questions_explained.append(np.round(((total_qs-questions_explained_fn(i))/tot
```

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

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

In [25]:
```
print(questions_explained)
```

```
[12.482]
```

In [26]:
```
multilabel_yx = tags_to_choose(1065)
print("number of questions that are not covered :", questions_explained_fn(23931)
```

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

number of questions that are not covered : 2427 out of  23931
```

In [27]:
```
print("Number of tags in sample :", multilabel_y.shape[1])
print("number of tags taken :", multilabel_yx.shape[1],"(",(multilabel_yx.shape[1
```

```
Number of tags in sample : 1065
number of tags taken : 1065 ( 100.0 %)
```

In [28]:
```
preprocessed_data.head(2)
```

Out[28]:

	questions_body	professionals_industry
0	junior h right thinking double major tech acad...	career_counseling computer_software
1	currently undergrad want go law school lawyer ...	insurance_law professional_training

In [29]:
```
total_size=preprocessed_data.shape[0]
train_size=int(0.80*total_size)

x_train=preprocessed_data.head(train_size)
x_test=preprocessed_data.tail(total_size - train_size)

y_train = multilabel_yx[0:train_size,:]
y_test = multilabel_yx[train_size:total_size,:]
```

In [30]:
```
print("Number of data points in train data :", y_train.shape)
print("Number of data points in test data :", y_test.shape)
```

```
Number of data points in train data : (19144, 1065)
Number of data points in test data : (4787, 1065)
```

In [31]:
```
print("Number of data points in train data :", type(y_train))
print("Number of data points in train data :", type(x_train))
```

```
Number of data points in train data : <class 'scipy.sparse.csr.csr_matrix'>
Number of data points in train data : <class 'pandas.core.frame.DataFrame'>
```

In [32]:
```
import datetime
# import datetime
start = datetime.date.today()
# start = datetime.now()
vectorizer = TfidfVectorizer(min_df=0.00009, max_features=200000, smooth_idf=True
                             tokenizer = lambda x: x.split(), sublinear_tf=False,
x_train_multilabel = vectorizer.fit_transform(x_train['questions_body'])
x_test_multilabel = vectorizer.transform(x_test['questions_body'])
print("Time taken to run this cell :", datetime.date.today() - start)
```

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

Time taken to run this cell : 0:00:00
```

In [33]:
```
print("Dimensions of train data X:",x_train_multilabel.shape, "Y :",y_train.shape
print("Dimensions of test data X:",x_test_multilabel.shape,"Y:",y_test.shape)
```

```
Dimensions of train data X: (19144, 67685) Y : (19144, 1065)
Dimensions of test data X: (4787, 67685) Y: (4787, 1065)
```

In [34]:
```
from sklearn.multiclass import OneVsRestClassifier
from sklearn.linear_model import SGDClassifier
import sklearn.metrics as metrics
classifier = OneVsRestClassifier(SGDClassifier(loss='log', alpha=0.00001, penalty
classifier.fit(x_train_multilabel, y_train)
predictions = classifier.predict(x_test_multilabel)

print("accuracy :",metrics.accuracy_score(y_test,predictions))
print("macro f1 score :",metrics.f1_score(y_test, predictions, average = 'macro')
print("micro f1 scoore :",metrics.f1_score(y_test, predictions, average = 'micro'
print("hamming loss :",metrics.hamming_loss(y_test,predictions))
print("Precision recall report :\n",metrics.classification_report(y_test, predict
```

```
accuracy : 0.12262377271777732
macro f1 score : 0.012517609578044025
micro f1 scoore : 0.14206981016533987
hamming loss : 0.0016488317832627686
Precision recall report :
              precision    recall  f1-score   support

           0       0.32      0.10      0.15       598
           1       0.40      0.12      0.18       548
           2       0.25      0.06      0.10       379
           3       0.55      0.32      0.40       293
           4       0.51      0.19      0.28       315
           5       0.38      0.06      0.10       288
           6       0.46      0.08      0.14       280
           7       0.45      0.09      0.15       272
           8       0.39      0.09      0.15       217
           9       0.64      0.20      0.30       200
          10       0.47      0.13      0.20       186
          11       0.68      0.10      0.18       163
```

In [35]:
```
from sklearn.metrics import precision_score,recall_score,f1_score
import datetime
from sklearn.linear_model import LogisticRegression

start = datetime.date.today()
classifier_2 = OneVsRestClassifier(LogisticRegression(C=1, penalty='l1', solver='
classifier_2.fit(x_train_multilabel, y_train)
predictions_2 = classifier_2.predict(x_test_multilabel)
print("Accuracy :",metrics.accuracy_score(y_test, predictions_2))
print("Hamming loss ",metrics.hamming_loss(y_test,predictions_2))


precision = precision_score(y_test, predictions_2, average='micro')
recall = recall_score(y_test, predictions_2, average='micro')
f1 = f1_score(y_test, predictions_2, average='micro')

print("Micro-average quality numbers")
print("Precision: {:.4f}, Recall: {:.4f}, F1-measure: {:.4f}".format(precision, r

precision = precision_score(y_test, predictions_2, average='macro')
recall = recall_score(y_test, predictions_2, average='macro')
f1 = f1_score(y_test, predictions_2, average='macro')

print("Macro-average quality numbers")
print("Precision: {:.4f}, Recall: {:.4f}, F1-measure: {:.4f}".format(precision, r

print (metrics.classification_report(y_test, predictions_2))
print("Time taken to run this cell :", datetime.date.today() - start)
```

```
Accuracy : 0.12429496553164822
Hamming loss  0.0015931253561337385
Micro-average quality numbers
Precision: 0.4786, Recall: 0.0580, F1-measure: 0.1035
Macro-average quality numbers
Precision: 0.0244, Recall: 0.0064, F1-measure: 0.0092
              precision    recall  f1-score   support

           0       0.34      0.03      0.06       598
           1       0.62      0.05      0.09       548
           2       0.35      0.03      0.05       379
           3       0.56      0.27      0.36       293
           4       0.63      0.17      0.27       315
           5       0.56      0.03      0.06       288
           6       0.57      0.07      0.13       280
           7       0.34      0.04      0.07       272
           8       0.72      0.06      0.11       217
           9       0.72      0.18      0.29       200
          10       0.40      0.05      0.09       186
```

In [37]:
```
import datetime
# import datetime
start = datetime.date.today()
# start = datetime.now()
vectorizer = TfidfVectorizer(min_df=0.00009, max_features=200000, smooth_idf=True
                             tokenizer = lambda x: x.split(), sublinear_tf=False,
X1 = vectorizer.fit_transform(Data['questions_body'])
X2 = vectorizer.fit_transform(Data['questions_title'])
print("Time taken to run this cell :", datetime.date.today() - start)
print("X1 :",X1.shape)
print("X2 :",X2.shape)
```

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

Time taken to run this cell : 0:00:00
X1 : (23931, 41409)
X2 : (23931, 11708)
```

In [38]:
```
import numpy as np
from scipy.sparse import hstack
X = hstack((X1,X2)).tocsr()
```

In [39]:
```
Y1 = multilabel_y
```

In [40]:
```
Y1.shape
```

Out[40]:
```
(23931, 1065)
```

In [ ]:
```
sum(list(np.array(Y1[0].todense())[0]))
```

In [41]:
```
print("Number of data points in train data :", type(Y1))
print("Number of data points in train data :", type(X))
```

```
Number of data points in train data : <class 'scipy.sparse.csr.csr_matrix'>
Number of data points in train data : <class 'scipy.sparse.csr.csr_matrix'>
```

In [42]:
```
X.shape
```

Out[42]:
```
(23931, 53117)
```

In [44]:
```
from numpy import mean
from numpy import std
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LogisticRegression

from sklearn.naive_bayes import MultinomialNB

cv = KFold(n_splits=3, random_state=1, shuffle=True)

model = classifier_2


scores = cross_val_score(model, X, Y1, scoring='f1_micro', cv=5, n_jobs=-1)

print('Scores :' , scores)
```

```
Scores : [0.10174484 0.09733618 0.10170266 0.09917165 0.08925955]
```

In [46]:
```
mean(scores)
```

Out[46]:
```
0.09784297832136385
```

In [ ]:
```

```