Google QUEST Q&A Labeling

1.Business/Real-world Problem

1.1. Description

Computers are really good at answering questions with single, verifiable answers. But, humans are often still better at answering questions about opinions, recommendations, or personal experiences.

Humans are better at addressing subjective questions that require a deeper, multidimensional understanding of context - something computers aren't trained to do well…yet.. Questions can take many forms - some have multi-sentence elaborations, others may be simple curiosity or a fully developed problem. They can have multiple intents, or seek advice and opinions. Some may be helpful and others interesting. Some are simple right or wrong.

1.2. Problem Statement

Build a model that predicts the probability of question and answer relevance.

1.3 Source/Useful Links

Source: https://www.kaggle.com/c/google-quest-challenge

2. Data

2.1. Data Overview

Data is in csv files.

- **train.csv** the training data (target labels are the last 30 columns).
- **test.csv** the test set (predict 30 labels for each test set row).

The data for this competition includes questions and answers from various StackExchange properties. Your task is to predict target values of 30 labels for each question-answer pair.

The list of 30 target labels are the same as the column names in the sample_submission.csv file. Target labels with the prefix question_ relate to the question_title and/or question_body features in the data. Target labels with the prefix answer_ relate to the answer feature.

Each row contains a single question and a single answer to that question, along with additional features. The training data contains rows with some duplicated questions (but with different answers). The test data does not contain any duplicated questions.

This is not a binary prediction challenge. Target labels are aggregated from multiple raters, and can have continuous values in the range [0,1]. Therefore, predictions must also be in that range.

3. Exploratory Data Analysis

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```
train = pd.read_csv("train.csv")
test = pd.read_csv("test.csv")
```

In [3]:

```
print('train', train.shape)
print('test', test.shape)
```

```
train (6079, 41)
test (476, 11)
```

In [4]:

```
train.head(3)
```

Out[4]:

	qa_id	question_title	question_body	question_user_name	question
0	0	What am I losing when using extension tubes in...	After playing around with macro photography on...	ysap	https://photo.stackexchange.con
1	1	What is the distinction between a city and a s...	I am trying to understand what kinds of places...	russellpierce	https://rpg.stackexchange.con
2	2	Maximum protusion length for through-hole comp...	I'm working on a PCB that has through-hole com...	Joe Baker	https://electronics.stackexchange.com/

3 rows × 41 columns

In [5]:

```
test.head(3)
```

Out[5]:

	qa_id	question_title	question_body	question_user_name	que
0	39	Will leaving corpses lying around upset my pri...	I see questions/information online about how t...	Dylan	https://gaming.stackexchange.
1	46	Url link to feature image in the portfolio	I am new to Wordpress. i have issue with Featu...	Anu	https://wordpress.stackexchange.
2	70	Is accuracy, recoil or bullet spread affected ...	To experiment I started a bot game, toggled in...	Konsta	https://gaming.stackexchange.

In [6]:

```
train.columns
```

Out[6]:

```
Index(['qa_id', 'question_title', 'question_body', 'question_user_name',
       'question_user_page', 'answer', 'answer_user_name', 'answer_user_pag
e',
       'url', 'category', 'host', 'question_asker_intent_understanding',
       'question_body_critical', 'question_conversational',
       'question_expect_short_answer', 'question_fact_seeking',
       'question_has_commonly_accepted_answer',
       'question_interestingness_others', 'question_interestingness_self',
       'question_multi_intent', 'question_not_really_a_question',
       'question_opinion_seeking', 'question_type_choice',
       'question_type_compare', 'question_type_consequence',
       'question_type_definition', 'question_type_entity',
       'question_type_instructions', 'question_type_procedure',
       'question_type_reason_explanation', 'question_type_spelling',
       'question_well_written', 'answer_helpful',
       'answer_level_of_information', 'answer_plausible', 'answer_relevanc
e',
       'answer_satisfaction', 'answer_type_instructions',
       'answer_type_procedure', 'answer_type_reason_explanation',
       'answer_well_written'],
      dtype='object')
```

In [7]:

```
test.columns
```

Out[7]:

```
Index(['qa_id', 'question_title', 'question_body', 'question_user_name',
       'question_user_page', 'answer', 'answer_user_name', 'answer_user_pag
e',
       'url', 'category', 'host'],
      dtype='object')
```

Above 11 columns are used for training the model.

In [8]:

```
#seperating the target features from the train columns
feature_columns = [col for col in test.columns if col not in 'qa_id']
print('Features : ', feature_columns)
```

```
Features :  ['question_title', 'question_body', 'question_user_name', 'quest
ion_user_page', 'answer', 'answer_user_name', 'answer_user_page', 'url', 'ca
tegory', 'host']
```

In [9]:

```
train[feature_columns].head(3)
```

Out[9]:

	question_title	question_body	question_user_name	question_user_p
0	What am I losing when using extension tubes in...	After playing around with macro photography on...	ysap	https://photo.stackexchange.com/users/1
1	What is the distinction between a city and a s...	I am trying to understand what kinds of places...	russellpierce	https://rpg.stackexchange.com/users/8
2	Maximum protusion length for through-hole comp...	I'm working on a PCB that has through-hole com...	Joe Baker	https://electronics.stackexchange.com/users/1(

from train data it is noted that question_user_page, answer_user_page, url, host columns or features contains url's of data. These are not useful for processing data.

In [10]:

```
feature_columns = [col for col in feature_columns if col not in ['question_user_page','answ
print(feature_columns)
```

```
['question_title', 'question_body', 'question_user_name', 'answer', 'answer_
user_name', 'category']
```

Among feature_columns the features 'question_user_name' and 'answer_user_name' are also not useful they are just names of the people who posted question and who answered thw question. So, these are not useful in predicting the question and answer relevences because the future data may not be posted and answered by the same persons.

In [11]:

```
feature_columns = [col for col in feature_columns if col not in [ 'question_user_name', 'an
print(feature_columns)
```

```
['question_title', 'question_body', 'answer', 'category']
```

In [12]:

```
#checking whether the features has any null values are not
train.isna().sum()
```

Out[12]:

```
qa_id                                    0
question_title                           0
question_body                            0
question_user_name                       0
question_user_page                       0
answer                                   0
answer_user_name                         0
answer_user_page                         0
url                                      0
category                                 0
host                                     0
question_asker_intent_understanding      0
question_body_critical                   0
question_conversational                  0
question_expect_short_answer             0
question_fact_seeking                    0
question_has_commonly_accepted_answer    0
question_interestingness_others          0
question_interestingness_self            0
question_multi_intent                    0
question_not_really_a_question           0
question_opinion_seeking                 0
question_type_choice                     0
question_type_compare                    0
question_type_consequence                0
question_type_definition                 0
question_type_entity                     0
question_type_instructions               0
question_type_procedure                  0
question_type_reason_explanation         0
question_type_spelling                   0
question_well_written                    0
answer_helpful                           0
answer_level_of_information              0
answer_plausible                         0
answer_relevance                         0
answer_satisfaction                      0
answer_type_instructions                 0
answer_type_procedure                    0
answer_type_reason_explanation           0
answer_well_written                      0
dtype: int64
```

In [13]:

```
#checking test features have null values or not
test.isna().sum()
```

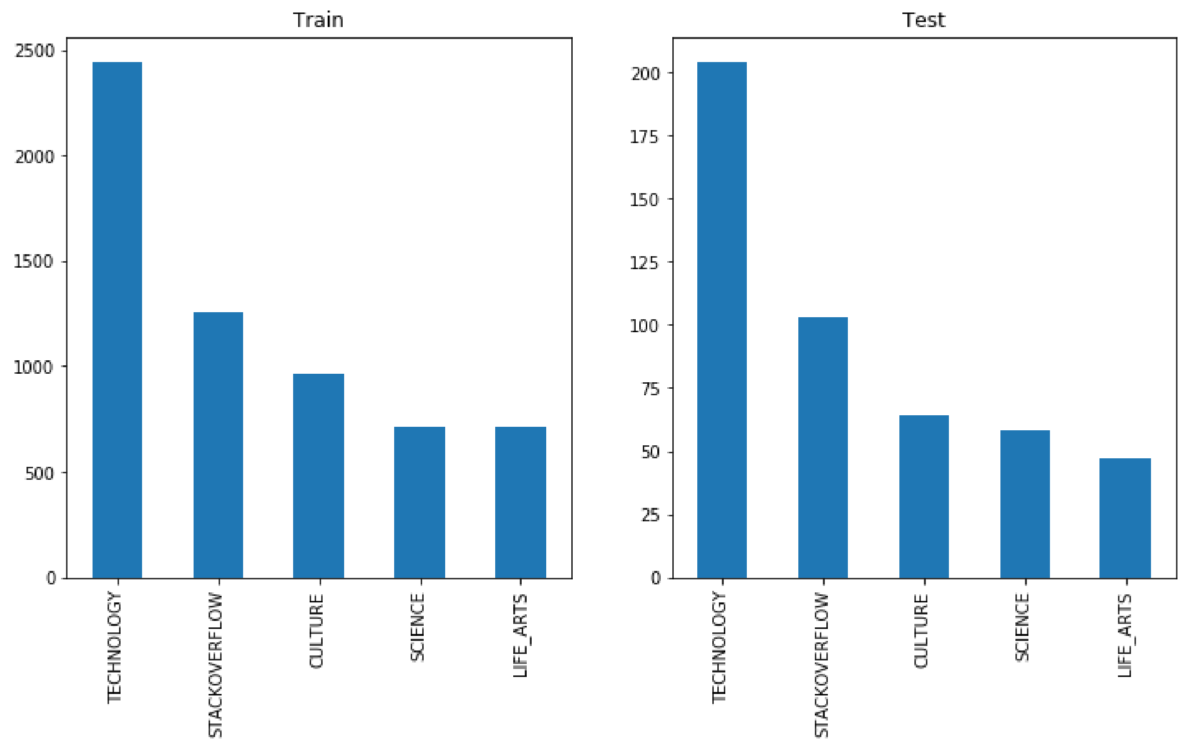
Out[13]:

```
qa_id                 0
question_title        0
question_body         0
question_user_name    0
question_user_page    0
answer                0
answer_user_name      0
answer_user_page      0
url                   0
category              0
host                  0
dtype: int64
```

In [14]:

```
#bar plot for train and test dataset based on category
train_cat = train['category'].value_counts()
test_cate = test['category'].value_counts()

fig, axes = plt.subplots(1, 2, figsize=(12, 6))
train_cat.plot(kind='bar', ax=axes[0])
axes[0].set_title('Train')
test_cate.plot(kind='bar', ax=axes[1])
axes[1].set_title('Test')
print('category distribution')
```

```
category distribution
```



In [15]:

```
#finding duplicate data present in question title of train data
train['question_title'].value_counts().head(2)
```

Out[15]:

```
What is the best introductory Bayesian statistics textbook?    12
What does mathematics have to do with programming?             11
Name: question_title, dtype: int64
```

In [16]:

```
#storing the indexes of question "What is the best introductory Bayesian statistics textboo
ind = train.index[train['question_title']=='What is the best introductory Bayesian statisti
```

In [17]:

```
#displaying indexes
ind
```

Out[17]:

```
[229, 1616, 1647, 2104, 3476, 3762, 3801, 3899, 4408, 5239, 5587, 5766]
```

In [18]:

```
#displaying category for all those indexes
for i in ind:
    print(train.loc[i]['category'])
```

```
SCIENCE
SCIENCE
SCIENCE
SCIENCE
SCIENCE
SCIENCE
SCIENCE
SCIENCE
SCIENCE
SCIENCE
SCIENCE
SCIENCE
```

The above data shows that all the catgories of different index question are same

In [19]:

```
ind = train.index[train['question_title']=='Important non-technical course for programmers?
```

In [20]:

```
ind
```

Out[20]:

```
[234, 747, 1217, 1452, 2807, 2809, 3240, 3862, 3964, 4602, 5975]
```

In [21]:

```
for i in ind:
    print(train.loc[i]['category'])
```

```
TECHNOLOGY
TECHNOLOGY
TECHNOLOGY
TECHNOLOGY
TECHNOLOGY
TECHNOLOGY
TECHNOLOGY
TECHNOLOGY
TECHNOLOGY
TECHNOLOGY
TECHNOLOGY
```

1.This data printed inorder to check the same question title may have same category or not. With this it can understood that the category has no role to play for improving any probabilities

2.The above data proves that 'category' feature also has no imporatnce for classifying the relevance of question and answer

In [22]:

```
feature_columns = [col for col in feature_columns if col not in [ 'category']]
print(feature_columns)
```

```
['question_title', 'question_body', 'answer']
```

checking Test data duplicates

In [23]:

```
test['question_title'].value_counts()
```

Out[23]:

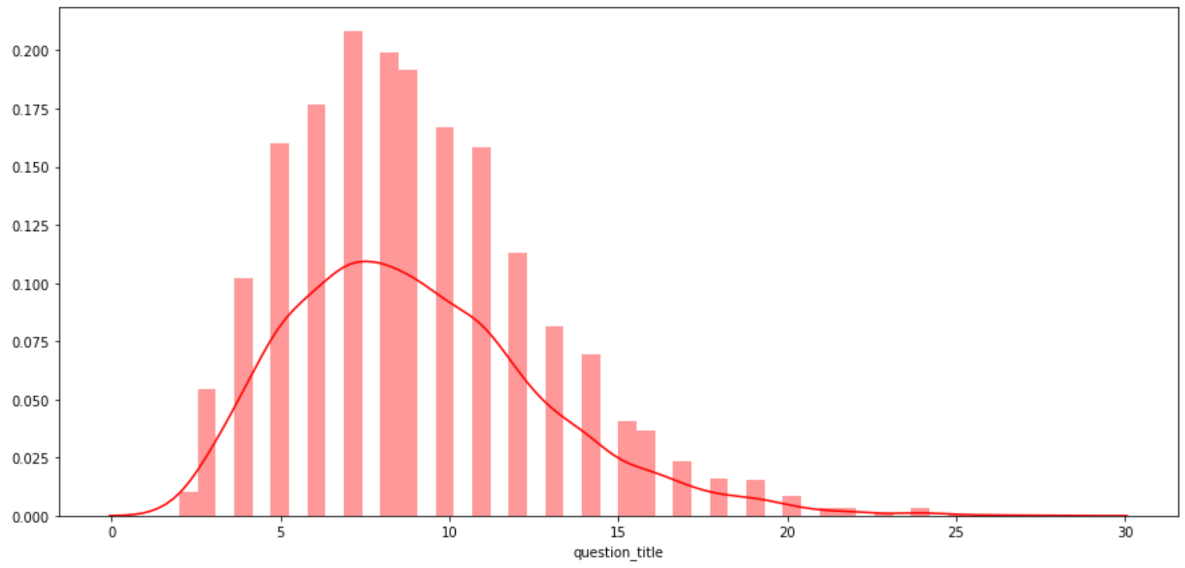
```
COPY command: copy only specific columns from csv
1
Reduce image size without resolution change
1
variation in antigen binding site of antibodies
1
How do I import an iPhoto library into Lightroom, keeping adjustments intac
t?               1
Multivariate exponential smoothing and Kalman filter equivalence
1

..
Can 'Dupe' be used as a verb instead of 'Duplicate'?
1
SharePoint 2010: Limit search to return a Document Set after matching on its
child item    1
How to find the actual working directory?
1
Monitor activity to an IP Address - Windows 2003 and 2008 Server
1
Oracle Regular Expression To replace A with Z, B with Y, C with X and So on
1
Name: question_title, Length: 476, dtype: int64
```

Test data has no duplicate question titles.

Train features word plots

In [37]:

```
words = train['question_title'].apply(lambda x : len(x.split(' ')))
plt.figure(figsize=(15,7))
sns.distplot(words, color='r')
plt.show()
```



In [48]:

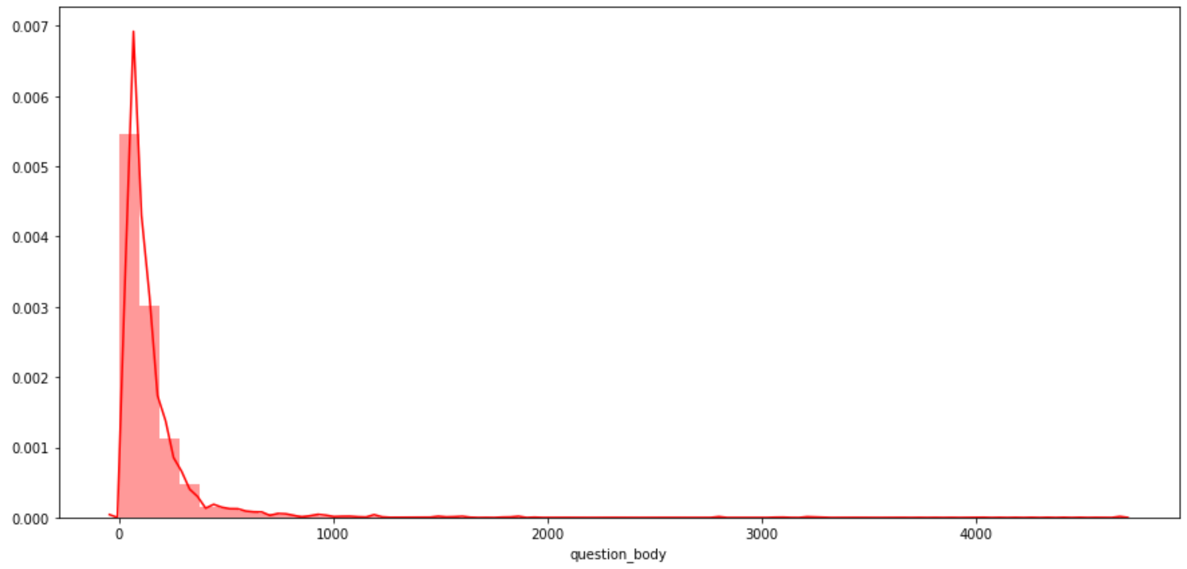
```
words = train['question_title'].apply(lambda x : len(x.split(' ')))
words = sorted(words.values)
for i in range(91,100):

    print("{} percentage question_title text contains words less than : {}".format(i,words[
```

```
91 percentage question_title text contains words less than : 14
92 percentage question_title text contains words less than : 15
93 percentage question_title text contains words less than : 15
94 percentage question_title text contains words less than : 16
95 percentage question_title text contains words less than : 16
96 percentage question_title text contains words less than : 17
97 percentage question_title text contains words less than : 17
98 percentage question_title text contains words less than : 19
99 percentage question_title text contains words less than : 20
```

1. 99 percantage of data contains 20 words as an average for each question.

In [36]:

```
words = train['question_body'].apply(lambda x : len(x.split(' ')))
plt.figure(figsize=(15,7))
sns.distplot(words, color='r')
plt.show()
```



In [42]:

```
words = train['question_body'].apply(lambda x : len(x.split(' ')))
words = sorted(words.values)
for i in range(91,101):

    print("{} percentage question_body text contains words less than : {}".format(i,words[r
```

```
91 percentage question_body text contains words less than : 300
92 percentage question_body text contains words less than : 318
93 percentage question_body text contains words less than : 342
94 percentage question_body text contains words less than : 372
95 percentage question_body text contains words less than : 433
96 percentage question_body text contains words less than : 499
97 percentage question_body text contains words less than : 580
98 percentage question_body text contains words less than : 722
99 percentage question_body text contains words less than : 1047
```

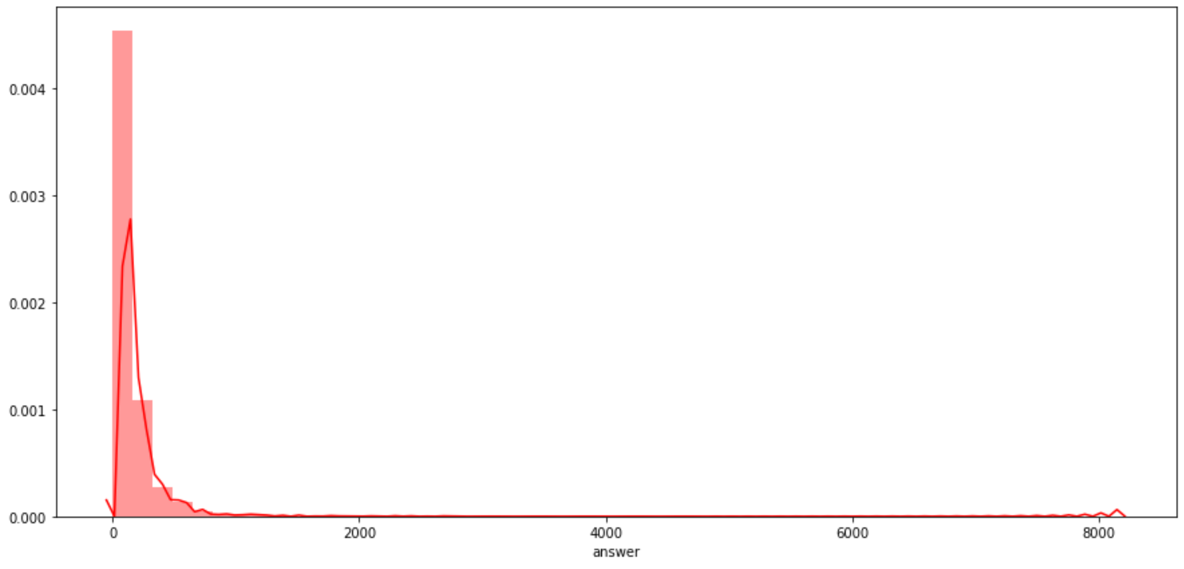
1. Highest number of words present for question is around 4000

2. 99 percentage of questions contains words 1047 but it is better to consider 97 pecentage of questions contains 580 words

3. If 99 percentage is considered then if we perform padding while processing data then we will be getting more zeroes in data. If we consider 97 percentage then we remove around 500 features wih zeroes.

In [35]:

```
words = train['answer'].apply(lambda x : len(x.split(' ')))
plt.figure(figsize=(15,7))
sns.distplot(words, color='r')
plt.show()
```



In [43]:

```
words = train['answer'].apply(lambda x : len(x.split(' ')))
words = sorted(words.values)
for i in range(91,101):

    print("{} percentage answer text contains words less than : {}".format(i,words[round((l
```

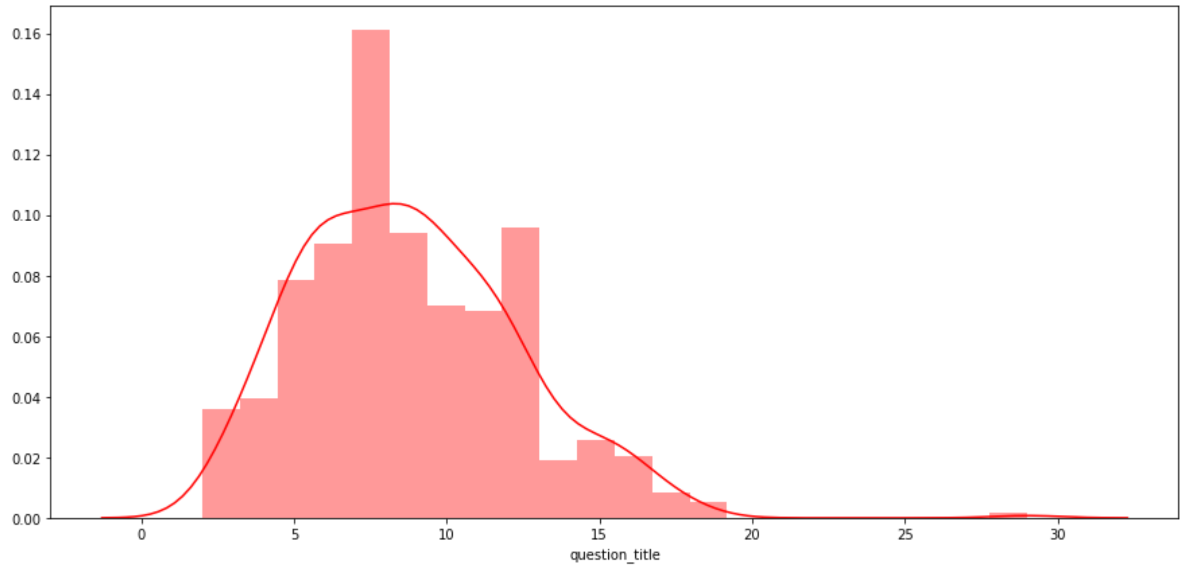
```
91 percentage answer text contains words less than : 314
92 percentage answer text contains words less than : 341
93 percentage answer text contains words less than : 364
94 percentage answer text contains words less than : 393
95 percentage answer text contains words less than : 430
96 percentage answer text contains words less than : 492
97 percentage answer text contains words less than : 552
98 percentage answer text contains words less than : 622
99 percentage answer text contains words less than : 884
```

From observing above data it is better to consider 96% questions containing words

Test features word plots

In [38]:

```
words = test['question_title'].apply(lambda x : len(x.split(' ')))
plt.figure(figsize=(15,7))
sns.distplot(words, color='r')
plt.show()
```



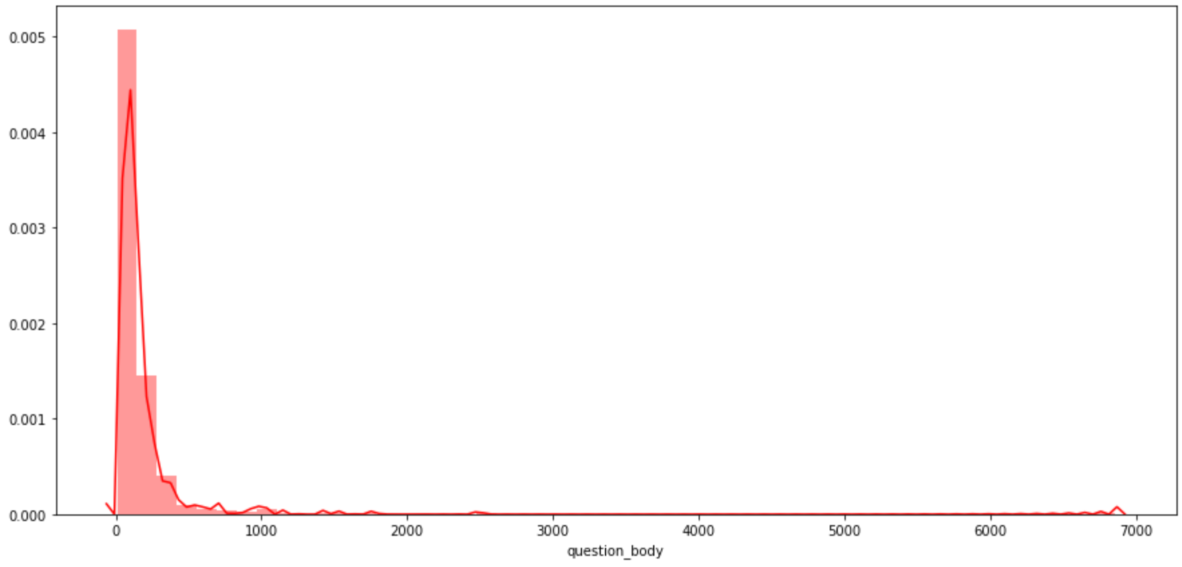
In [44]:

```
words = test['question_title'].apply(lambda x : len(x.split(' ')))
words = sorted(words.values)
for i in range(91,100):

    print("{} percentage question_title text contains words less than : {}".format(i,words[
```

```
91 percentage question_title text contains words less than : 14
92 percentage question_title text contains words less than : 14
93 percentage question_title text contains words less than : 15
94 percentage question_title text contains words less than : 15
95 percentage question_title text contains words less than : 15
96 percentage question_title text contains words less than : 16
97 percentage question_title text contains words less than : 16
98 percentage question_title text contains words less than : 16
99 percentage question_title text contains words less than : 17
```

In [39]:

```
words = test['question_body'].apply(lambda x : len(x.split(' ')))
plt.figure(figsize=(15,7))
sns.distplot(words, color='r')
plt.show()
```



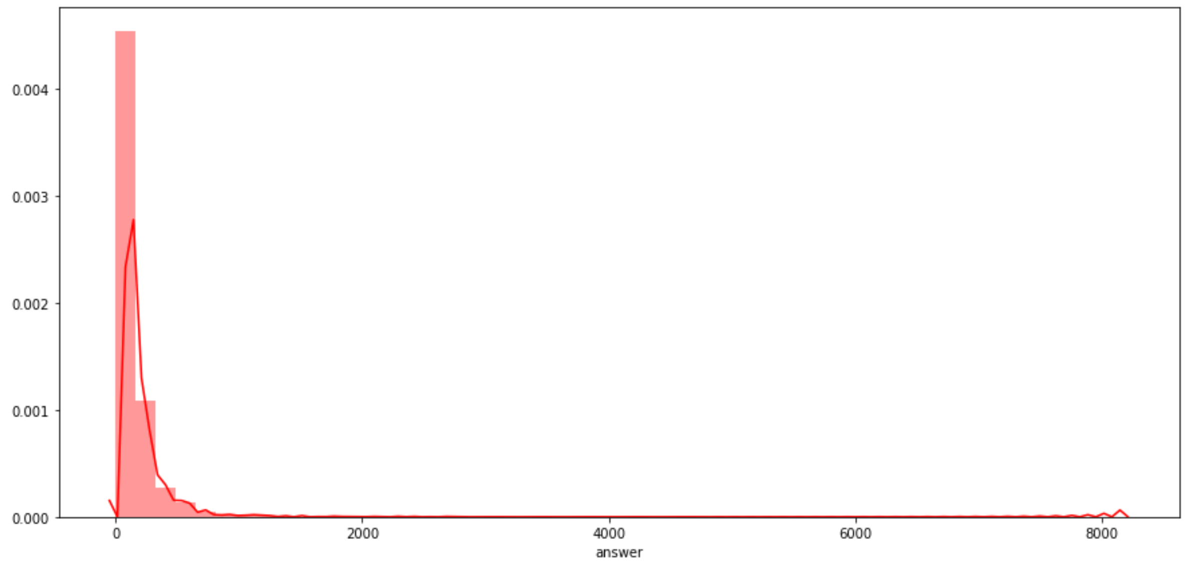
In [45]:

```
words = test['question_body'].apply(lambda x : len(x.split(' ')))
words = sorted(words.values)
for i in range(91,100):

    print("{} percentage question_body text contains words less than : {}".format(i,words[r
```

```
91 percentage question_body text contains words less than : 307
92 percentage question_body text contains words less than : 338
93 percentage question_body text contains words less than : 376
94 percentage question_body text contains words less than : 395
95 percentage question_body text contains words less than : 442
96 percentage question_body text contains words less than : 582
97 percentage question_body text contains words less than : 717
98 percentage question_body text contains words less than : 976
99 percentage question_body text contains words less than : 1422
```

In [40]:

```
words = train['answer'].apply(lambda x : len(x.split(' ')))
plt.figure(figsize=(15,7))
sns.distplot(words, color='r')
plt.show()
```



In [46]:

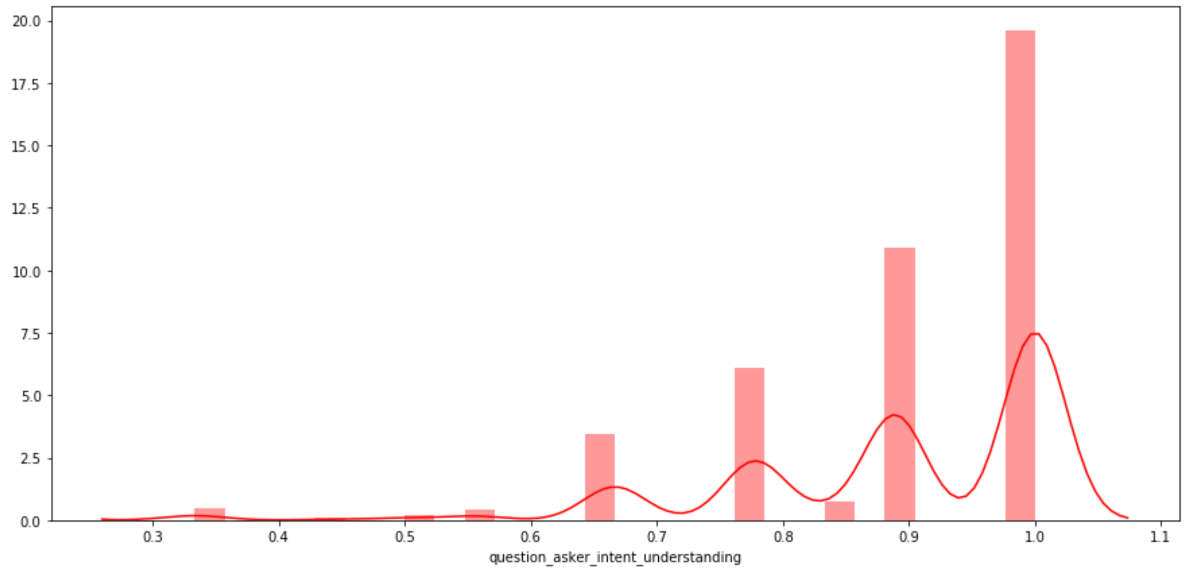
```
words = test['answer'].apply(lambda x : len(x.split(' ')))
words = sorted(words.values)
for i in range(91,100):

    print("{} percentage answer text contains words less than : {}".format(i,words[round((l
```

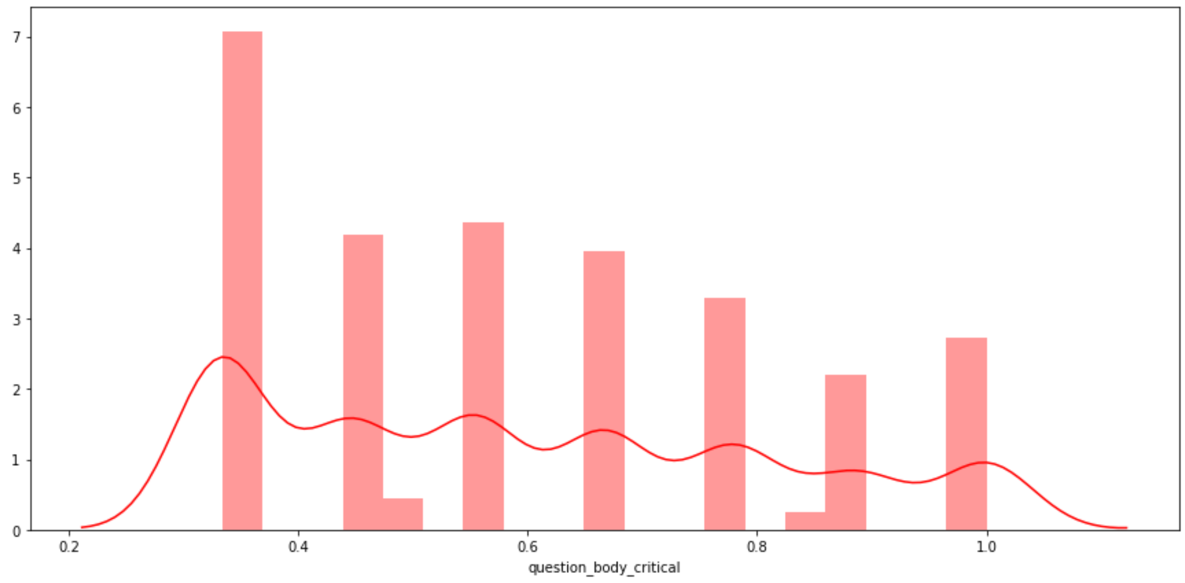
```
91 percentage answer text contains words less than : 383
92 percentage answer text contains words less than : 399
93 percentage answer text contains words less than : 415
94 percentage answer text contains words less than : 437
95 percentage answer text contains words less than : 478
96 percentage answer text contains words less than : 490
97 percentage answer text contains words less than : 601
98 percentage answer text contains words less than : 977
99 percentage answer text contains words less than : 1235
```

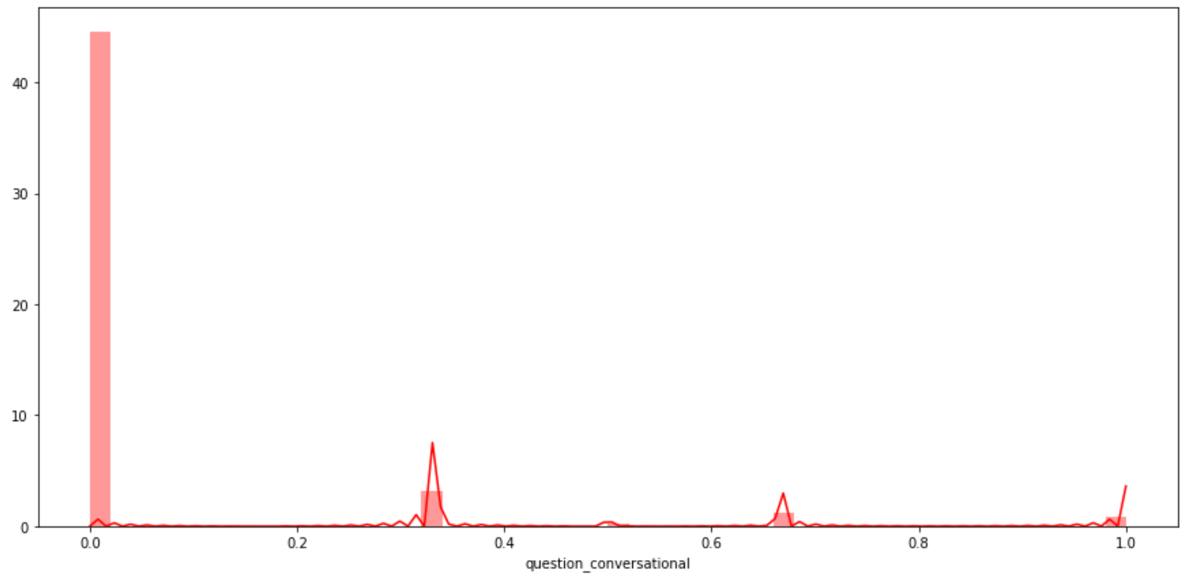
Target features

In [34]:

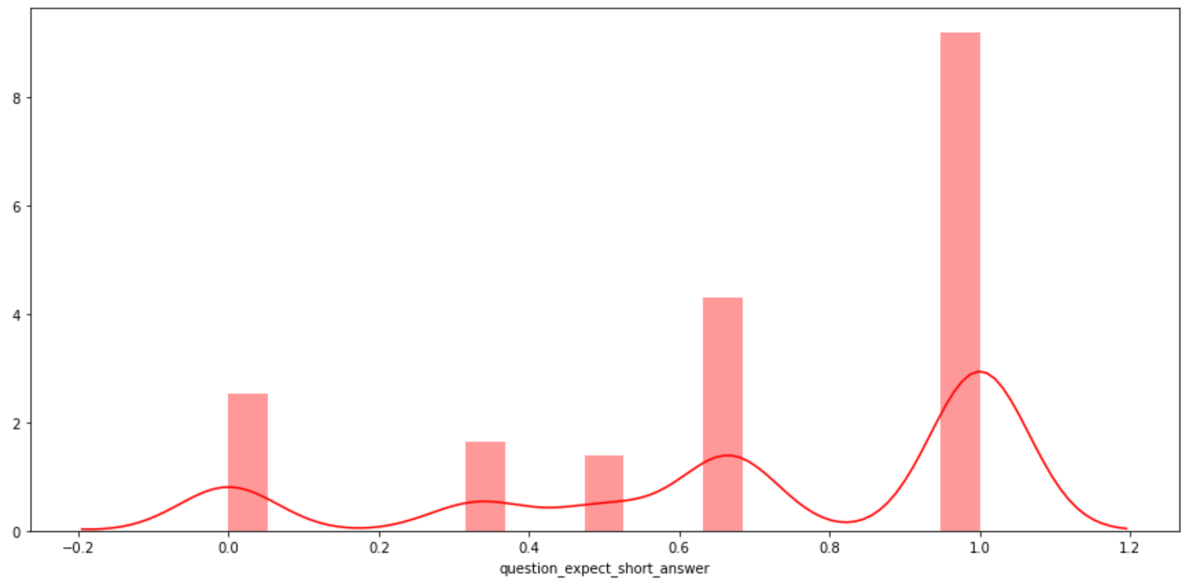
```
for i in train.columns:
    if i not in test.columns:
        plt.figure(figsize=(15,7))
        sns.distplot(train[i], color='r')
        plt.show()
```



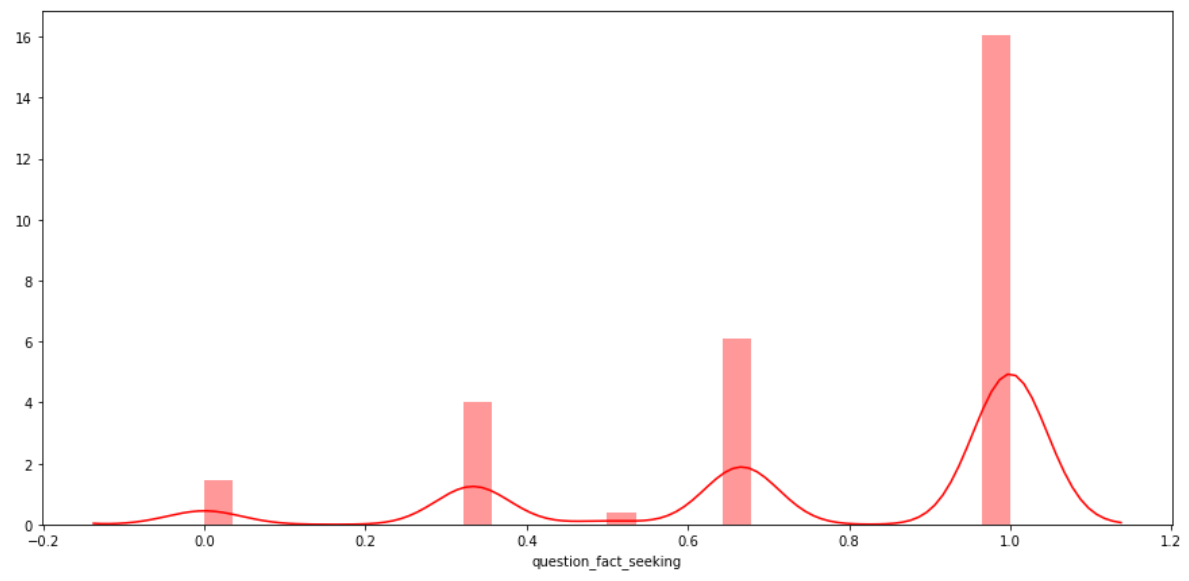




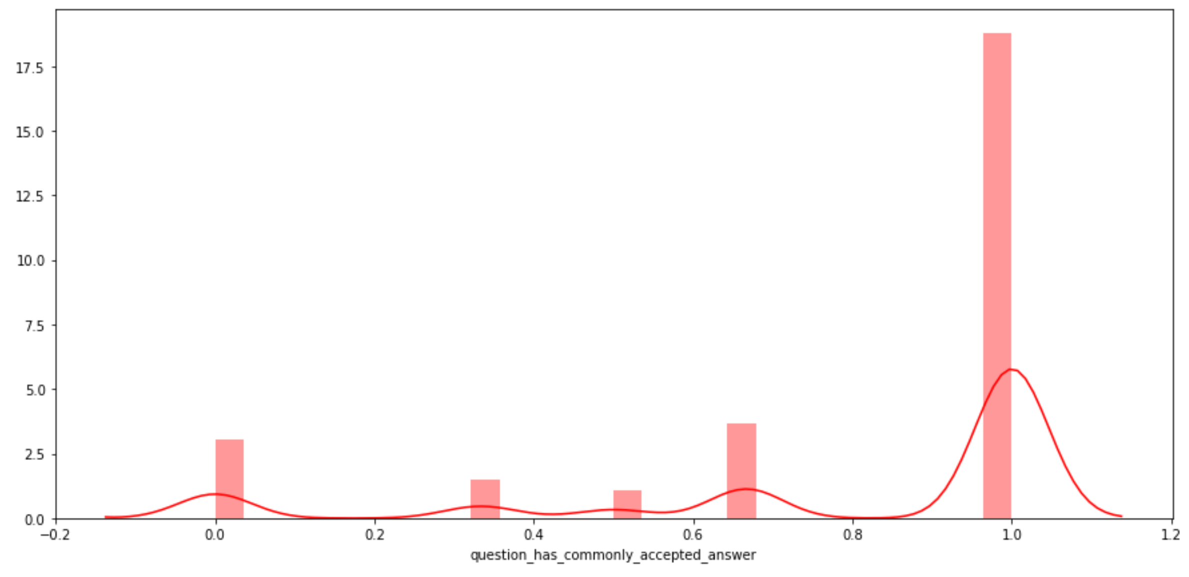
0
10
20
30
40
0.0
0.2
0.4
0.6
0.8
1.0
question_conversational



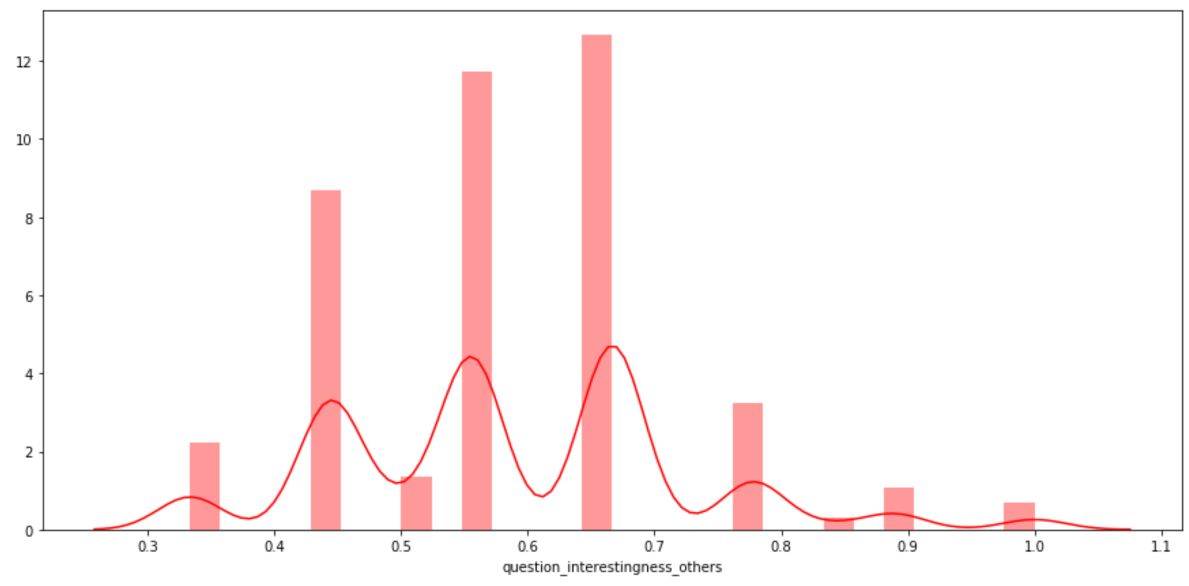
0
2
4
6
8
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2
question_expect_short_answer



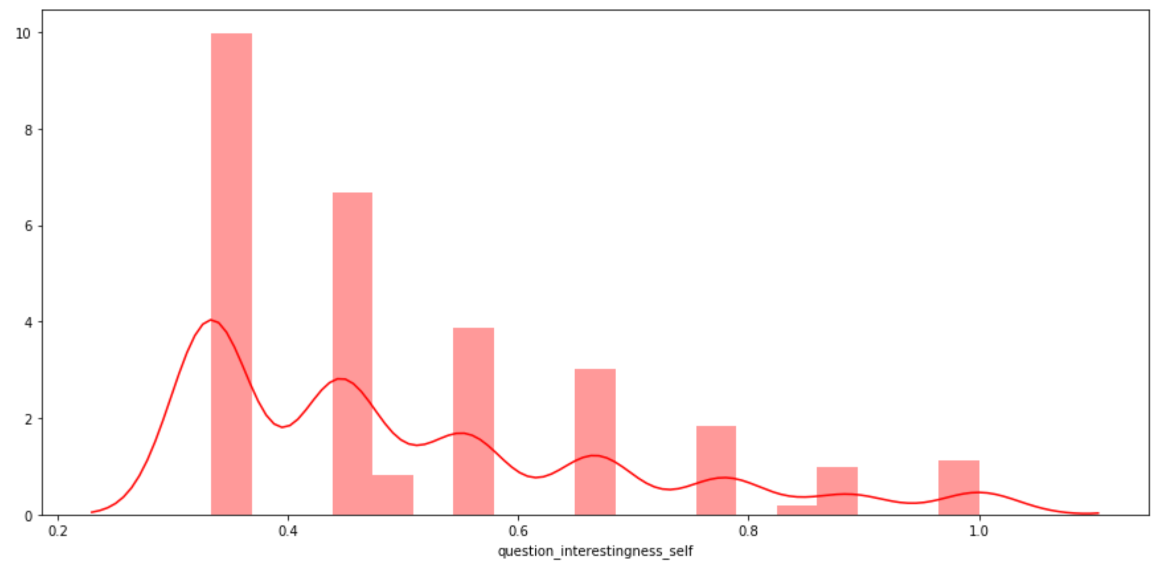
16
14
12
10
8
6
4
2
0
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2
question_fact_seeking



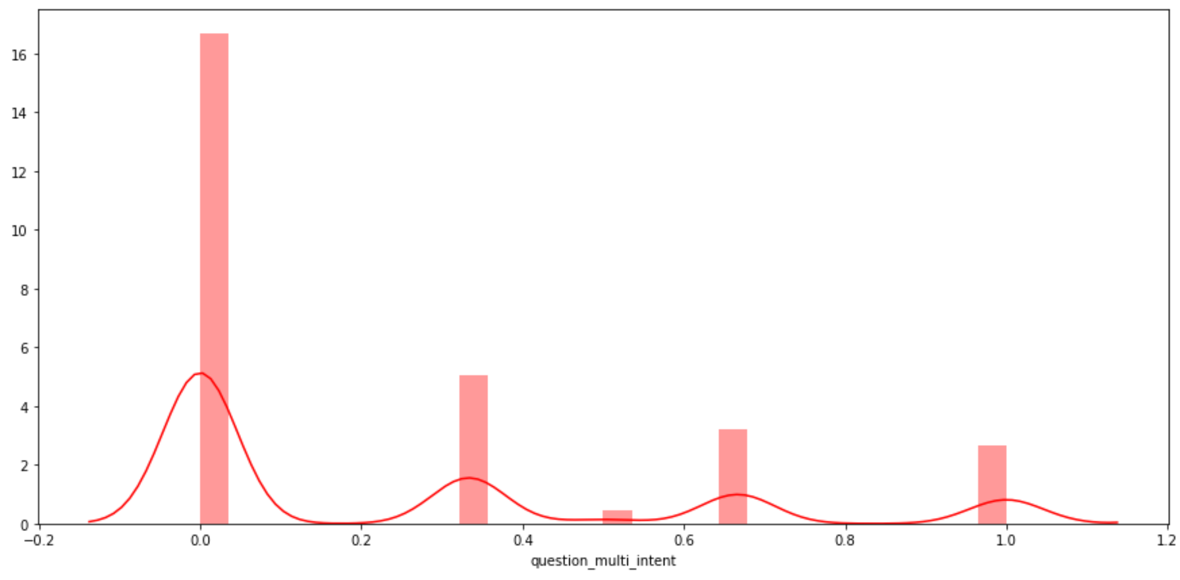
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2
question_has_commonly_accepted_answer

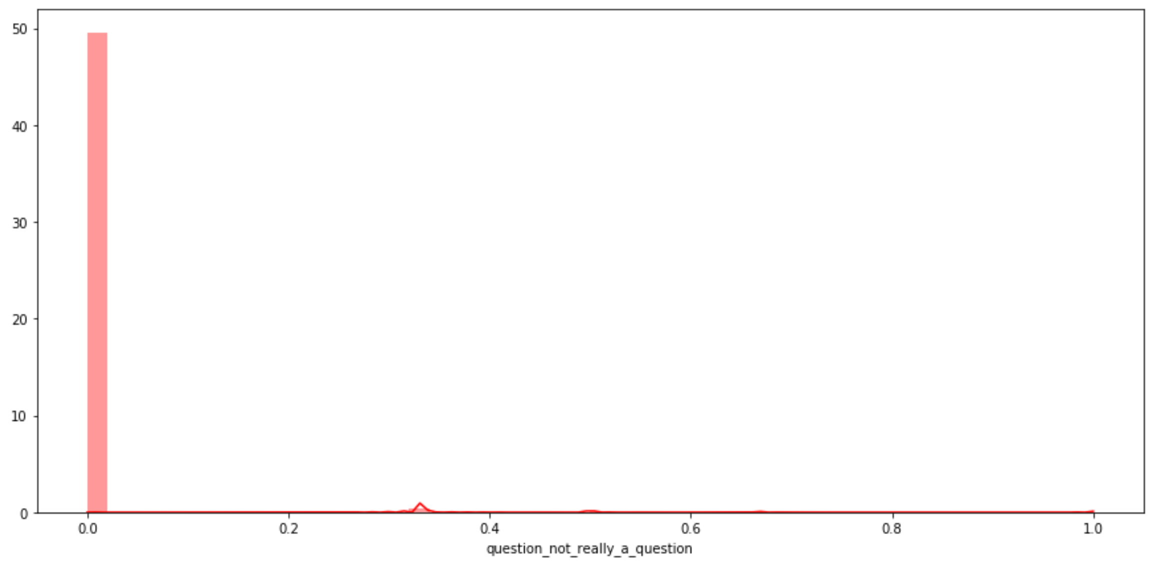


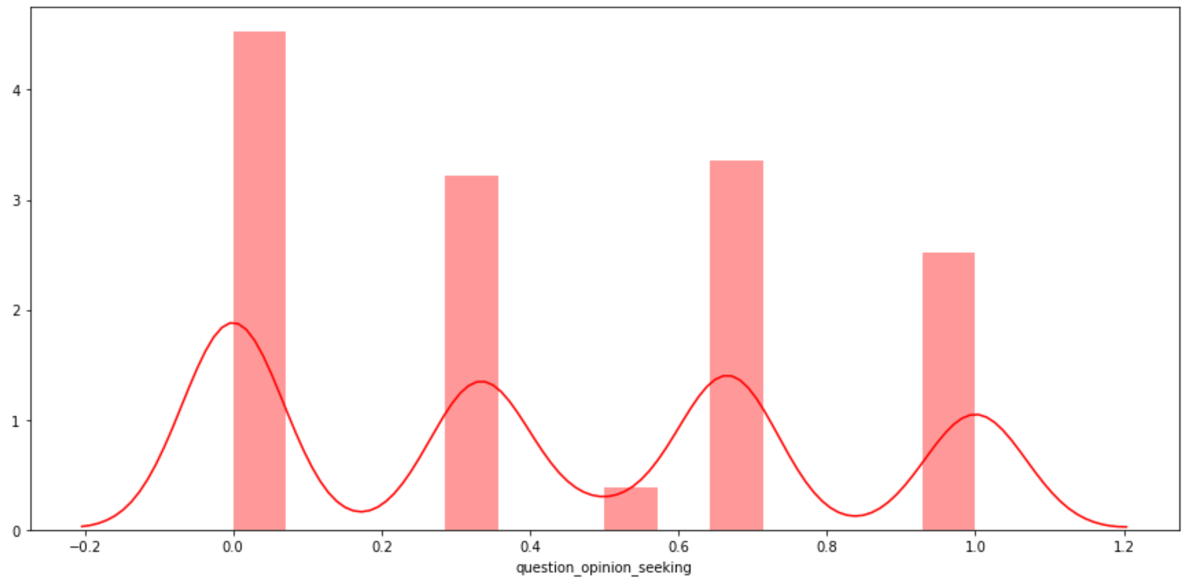
0
2
4
6
8
10
12
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
1.1
question_interestingness_others



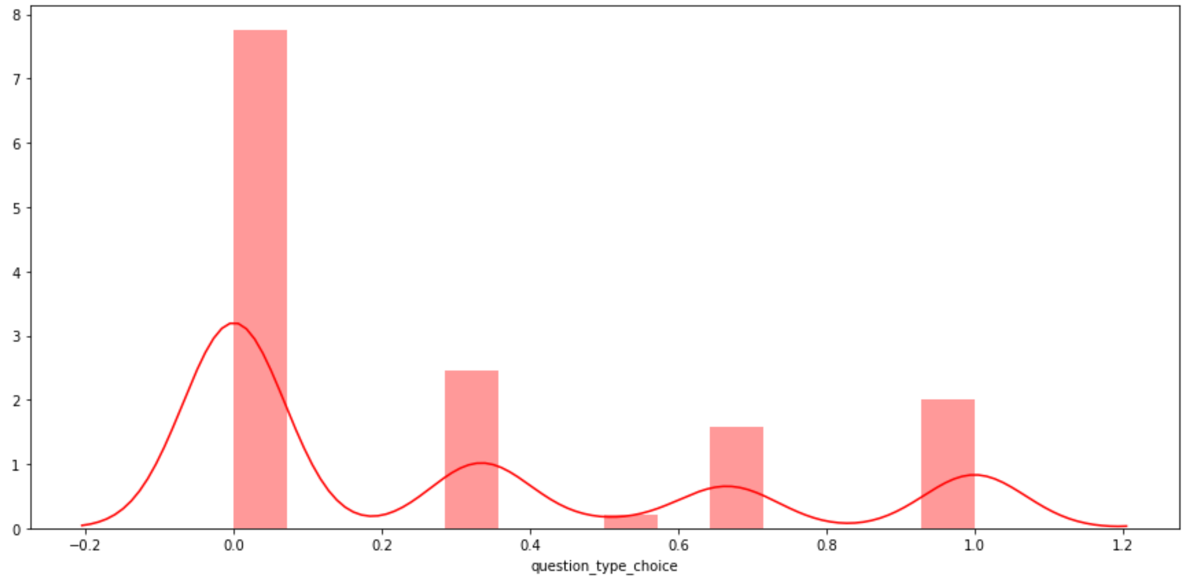
0
2
4
6
8
10
0.2
0.4
0.6
0.8
1.0
question_interestingness_self


question_multi_intent

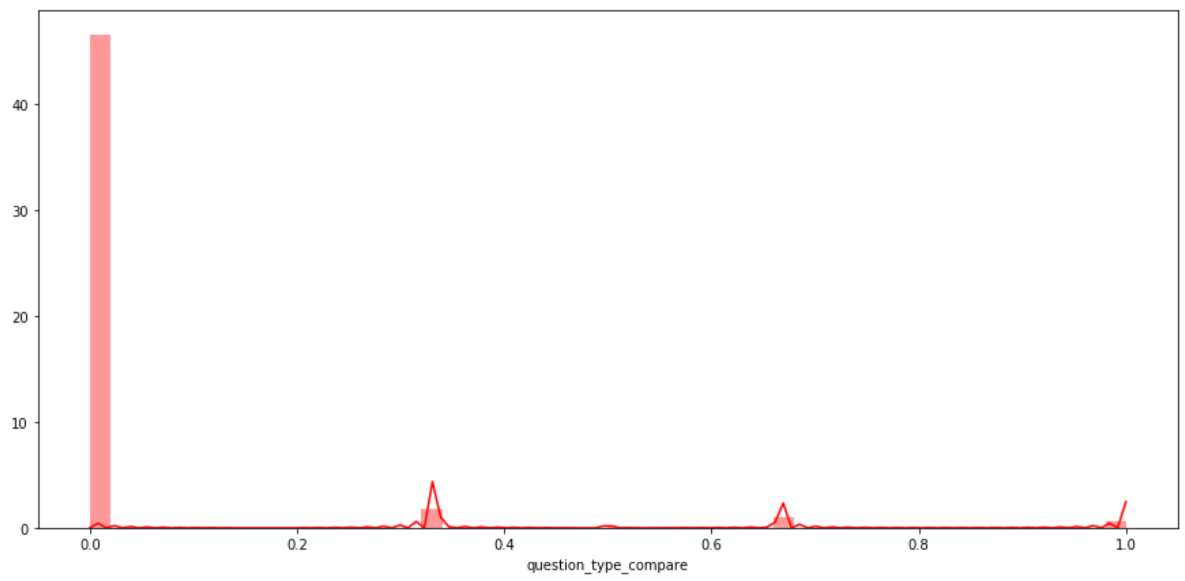

question_not_really_a_question

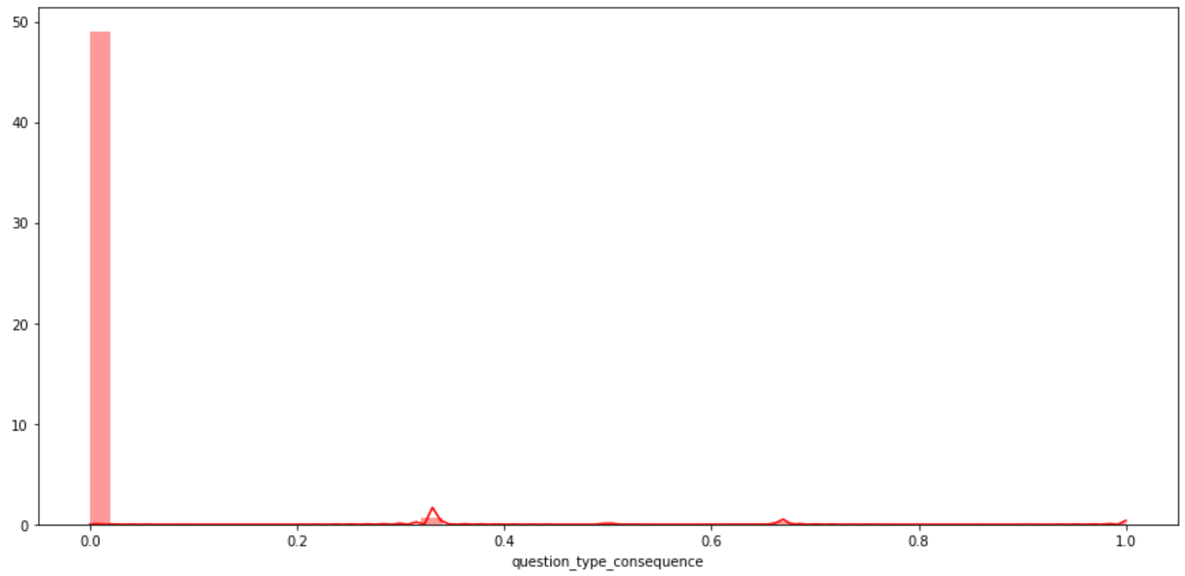


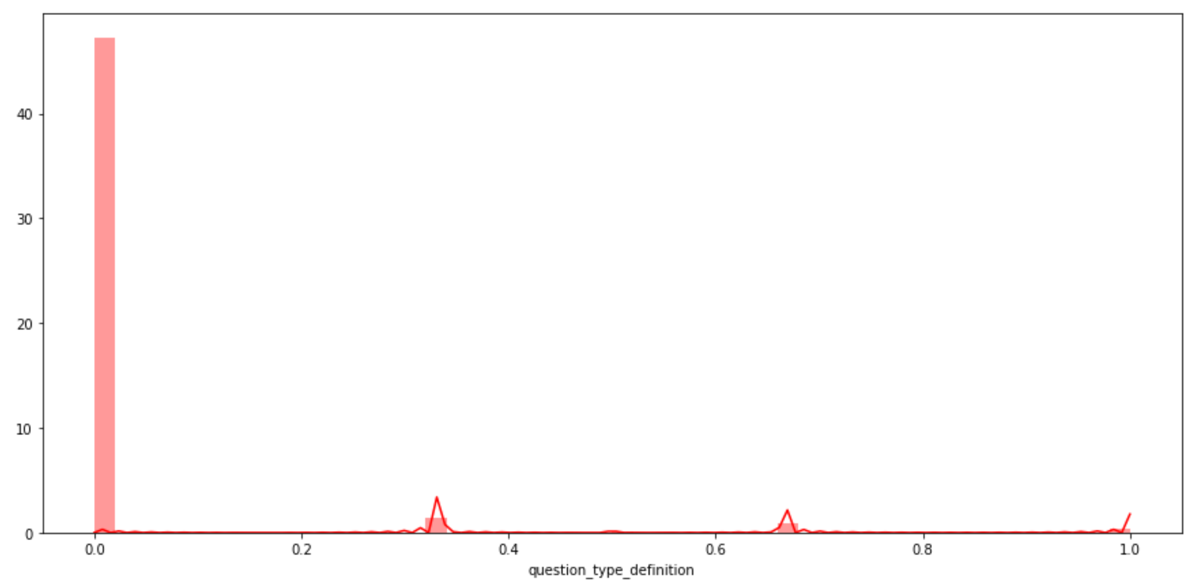
4
3
2
1
0
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2
question_opinion_seeking

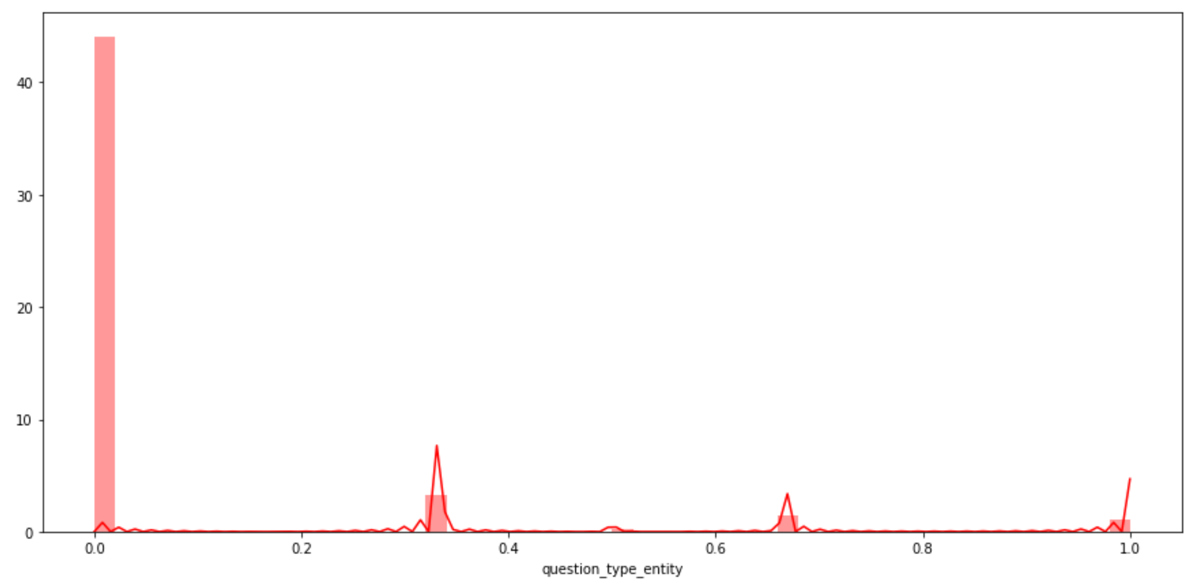


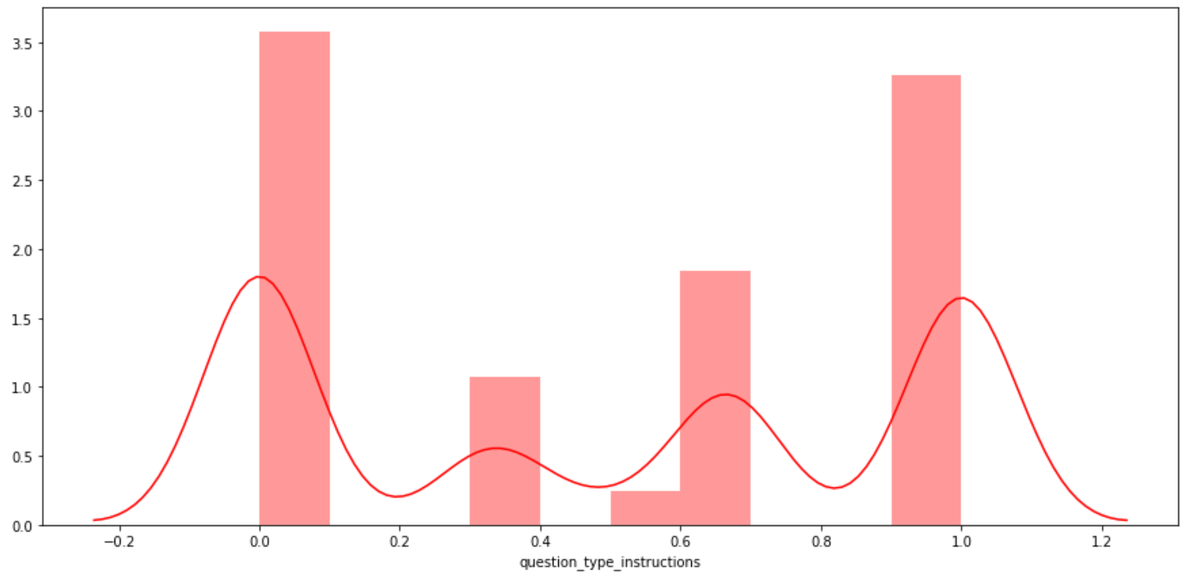
8
7
6
5
4
3
2
1
0
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2
question_type_choice

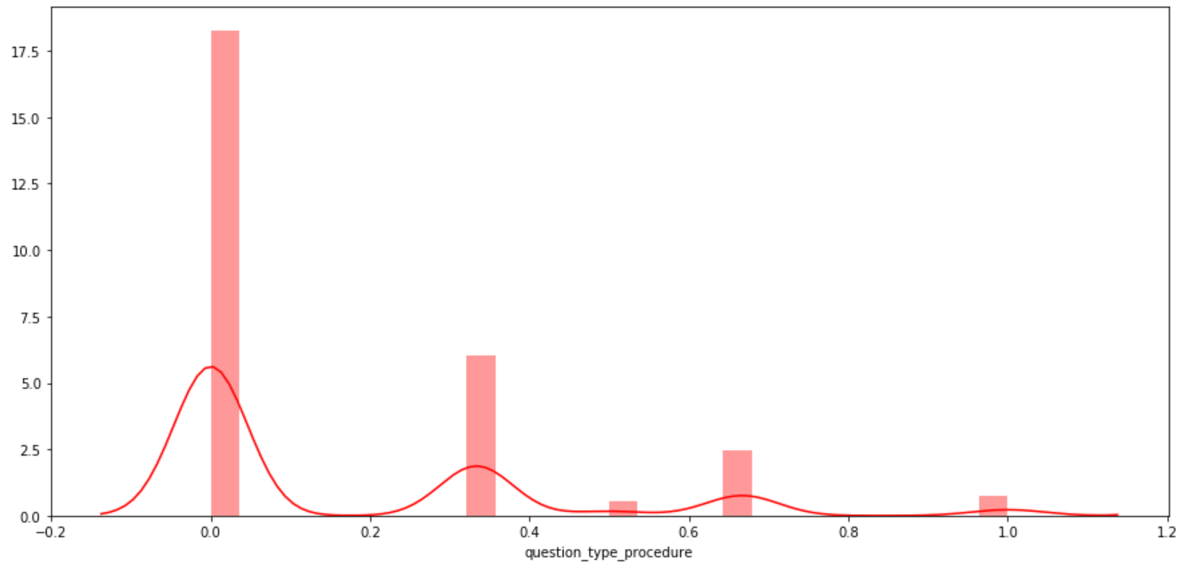

40
30
20
10
0
0.0
0.2
0.4
0.6
0.8
1.0
question_type_compare

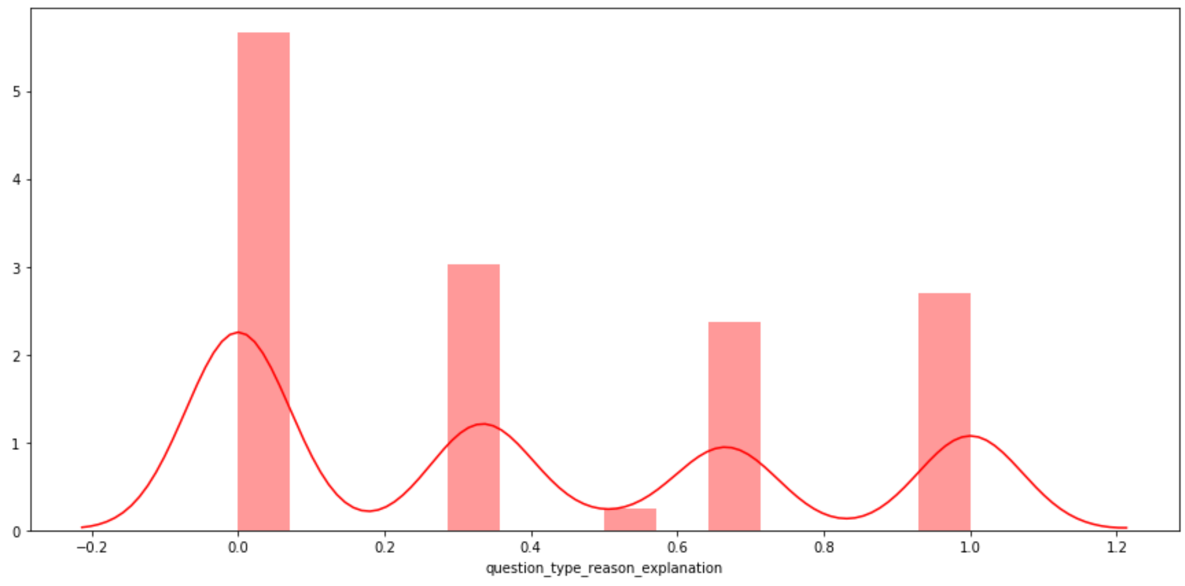

50
40
30
20
10
0
0.0
0.2
0.4
0.6
0.8
1.0
question_type_consequence


question_type_definition
0
10
20
30
40
0.0
0.2
0.4
0.6
0.8
1.0

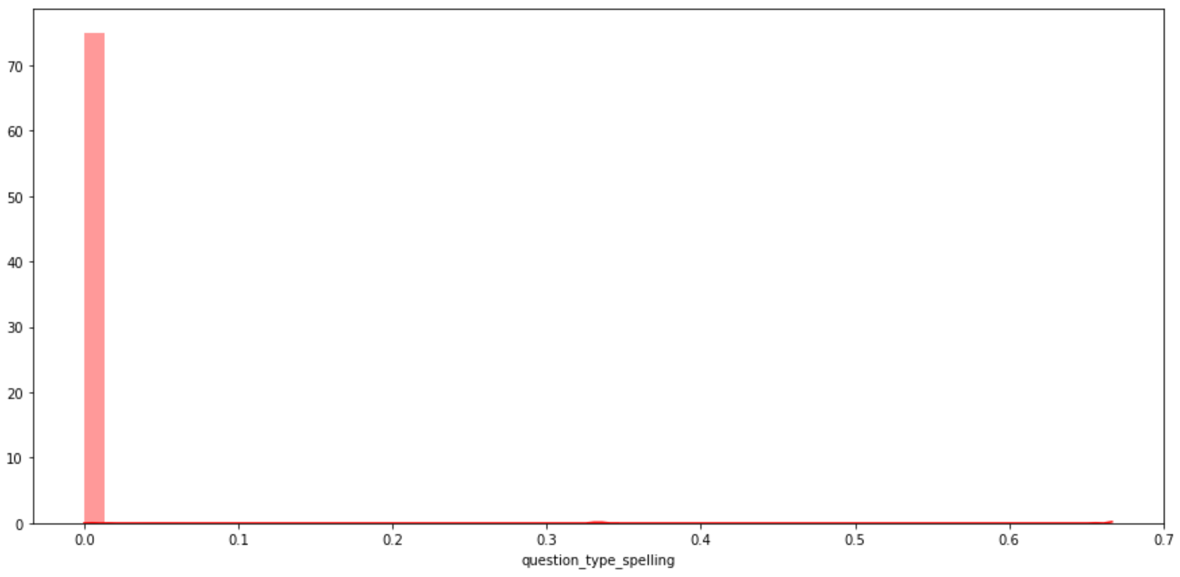

question_type_entity
0
10
20
30
40
0.0
0.2
0.4
0.6
0.8
1.0


3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2
question_type_instructions

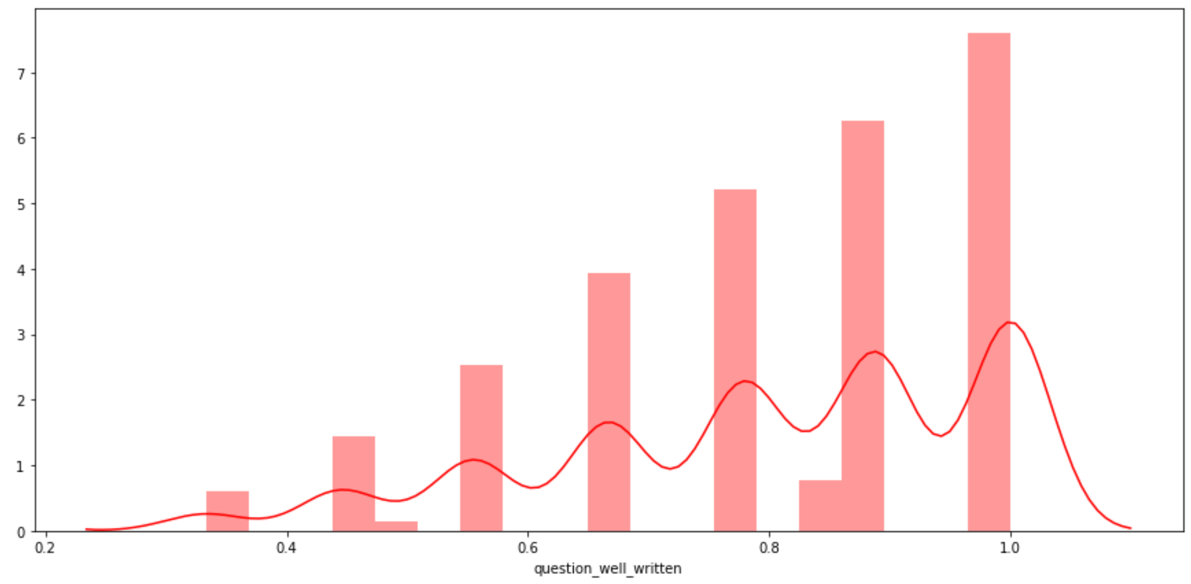

17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2
question_type_procedure



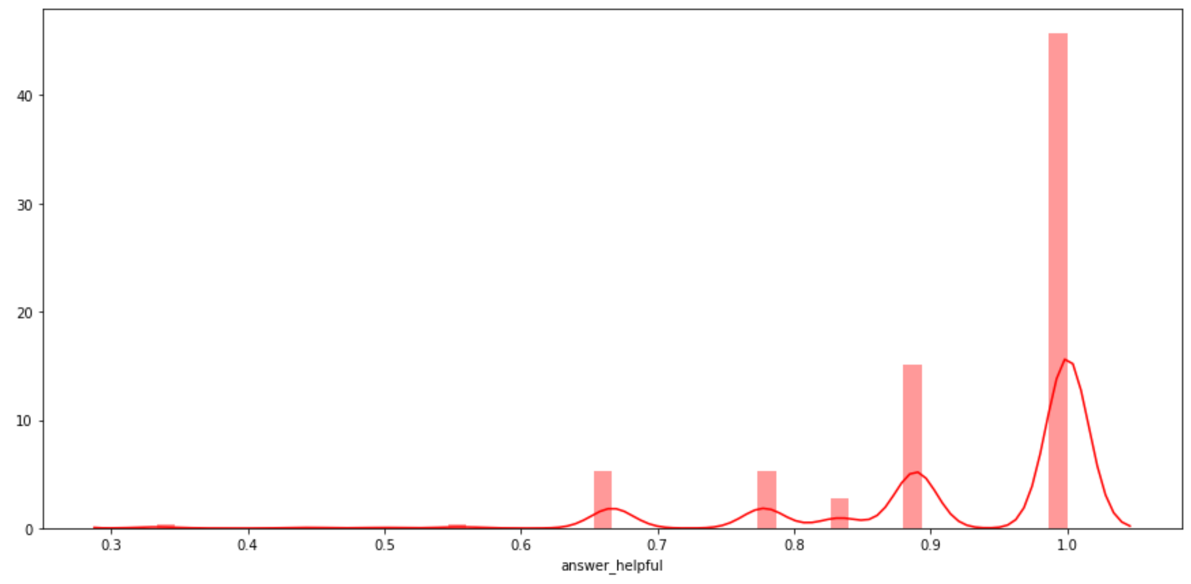
question_type_reason_explanation



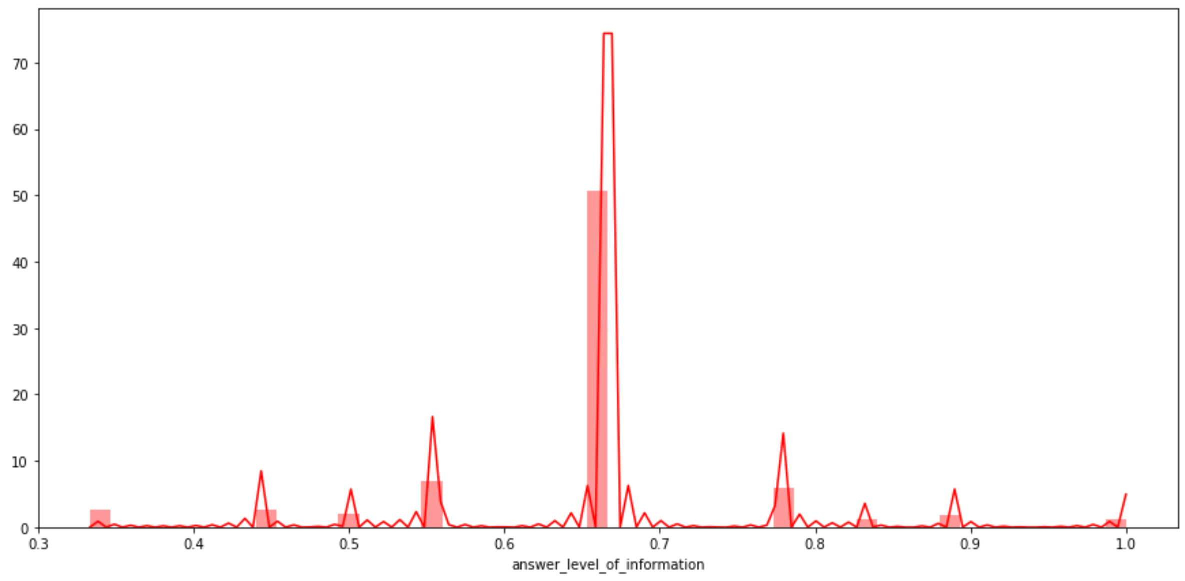
question_type_spelling

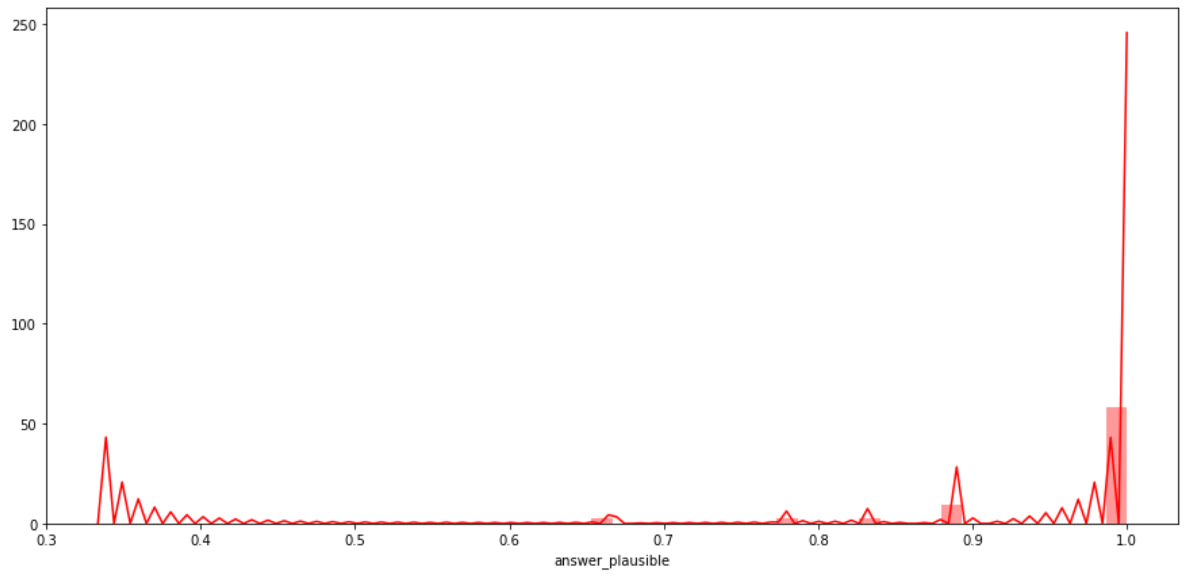


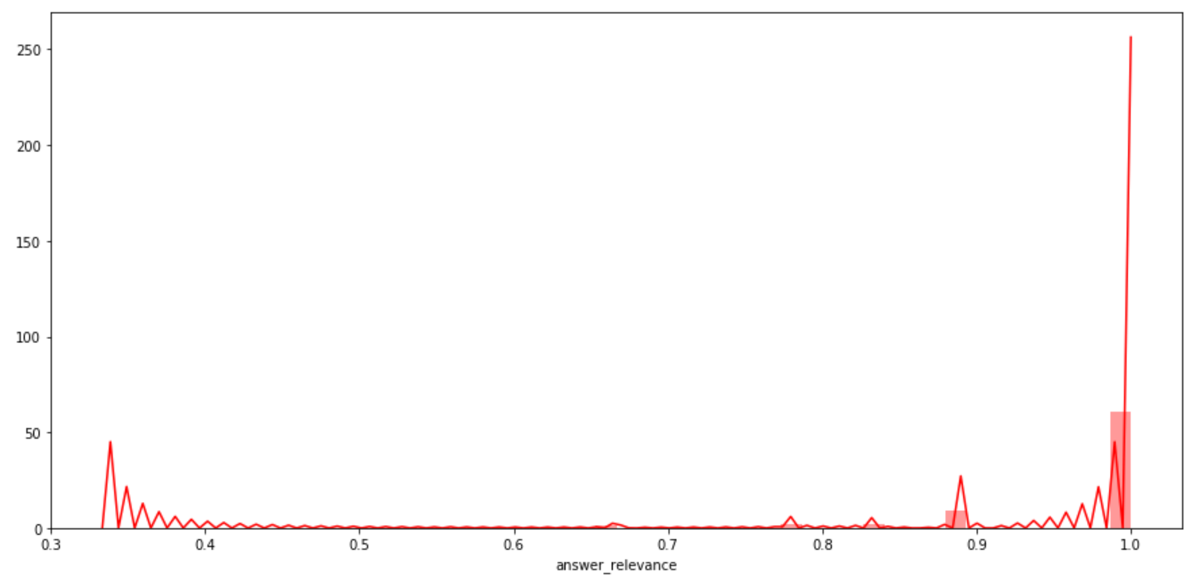
0
1
2
3
4
5
6
7
0.2
0.4
0.6
0.8
1.0
question_well_written



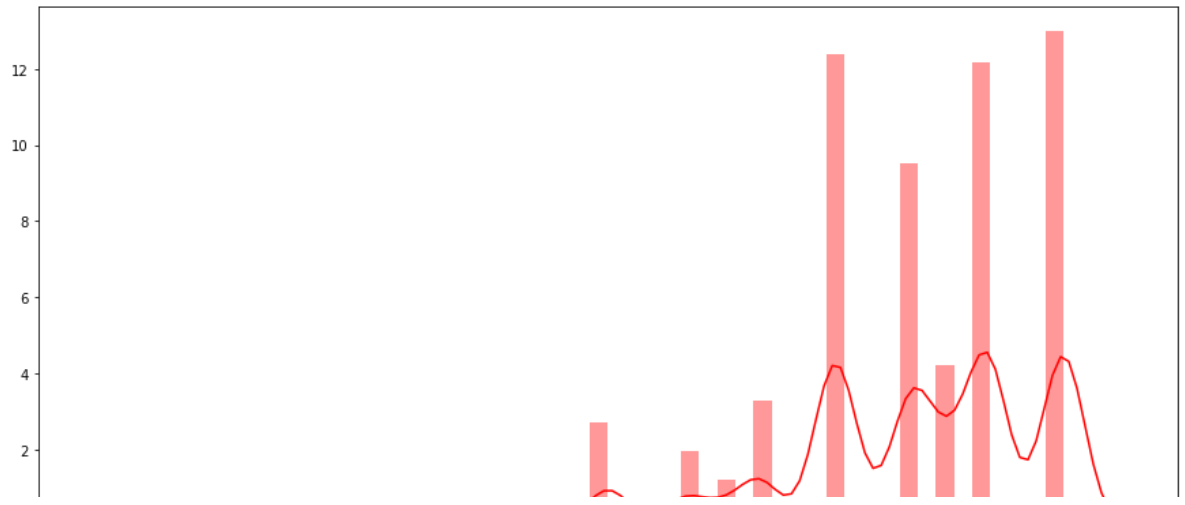
0
10
20
30
40
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
answer_helpful

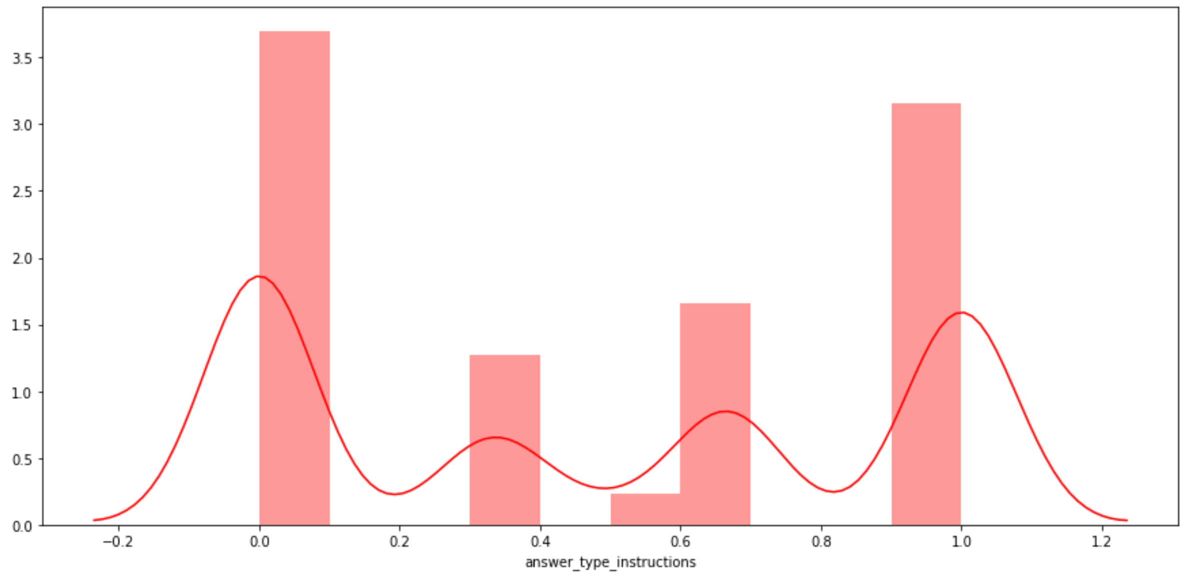

70
60
50
40
30
20
10
0
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
answer_level_of_information

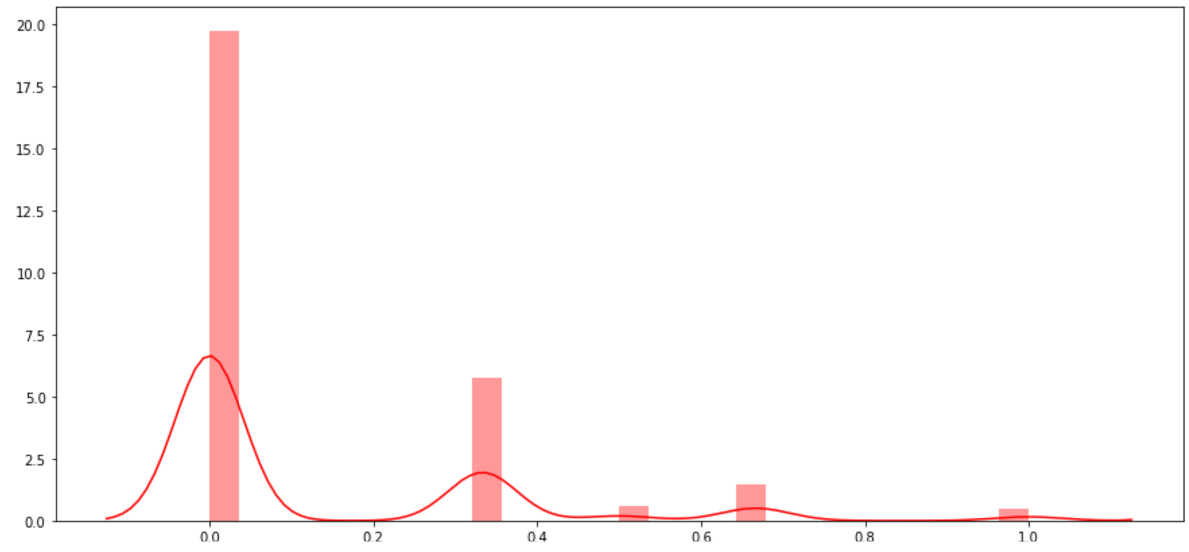

250
200
150
100
50
0
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
answer_plausible

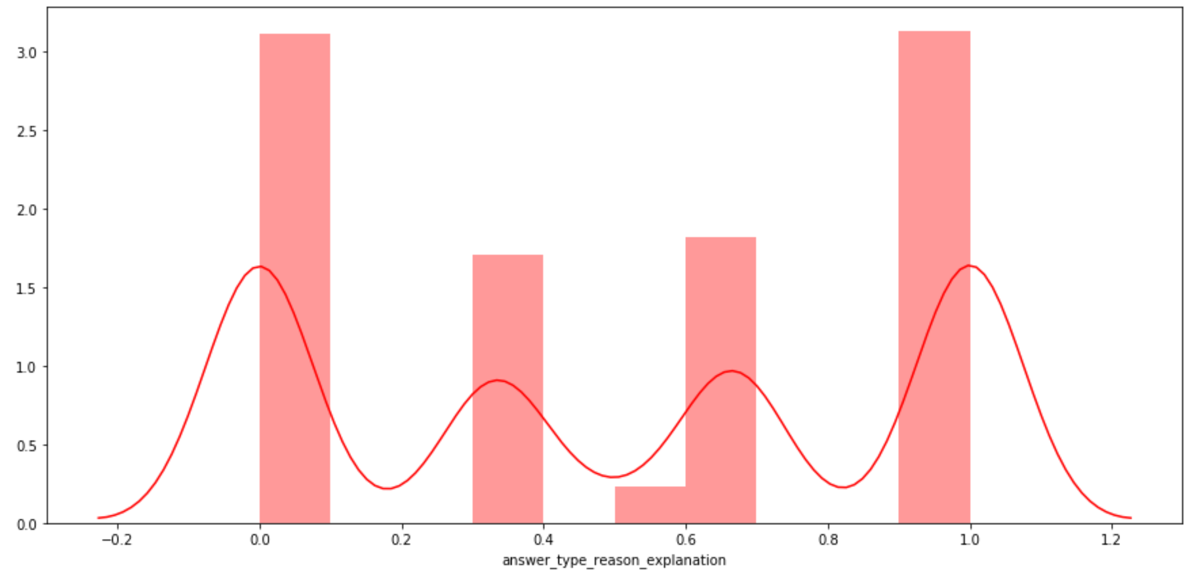


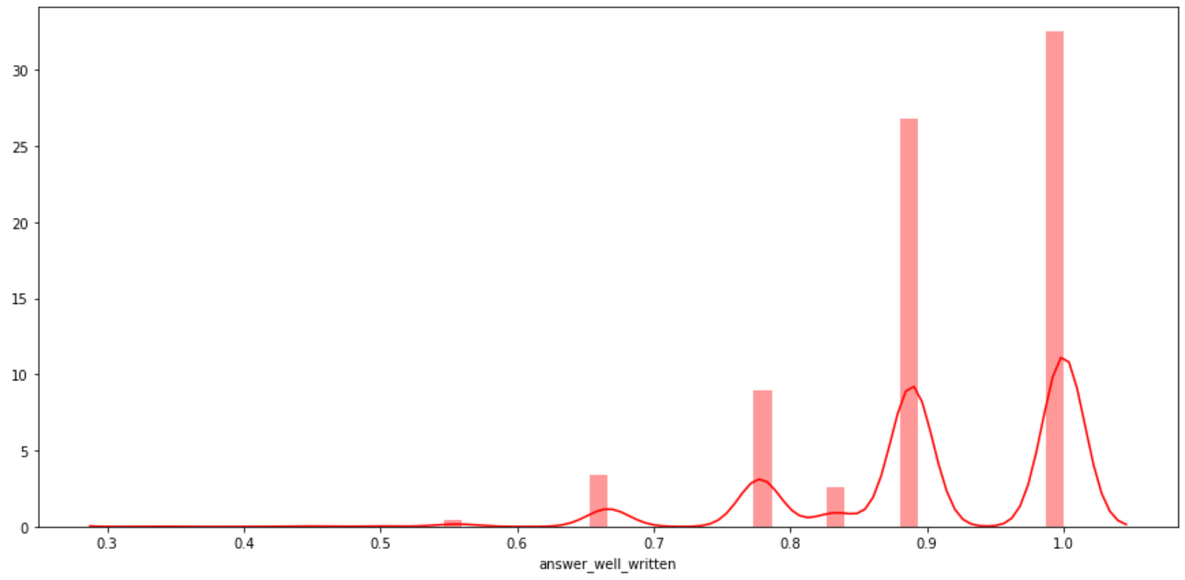
250
200
150
100
50
0
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
answer_relevance


12
10
8
6
4
2


3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2
answer_type_instructions


20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0


3.0
2.5
2.0
1.5
1.0
0.5
0.0
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2
answer_type_reason_explanation



obseravtions

1. The train data contains many features among them 20 features are for training and 30 features are for target features.

2. Among these 20 features only 3 features are useful for predicting the target.

3. The three train features have more words for each row. Only small percentage of questions have highest number of words. So we can avoid not taking the highest number of words as processing. It is better to take number of words which have average number of words.

4. After observing the target feature, all are well distributed from o to 1 probabities but these features question_not_really_question, queation_type_consequence, question_type_spelling data have all data features with probabaility 0. If we implement these features for training we almost will get probabality zero for test data.

```
In [ ]:
```