EDA_Questions

December 20, 2018

1 Exploratory Data Analysis

1.0.1 Project: Quora Insincere Questions Classification Project

The following notebook will explore the data set provided in the quora kaggle competition.

In [1]:
```
# Loading libraries
from sklearn import model_selection, preprocessing, linear_model, naive_bayes, metrics
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
import pandas as pd
import numpy as np
import string as st
from tqdm import tqdm
import matplotlib.pyplot as plt
import seaborn as sns
import nltk
from wordcloud import WordCloud, STOPWORDS
%matplotlib inline

# Loading helper functions
import helper as h
```

1.0.2 1. Basic Statistics

In [2]:
```
# Loading Data
tqdm.pandas()
train_set = pd.read_csv('Data/train.csv', encoding = 'latin1')
train_set.sample(5)
```

Out[2]:

	qid	question_text	target
397153	4dce76d57a94d5e67e9d	I'm currently in college and single. I'm inter...	0
747047	92562da7ec89d3eeb5e1		
706777	8a6406b6fdcd7fe14b18		
883533	ad19ffb2f0ad846eb5cc		
402787	4eed59d4db910b270c40		

```
747047         Which Dolce Gusto coffee is the strongest?       0
706777  How did the invention of electricity impact in...       0
883533     What is the best way to deal with gobby women?       1
402787                   What me must study for C. B. I.?       0
```

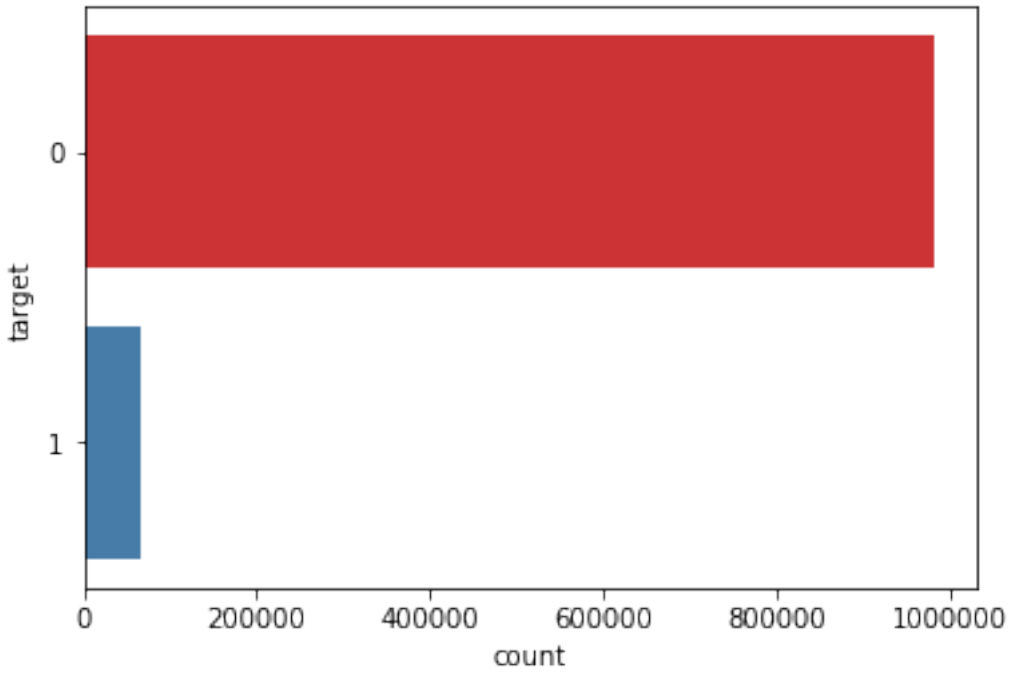
```
In [3]: train_set.shape
```

```
Out[3]: (1048575, 3)
```

The data consists of 3 columns, the question id (qid), the questions text (question_text) and target column (target). The is a total 1048575 rows.

```
In [4]: # Target variable count
        ax = sns.countplot(y="target", data=train_set, palette="Set1") # strong class imbalanc
        cnts = train_set.target.value_counts()
        print("Strong class imbalance. "
              "There are {} of insincere question that represent"
              "the {}% of the data".format(cnts[1], round(cnts[1]/sum(cnts)*100, 1)))
```

```
Strong class imbalance. There are 64774 of insincere question that representthe 6.2% of the da
```



1.0.3 2. Text Cleaning and preprocessing

Text is the most unstructured form of all the available data, thus various types of noise are present in it. It's necessary to clean, standarise and normalize to make it readily analyzable. I will perform the following text preprocessing pipeline:

- Noise Removal: stopwords and punctuations mark.
- Lexicon Normalization: Mainly lemmatization.

```
In [5]: # Lowercasing
        train_set['qt_cleaned'] = train_set.question_text.progress_apply(lambda t: t.lower())

        # Stopwords removal
        train_set['qt_cleaned'] = train_set.qt_cleaned.progress_apply(lambda t: h.remove_stopw

        # Punctuation marks removal
        translator = str.maketrans('', '', st.punctuation)
        train_set['qt_cleaned'] = train_set.qt_cleaned.progress_apply(lambda t: t.translate(tra

        # Lemmatization
        train_set['qt_cleaned'] = train_set.qt_cleaned.progress_apply(lambda t: h.lemmatizer(t)

100%|| 1048575/1048575 [00:01<00:00, 711358.40it/s]
100%|| 1048575/1048575 [02:34<00:00, 6782.09it/s]
100%|| 1048575/1048575 [00:02<00:00, 355696.43it/s]
100%|| 1048575/1048575 [02:28<00:00, 7082.57it/s]
```