Big Data Project-1
Quora Insincere Questions Classification

By

Yash Kasundra (a1838670)

Masters of Data Science

Outline

- Introduction
- Methodology
- Implementation
- Results and Findings

Introduction

- To classify the questions asked on Quora:
  - Sincere (Good, informative, genuine questions)
  - Insincere (Bad, troll, offending, sexually inappropriate questions)
  - Dataset: Obtained from Kaggle website
  - Goal: To build or find a good model which can classify these questions into above mentioned categories.

- Property:
  - Binary Classification problem
  - There is only 1 actual feature that can be used i.e. “question_text”.
  - We are to make some features from this column in order to get good accuracy
  - Check number of samples in both files.

```
Number of data points in training data: 1306122
Number of data points in test data: 375806
```

Methodology

- Use 3 simple algorithms with same preprocess and method to convert text into vector/ tokens(TF-IDF).
  - Logistic Regression
  - Linear Support Vector Classifier(Linear SVC)
  - Stochastic Gradient Descent (SGD)
- Methods to solve imbalance in the class
  - UnderSampling
  - OverSampling
- Using Neural Network and Convolutional Neural Network
  - Check how different parameters affects the results of Neural Networks
  - Used an pre trained embedding layer with transfer learning to make it learn from our data as well
  - Also used tokenizer library in CNN

Implementation

1. Exploratory Data Analysis (EDA)
2. Pre-processing the data to clean unwanted text
3. Applying Simple models
4. Applying methods to deal with imbalance classes
5. Applying Neural Networks and CNN

1) Exploratory Data Analysis (EDA)

- Check if any Null or duplicate values.
- Distribution among the 2 classes.
- Wordclouds to see most used words. This first image is an example of word cloud created for insincere questions.
- N-gram frequency plots to see word combos used most. This is a frequency plot for single most used words in our data.
    - There are 2 more plots with 2-words and 3-words frequency plot.

```
df_train['question_text'].isnull().sum(), df_test['question_text'].isnull().sum()
```

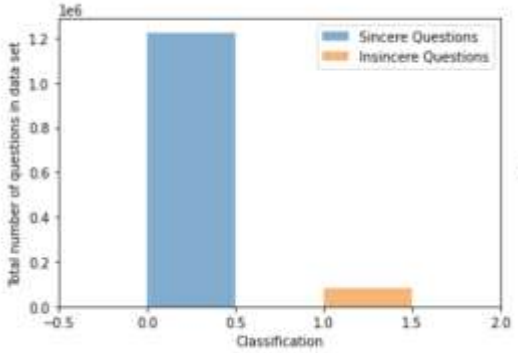
```
]: (0, 0)
```

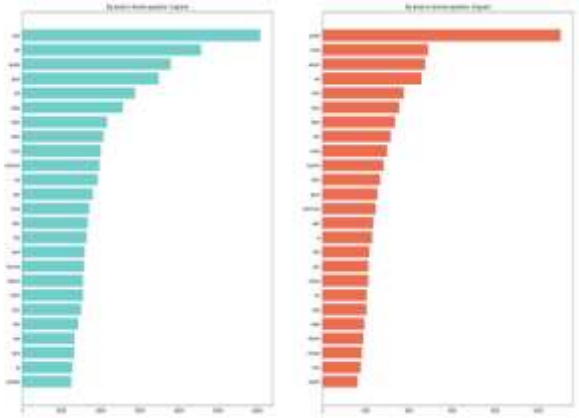
No null values is present in this data set

```
df_train.duplicated(subset={"question_text","qid","target"}).value_counts()
```

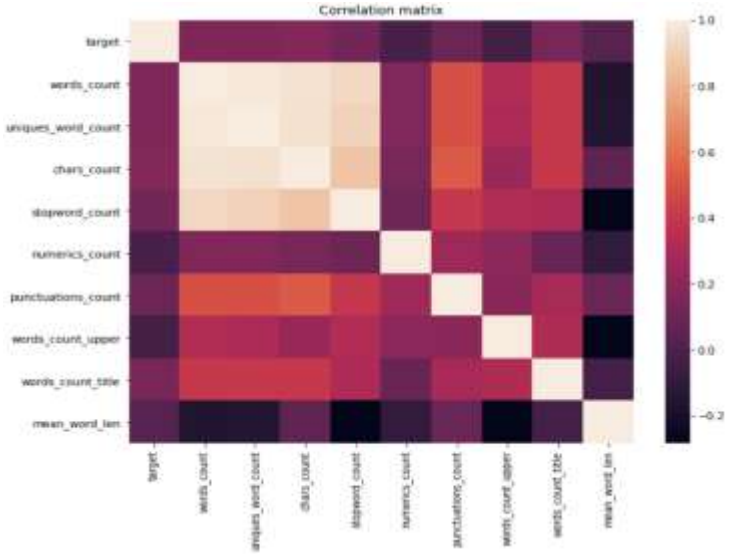
```
7]: False    1306122
    dtype: int64
```

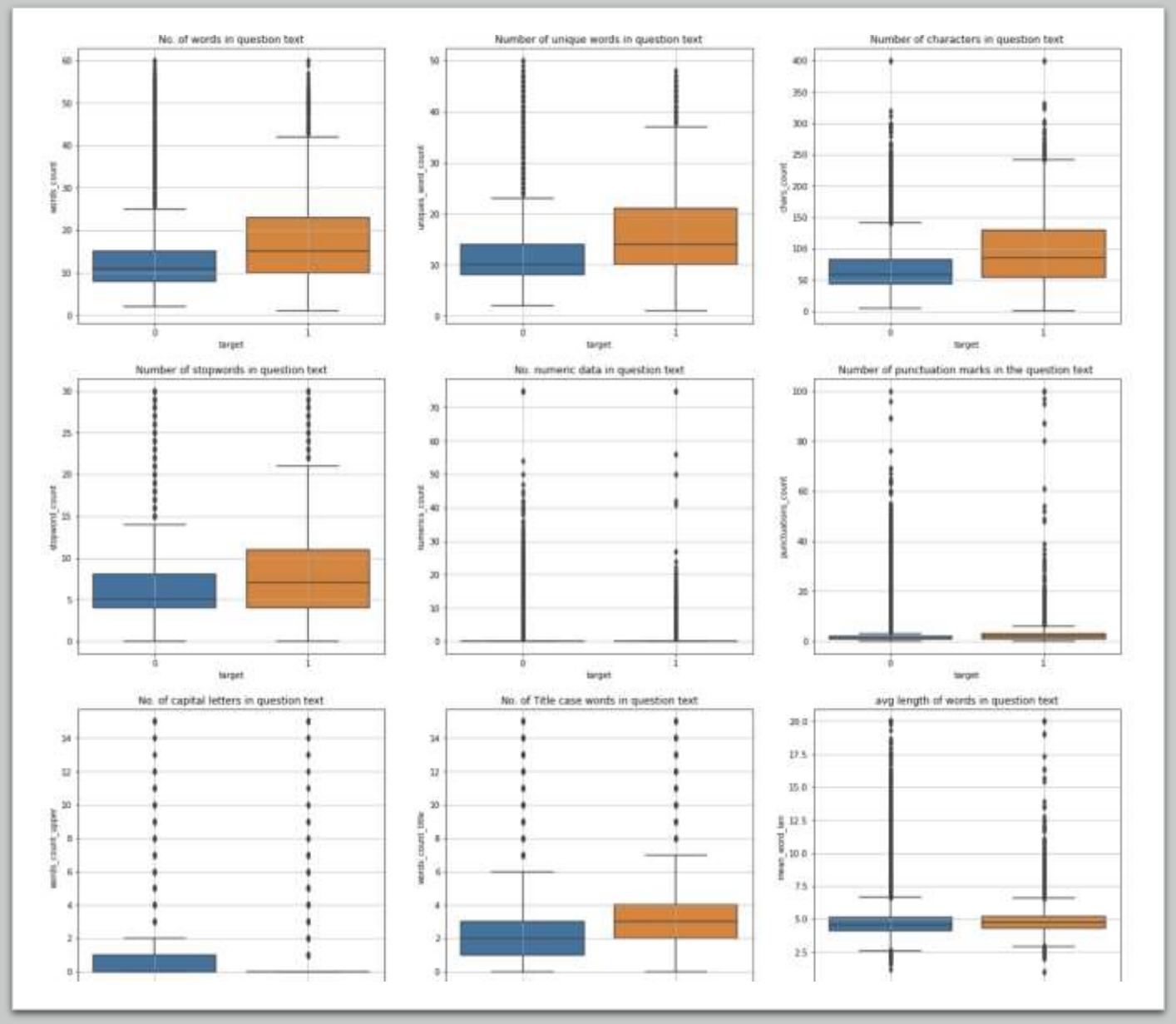
No duplicate values have been found for all rows i.e. 1306122





- Now time to create some freatures from text data and then visualize them using graphs
    1. Number of words in the text
    2. Number of unique words in the text
    3. Number of characters in the text
    4. Number of stopwords
    5. Number of special characters
    6. Number of punctuation marks
    7. Number of upper case words
    8. Number of title case words
    9. Average length of words
- Also I have plotted violin graphs for better understanding of these features
- Below is the correlation matrix for these features.





2) Data Pre-processing

- I created few function for :
  - Replacing math equations and url's with text “MATH EQUATION” and “URL”.
  - Cleaning contractions. ("We'd": "We had“ , "O'Clock": "Of the clock")
  - Spelling Correction. ('organisation': 'organization’, 'cryptocoin':'bitcoin’)
  - Removing punctuations. ('@', '£', '·', '_', '{‘)
  - Removing Stopwords. (‘is’, ’a’, ’an’)
  - Using WordNet Lemmatizer (“Causing” : “cause” , “hunted” : “hunt”)
- Then again visualize by creating few features, to gain insights.

3) Simple Model Creation

1. To split our training data into 3 parts i.e. train,test and validation data

2. To convert the text into vectors/tokens so that it can be understood by our models. For now we are using TF-IDF(term frequency–inverse document frequency) for this task.

3. Logistic Regression it the first model used.

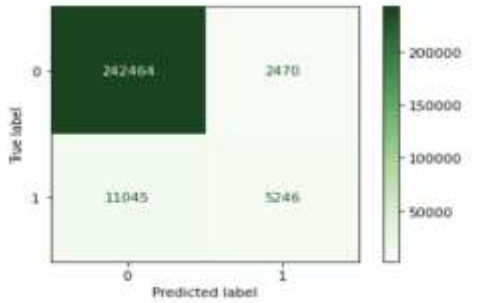
4. Stochastic Gradient Descent

   It had the worst outcome of all, and I think it was due to the fact that there were too many local minima to converge to and the data had too high imbalance between the classes.

```
X_train:  (940407, 13) (940407,)
X_test:  (261225, 13) (261225,)
X_val:  (104490, 13) (104490,)
```

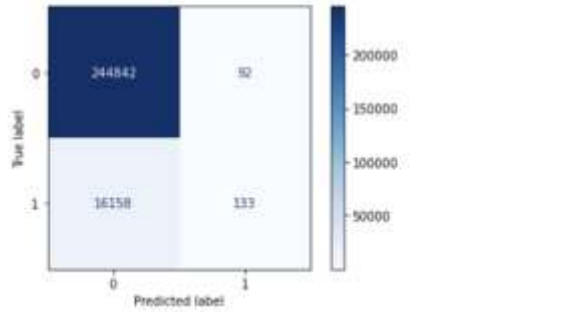
	precision	recall	f1-score	support
0	0.96	0.99	0.97	244934
1	0.68	0.32	0.44	16291
accuracy			0.95	261225
macro avg	0.82	0.66	0.70	261225
weighted avg	0.94	0.95	0.94	261225

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisp
```



	precision	recall	f1-score	support
0	0.94	1.00	0.97	244934
1	0.59	0.01	0.02	16291
accuracy			0.94	261225
macro avg	0.76	0.50	0.49	261225
weighted avg	0.92	0.94	0.91	261225

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDispla
```



4) Handling Imbalance Classes

- Using Undersampling
  - Problem with this approach is we have to ignore too much good data to balance out both classes.

```
# Undersample 0-class and concat the DataFrames of both class
df_class_0_under = df_class_0.sample(250000) # selects random samples
df_test_under = pd.concat([df_class_0_under, df_class_1], axis=0)

print('Random under-sampling:')
print(df_test_under.target.value_counts())
```

```
Random under-sampling:
0    250000
1     80810
Name: target, dtype: int64
```

- Hybrid method of Using oversampling as well as undersampling
  - To handle the above mentioned problem

```
# Oversample 1-class and concat the DataFrames of both classes
df_class_1_over = df_class_1.sample(320000, replace=True)
df_test_over = pd.concat([df_class_0_sample, df_class_1_over], axis=0)

print('Random over-sampling:')
print(df_test_over.target.value_counts())
```

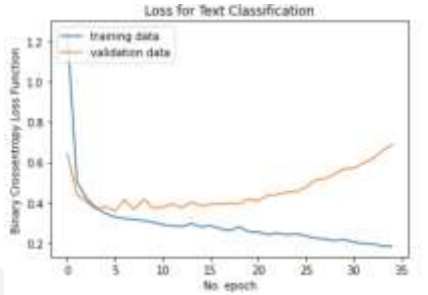
```
Random over-sampling:
0    650000
1    320000
Name: target, dtype: int64
```

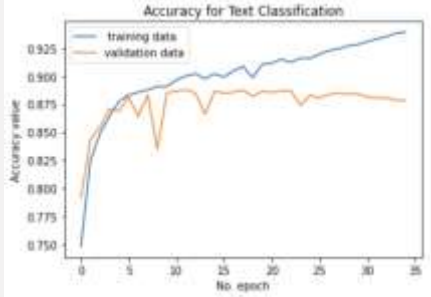
5) Applying Neural Networks and CNN

- Adding graphs for basic Neural Network

1. With undersampling and basic model (1 input layer, 1 dense, 1 output layer)

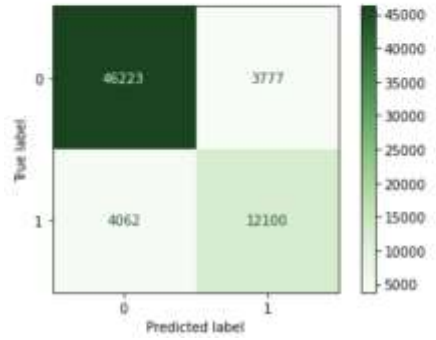
   Overfitting on training data





	precision	recall	f1-score	support
0	0.92	0.92	0.92	50000
1	0.76	0.75	0.76	16162
accuracy			0.88	66162
macro avg	0.84	0.84	0.84	66162
weighted avg	0.88	0.88	0.88	66162

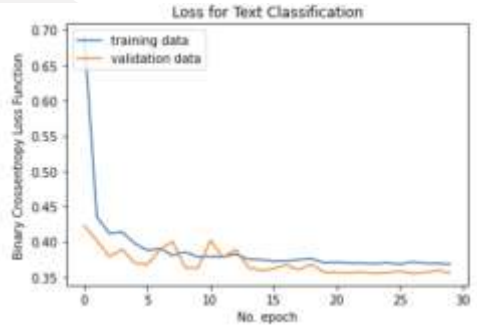
<sklearn.metrics._plot.confusion_matrix.ConfusionMatri

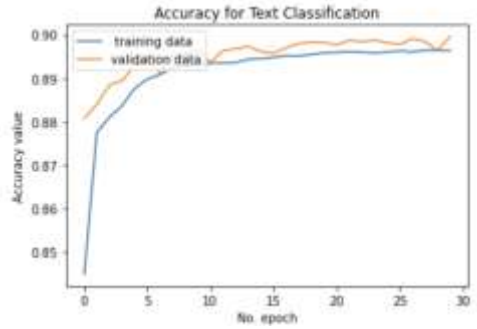


- Convolutional Neural network

  Using Convolutional-1d layer and maxpooling

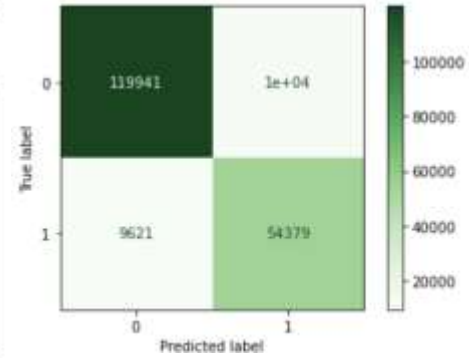
  As well as dropout layer





	precision	recall	f1-score	support
0	0.93	0.92	0.92	130000
1	0.84	0.85	0.85	64000
accuracy			0.90	194000
macro avg	0.88	0.89	0.89	194000
weighted avg	0.90	0.90	0.90	194000

<sklearn.metrics._plot.confusion_matrix.ConfusionMatri



Results and Findings

	Model	Score
1	Linear SVC	0.473760
0	Logistic Regression	0.437039
2	Stochastic Gradient Decent	0.016106

- Evaluations using F-1 scores and later on MCC(Matthew Correlation Coefficient).

	Models	Matthew Correlation Coefficient
1	Over/UnderSampling and Simple NN	0.859949
2	Deeper Neural Network	0.858250
4	CNN	0.773810
3	Adding dropour layers	0.740290
0	UnderSampling and Simple NN	0.665191

- Even though MCC of CNN and model-3(dropout layer) is lower than other 2, they are actually performing better than simple models, due to the fact that loss and accuracy curves are not converging.

  Thus, they were overfitting on train data.

What next ?

- To increase the accuracy of the models, we can try using different activation function, optimizers , loss functions, I’ve not used too much sigmoid or other functions due to high computational cost.
- We can explore other libraries to handle imbalance of the 2 classes.
- We can also create a deeper or more wider or both Neural Network to see, which is more suitable for our use.
- We can also use deep Learning algorithms or pretrained models like BERT (from google) or roBERTa (facebook ai).