

Dissertation Project

Subject: Quora Insincere Question Classification

PGDPA 2021-22

For

Toxic Content Detection on Quora's Question Answer Website

Submitted to

Bombay Stock Exchange Institute, Mumbai

18th & 19th Floor, P J Towers, Dalal Street

Mumbai – 400001

(2021-22)

By

Mr. Arvind Hemant Bondkar – PGDPA2101

Under the Guidance of

Prof. Chirag Jhumkhawala



Acknowledgement

The successful completion of this project, “**Toxic Content Detection on Quora's Question Answer Website**” would be incomplete without my mentor Mr. Chirag Jhumkhawala.

I express gratitude and respect towards all those who guided me through completion of this project.

I convey thanks to BSE Institute Ltd for providing me with the opportunity to work on this project.

Mr. Arvind Hemant Bondkar

(Research Candidate)

Date:

Place: Mumbai

Declaration

I hereby declare that the project entitled “**Toxic Content Detection on Quora's Question Answer Website**” is submitted by me in partial fulfilment of the requirement for the award of Post Graduate Diploma in Predictive Analytics. This project report is my original work and has not been submitted to any other university or institute for the award of any degree or diploma.

Mr. Arvind Hemant Bondkar

Roll No. PGDPA2101

Batch: 2021-22

CERTIFICATE

I, **Mr. Chirag Jhumkhawala** Certify that,
Mr. Arvind Hemant Bondkar is a student of BSE Institute Ltd of
Post Graduate Diploma in Predictive Analytics (PGDPA) and
I hereby declare that he has

Successfully completed his project on

**“Toxic Content Detection on Quora's Question Answer
Website”**

In the Academic year 2021-22

The information submitted is true, original, to the best of my knowledge
and appropriate sources have been cited.

Date:

Place: Mumbai

Signature of Faculty

Prof. Chirag Jhumkhawala

Signature of Student

Mr. Arvind Bondkar

TABLE OF CONTENTS

Sr No.	Title	Page No.
	Declaration	3
	Certificate	4
	List of Figures	7
	Abstract	9-10
Chapter 1	Introduction	11-13
1.1	Aim	12
1.2	Motivation of the research	13
Chapter 2	Statement of the problem	14
Chapter 3	Literature Review	15-18
Chapter 4	Need, Scope & Objective of the study	19
Chapter 5	Research Methodology	20-21
Chapter 6	Analysis & Findings	22-79
6.1	Data Understanding	25-26
6.2	EDA - Exploratory Data Analysis	27-39
6.3	Data Preprocessing & Cleaning	30-46
6.4	Modeling	47-48
6.5	Algorithms & Evaluation of Metrics	50-54
6.6	Model Building	55-64
6.7	Resampling	65-76
6.8	Prediction System	77-80
6.9	Model Deployment	81-88
Chapter 7	Conclusion	90
Chapter 8	Recommendations	91

Chapter 9	Bibliography	91
Chapter10	Annexure	92
Chapter11	Plagiarism	93-94

List of Figures

Fig Flow Chart on Methodology

Fig(1) First 5 rows

Fig (2) Target Count & Distribution

Fig (3) WordCloud of Sincere Questions

Fig(4): WordCloud of Insincere Questions

Fig(5): Top Words in Sincere & Insincere questions

Fig(6): Analysis of Extracted features

Fig(7) Correlation Matrix

Fig(8): Dependencies

Fig(9): Classification Report for Imbalance NB

Fig(10): Confusion Matrix Imbalance NB

Fig(11): ROC – AUC Curve Imbalance NB

Fig(12): Classification Report Imbalance LR

Fig(13): Confusion Matrix Imbalance LR

Fig(14): Over Sampling

Fig(15) Classification Report LR

Fig(16): Vectorizer

Fig(17): CR for LR

Fig(18): ROC – AUC curve LR

Fig(20):After Model Building words

Fig(21): Prediction System

Fig(22): Prediction Output

Fig(23)Model Deployment

Fig(24): Streamlit Interface

Fig(25):Insincere question output

Fig(26): Sincere question Output

Fig(27):Plagiarism Report

Abstract

The internet has opened up a whole new world of possibilities. It has answers to all questions, and if none are found, there are websites where you can ask any question and have people from all over the world respond. One such question-and-answer website is Quora. Quora is a website that encourages people to ask questions and learn from one another. Anyone can post a question from any topic, and others will respond. However, some inquiries do not make sense and contain a variety of emotions. People abuse this benefit and cause turmoil by asking questions that should not be asked in public.

The goal of this project is to create a system that detects "insincere" questions and takes necessary action against them. If the queries have a non-neutral tone, are insulting, provocative, or contain negative sexual overtones, and are not anchored in truth, they are classified as insincere.

Identifying and removing 'toxic' content is an issue for today's large websites. By the time a user reports hazardous content and the website takes action, the content may have already caused significant damage.

My research is based on the 'Quora Insincere Questions Classification' dataset (from Kaggle). The dataset contains both serious and insincere queries, with the majority being sincere.

Python and its libraries, such as sklearn (Sci-kit Learn), numpy, pandas, and Nltk (Natural Language Tool Kit), are used to process and analyse the data. Word embeddings such as GloVe, Wiki-news, and TF-IDF are used to convert the dataset to vector form, although I only implemented TF-IDF.

Resampling strategies are used to deal with the dataset's imbalance. To find the best outcomes, we train and compare two machine learning models called Logistic Regression and Naive-Bayes.

Box-Plots, Distribution plot, WordCloud Visualization, Bar Chart, etc. I made for visualization and findings.

F1 score, recall, precision, and Area Under Curve are among the metrics used to assess the outcome.

Chapter 1 – Introduction

Over time, the internet has grown in popularity.

It has now become one of the most critical aspects of human survival. The way the internet simplifies tasks that were before difficult is the fundamental reason for its appeal. One of the most common uses of the internet is to look for answers to queries. This online capability has aided in numerous ways. However, this was quickly improved to asking questions on a website where individuals from all over the world could contribute to the answers. Question forums are the name for these kinds of websites. They've grown in appeal as a result of their simple to utilise and comprehend methods. Question forum websites such as Quora, Stack Overflow, and Yahoo Answers are examples.

Stack Overflow is mostly a technical question site where users post questions or code faults, and other users from across the world read, evaluate, and try to find a solution. Yahoo Answers and Quora are both based on the same concepts. Simple, personal, and professional questions can be asked on both websites. Some people utilise these websites to seek job guidance from a personal perspective. This facilitates the formation of a community and the assistance of one another during difficult times.

However, there is a significant risk of persons abusing these boards. People have a habit of asking inappropriate inquiries. They could be directed at a certain group. Or make no sense at all. These inquiries are common to raise a ruckus and so disrupt the main cause designing these types of websites that assist one another. As a result, it is critical to take action.

categorise and eliminate these questions before they become a problem.

harm the website's reputation. This paper is about We present categorization models for disingenuous people. questions that will aid in the preservation of the integrity of the internet website.

Quora is the website where the information is gathered. It is one of the most frequently visited Question Forums.

It believes in encouraging people to share their knowledge. It is critical for the website to ensure that people from all over the world can safely share information and ask inquiries. Manual Reviews were previously used by Quora to distinguish between genuine and fake queries. This procedure, however, can be made more efficient with the application of Machine Learning Models.

For the classification of the questions, supervised learning models are used. Naive Bayes and Logistic Regression are two of the models developed. Both models have a high level of accuracy. However, because the data is imbalanced, F1 score and ROC will be used to evaluate the models' performance. For counting the occurrences of words in the questions, Term Frequency–Inverse Document Frequency (TF-IDF) is used. Lemmatization and stemming are also employed to reduce inflection forms and remove ends, resulting in a base word.

We will later discuss about some of the related work, focuses on the problem statement, description of the dataset and Exploratory data analysis. How the data was prepared and represented. how the imbalance in the dataset was handled, and so on.

Aim:

To balance the data and then clean the data to get best model accuracy to make prediction system for insincere question detection.

Motivation of the research:

The main goal of this research is to try to attempt an NLP problem into a generalised machine learning problem that can be solved using a variety of machine learning approaches.

- To formulate/featurize it as a machine learning challenge, use various language models and solve it using various generalised ML approaches.
- Data size, algorithmic choice, and hyper parameter choice benchmarks.

Quora

Chapter 2 – Statement of the problem

Quora is a prominent online platform where users may ask questions and (usually) receive thoughtful responses.

While the majority of questions are posed in good faith, a small minority of bad actors offer questions that are either disingenuous or problematic (for example, questions based on misleading premises or questions that are just intended to make a statement).

In order to improve the overall community experience, **a task is to create a classifier that will take a user-generated question as input and automatically categorise it as Sincere or Insincere.**

Chapter 3 – Literature Review

In any fraud detection or more of the binary classification dataset imbalance data problem we face in this kind of data sets, and we have to use resampling, SMOTE library, or embedding methods to balance the dataset. In our problem I used resampling method.

For text mining, there are so many methods have been used like Neural Network, LSTM and GRU, etc. But I used Supervised machine learning method for this like Logistic regression and Naïve Bayes. Before going towards algorithms. We have to do preprocessing like Cleaning, Transformation and lemmatization, Stemming and many things.

For this project I searched and read many articles and research papers, there are so many great researchers have been completed research on this Quora insincere classification.

Following are the researchers' thoughts and findings:

As described in the title Convolutional Neural Networks for Sentence Classification, Yoon Kim has performed a series of experiments with CNN on pretrained word-vector for sentence classification. According to his research, a simple model (CNN-statics) with some hyper parameter turning and static vectors outperforms a more intricate deep learning model using complex pooling algorithms. Four out of seven tests, including sentimental

analysis and question classification, show that this CNN model exceeds the competition.

CNN is commonly used for image classification, according to their article. CNN for NLP is not widely utilised and is simple to use. They employed a two-tier CNN system that categorises questions into main and subcategories. They employed a two-tier CNN system that classifies questions into main and subcategories. The architecture comprises of a Convolutional layer that learns many filters for different heights (Bi-grams to Pent-grams), followed by a 2-max-pooling layer that gathers more data from the convolution layer. With nodes 128 and 64, all of the maxpooled layers were merged to generate a 2-fully linked layer.

The University of Illinois, Urbana Champaign, questioned the classification dataset used for training. The Quora dataset, which was manually collected, was determined to have 90.43 percent main category accuracy and 76.52 percent subcategory accuracy while testing their algorithm. The major category accuracy for TREC was 93.4 percent, while the subcategory accuracy was 87.4 percent.

'Deep Learning for Sentence Classification' was studied by Abdalraouf Hassana and Ausif Mahmood at the University of Bridgeport. The majority of machine learning algorithms, according to the article, need the input to be indicated as a fixed-length feature. A popular fixed-length feature is a bag of words. It's straightforward, yet it's limiting in many ways. They disregard word semantics and word loss ordering. To capture semantic and grammatical information, Long Short-Term Memory (LSTM) is applied

over a pre-trained word vector. They used a pre-trained word vector that was trained on 100 billion words from Google News to try to predict whether an inquiry is genuine. Using a pre-trained word vector has a number of advantages. Similar words are clustered together. To avoid the problem of vanishing gradient, LSTM is utilised. For sentiment analysis, they employed two datasets: Stanford Large Movie Review Dataset IMDB and Stanford Sentiment Treebank (SSTb). Stochastic gradient descent was used to train over shuffled minibatches. The concealed state was to be 128 bytes in size, while the mini-batch size was to be 64 bytes. Validation data was chosen from 0.5-10 percent of the training data. SSTb has a 14.3% mistake rate, while IMDB has an 11.3 percent error rate, according to their model.

In their paper "Exploring Deep Learning in Semantic Question Matching," Ashwin Dhakal and his co-authors used an Artificial Neural Network approach to predict semantic coincidence between question pairs, extracting highly dominant features, and determining the likelihood of a question being duplicated on Quora. Words and phrases are mapped into real-number vectors in their research, then feature engineering is applied, which comprises NLTK mathematics, Fuzzy wuzzy features, and Word mover distances combined with vector distances.

Hence, the research has discussed and following the architecture used by the Quora itself along with the knowledge of natural language processing and machine learning.

In this Kaggle competition 4037 teams have participated till date and every person and team has their own finding and method on text mining. Many of them have used Supervised and Unsupervised methods any many of used only Supervised or Unsupervised only. For Imbalance data problem there are 6% of Insincere minority data imbalance so many of them have not used Resampling or any other method to balance dataset. They used other platforms to get Insincere questions. They Scrap data from their and append those questions.

If this method is giving us best accuracy, then this method is also good to train the model.

Chapter 4 – Need, Scope & Objective of the study

Need:

On any social media platform removing toxic content and abusive or adult content is another task for all.

To Extract toxic content, we need to use NLP text mining techniques scrapping data from any website. It aids in the resolution of verbal ambiguity and provides data with valuable quantitative structure for many downstream applications.

Scope:

Google is also used NLP concepts like Lemmatization and Stemming for SEO optimization and in today's generation keywords are more important in social media life so this technique will use at anywhere.

Example; Google Audio recognition and Text Translation.

Objective of the study:

Read, Understand and decode sentiments from it to train the model.

Chapter 5 – Research Methodology

Methodology used in work:

- 1. Business Understanding:** The quality of business suffers when there are more insincere questions. As a result, this project aims to anticipate these false questions, so increasing the value of business. This also aids in building user confidence in the website.
- 2. Problem Statement**
- 3. Data Collection:** Data collection is the biggest task other than doing data analysis.
- 4. Data Understanding:** By checking data type and shape of the data size. Plotting graphs we can analyse data imbalance, Using WordCloud visualize words occurring in data.
- 5. Data Preparation:** Balance data in well format. Do tokenization, stemming, Lemmatization, remove Stop words, Vectorization, etc.
- 6. Modeling:** Use Classification models and choose the best model. E.g. Logistic Regression, Naïve Bayes, etc.
- 7. Evaluation:** Check Accuracy Score, F1 Score, Precision and Recall.
- 8. Deployment:** After the data is trained, the testing data is fed into the models to see how well the models perform. This testing data is used to calculate the evaluation values.

Flow chart on Methodology:

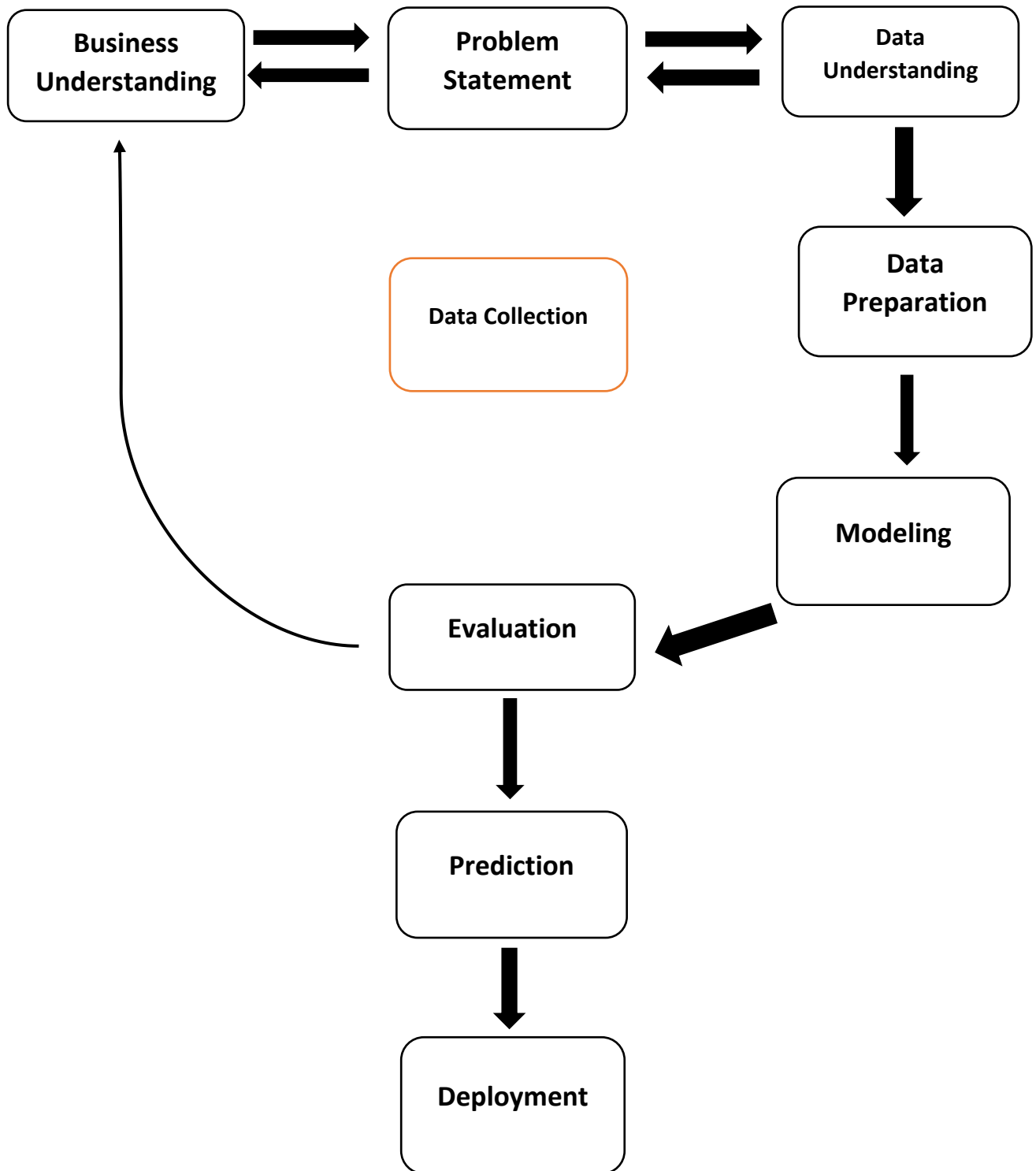


Fig Flow Chart on Methodology

Chapter 6 – Analysis & Findings

Quora Insincere Questions Classification

I used Google Colab Python Notebook for Analysis and Predictions.

Problem Statement

- Detect toxic content to improve online conversations.
- We have to develop models that identify and flag insincere questions.
- Source: [kaggle.com / Quora Insincere Question Classification](https://www.kaggle.com/quora-insincere-question-classification).

Following Information is given on Kaggle website,

General Description

An insincere question is defined as a question intended to make a statement rather than look for helpful answers. Some characteristics that can signify that a question is insincere:

- Has a non-neutral tone
- Has an exaggerated tone to underscore a point about a group of people
- Is rhetorical and meant to imply a statement about a group of people
- Is disparaging or inflammatory
- Suggests a discriminatory idea against a protected class of people, or seeks confirmation of a stereotype
- Makes disparaging attacks/insults against a specific person or group of people
- Based on an outlandish premise about a group of people
- Disparages against a characteristic that is not fixable and not measurable
- Isn't grounded in reality
- Based on false information, or contains absurd assumptions

- Uses sexual content (incest, bestiality, pedophilia) for shock value, and not to seek genuine answers

Data fields

- qid - unique question identifier
- question_text - Quora question text
- target - a question labeled "insincere" has a value of 1, otherwise 0

Note: Some Question text in notebook is offensive.

1)First of all, we have to import dependencies or libraries we say.

```
import numpy as np
import pandas as pd
import re
import string
import nltk
import matplotlib.pyplot as plt
import seaborn as sns
from wordcloud import WordCloud, STOPWORDS
from nltk.util import ngrams
from nltk.stem import PorterStemmer, WordNetLemmatizer
from nltk.stem.lancaster import LancasterStemmer
from nltk.util import ngrams
```

- **numpy**- For numerical sum we use numpy
- **pandas**- For DataFrame
- **re** – Regular Expression for textual data arrangement.
- **String** – For string data type
- **Matplotlib and seaborn** – Both are Graphs visualization libraries.
- **Wordcloud** – Is the textual visualization library which can visualize without stopwords.
- **Ngram** – Collection of words or corpus.
- **PorterStemmer, WordNetLemmatizer, LancasterStemmer**- All are Stemming libraries.

2) Import dataset using pandas dataframe.

```
df_train = pd.read_csv('/content/train.csv')
df_test = pd.read_csv('/content/test.csv')
```

```
df_train.head()
```

	qid	question_text	target	
0	00002165364db923c7e6	How did Quebec nationalists see their province...	0	
1	000032939017120e6e44	Do you have an adopted dog, how would you enco...	0	
2	0000412ca6e4628ce2cf	Why does velocity affect time? Does velocity a...	0	
3	000042bf85aa498cd78e	How did Otto von Guericke used the Magdeburg h...	0	
4	0000455dfa3e01eae3af	Can I convert montra helicon D to a mountain b...	0	

```
df_test.head()
```

	qid	question_text	
0	0000163e3ea7c7a74cd7	Why do so many women become so rude and arroga...	
1	00002bd4fb5d505b9161	When should I apply for RV college of engineer...	
2	00007756b4a147d2b0b3	What is it really like to be a nurse practitio...	
3	000086e4b7e1c7146103	Who are entrepreneurs?	
4	0000c4c3fbe8785a3090	Is education really making good people nowadays?	

Fig(1) First 5 rows

->Using pandas dataframe, I defined **df_train** as train data and **df_test** as test data.

There are two datasets are given

1) Train

2) Test

1.Data Understanding:

>Check Shape of the both datasets.

```
print("Shape of train data:", df_train.shape)
print("Shape of test data:", df_test.shape)
```

Shape of train data: (1306122, 3)
Shape of test data: (375806, 2)

Shape of train data is rows- **1306122**, Columns – **3**

Shape of test data is rows – **375806**, Columns - **2**

*We only Require train data set for training, analysis and modelling purpose.

Train data contains:

>Data fields

- **qid** - unique question identifier
- **question text** - Quora question text
- **target** - a question labeled "insincere" has a value of 1, otherwise 0

The sincerity of questions is decided on the basis of following parameters:

- Has a non-neutral tone
- Is inflammatory or disparaging
- Is not grounded in reality.
- Uses sexual content like incest, bestiality or pedophilia for shock value and not to seek answers of sort.

> Checking for Null values

```
#Checking for Null values  
df_train.isna().sum()
```

```
qid          0  
question_text 0  
target       0  
dtype: int64
```

Conclusion: All 3 variables don't have any null values.

>Checking for target (Output) Variable balance.

```
df_train['target'].value_counts()
```

```
0    1225312  
1      80810  
Name: target, dtype: int64
```

Here we can see the imbalance in the dataset.

“1” means Insincere data count is very less than sincere (“0”).

Finding:

The dataset has **High Variance** and **High Bias** towards Majority of data which is Sincere count.

2.EDA – Exploratory Data Analysis

1. Distribution of data points among output class

```
# Bar chart
plt.subplot(1, 2, 1)
df_train.groupby('target')['qid'].count().plot.bar()
plt.grid(True)
plt.title('Target Count')
plt.subplots_adjust(right=1.9)

# Pie Chart
plt.subplot(1, 2, 2)
values = [df_train[df_train['target']==0].shape[0], df_train[df_train['target']==1].shape[0]]
labels = ['Sincere questions', 'Insincere questions']

plt.pie(values, labels=labels, autopct='%1.1f%%', shadow=True)
plt.title('Target Distribution')
plt.tight_layout()
plt.subplots_adjust(right=1.9)
plt.show()
```

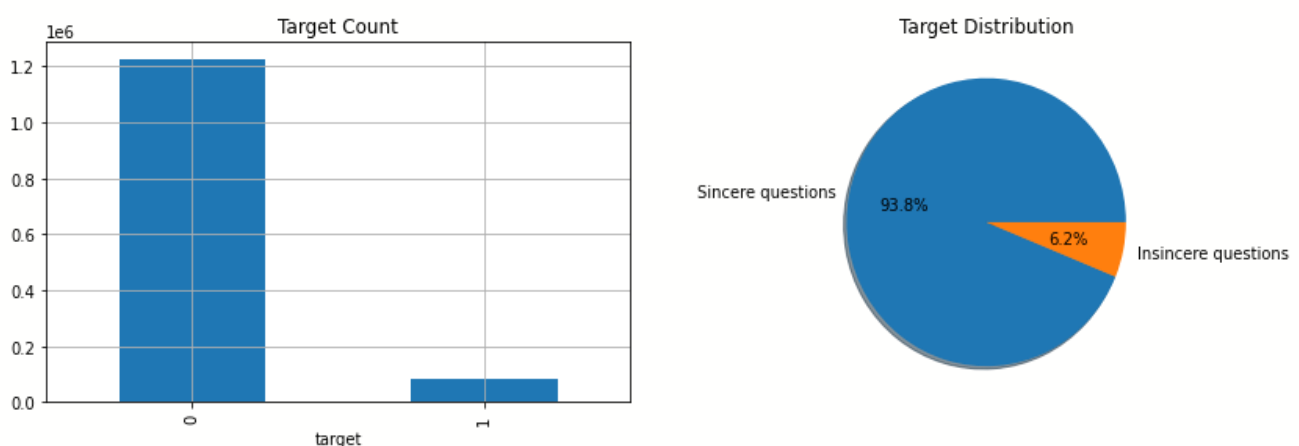


Fig (2) Target Count & Distribution

I plotted a Bar Chart and Pie Chart using Matplotlib Library

Observations:

- Data is highly imbalanced with only 6.2% of insincere questions.

2. Word cloud for both sincere and insincere questions

```
def display_wordcloud(data, title):  
    words_list = data.unique().tolist()  
    words = ' '.join(words_list)  
  
    wordcloud = WordCloud(width = 800, height = 400,  
                           stopwords = set(STOPWORDS)).generate(words)  
  
    plt.figure(figsize=(20, 12), facecolor=None)  
    plt.imshow(wordcloud)  
    plt.title(f'Words in {title}')  
    plt.axis("off")  
    plt.show()
```

Using WordCloud and Matplotlib and removing STOPWORDS visualize this plot.

Words in Sincere Questions

Wordcloud for Sincere Questions

```
display_wordcloud(df_train[df_train['target']==0]['question_text'], 'Sincere Questions')
```

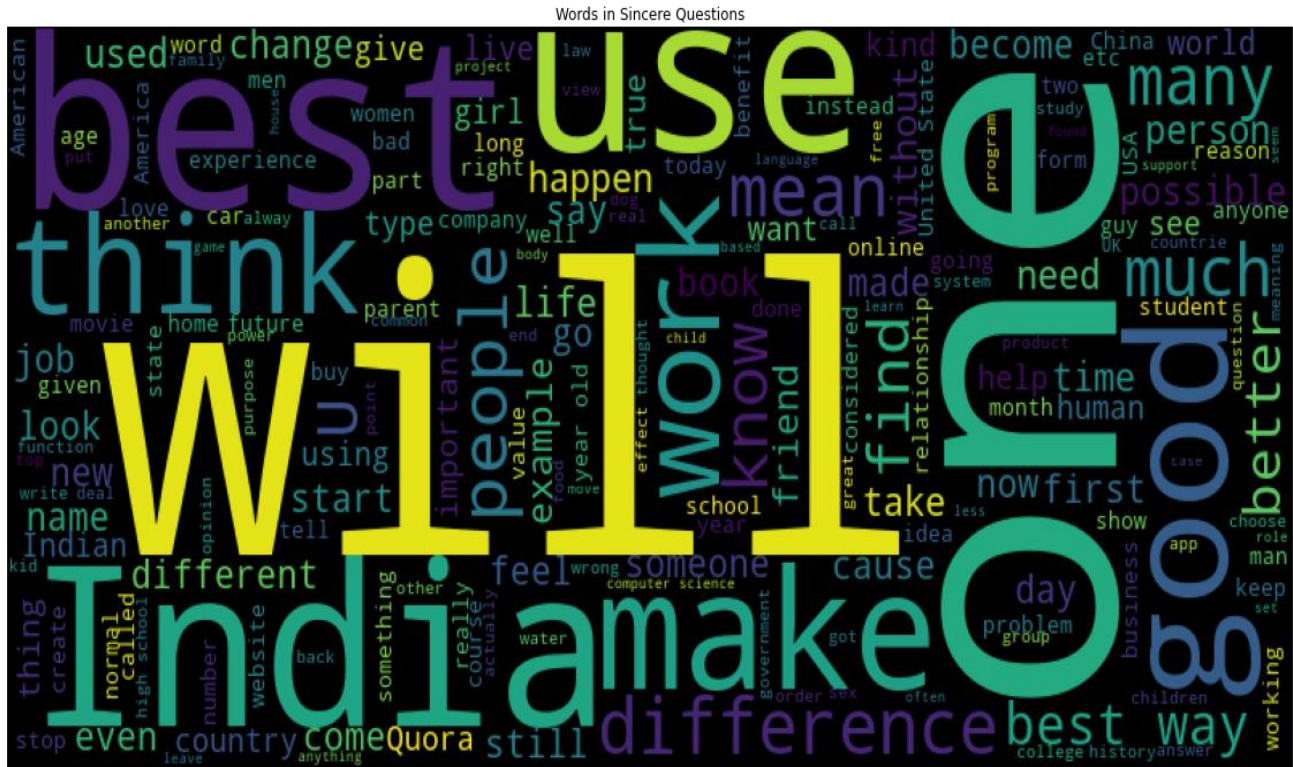


Fig (3) WordCloud of Sincere Questions

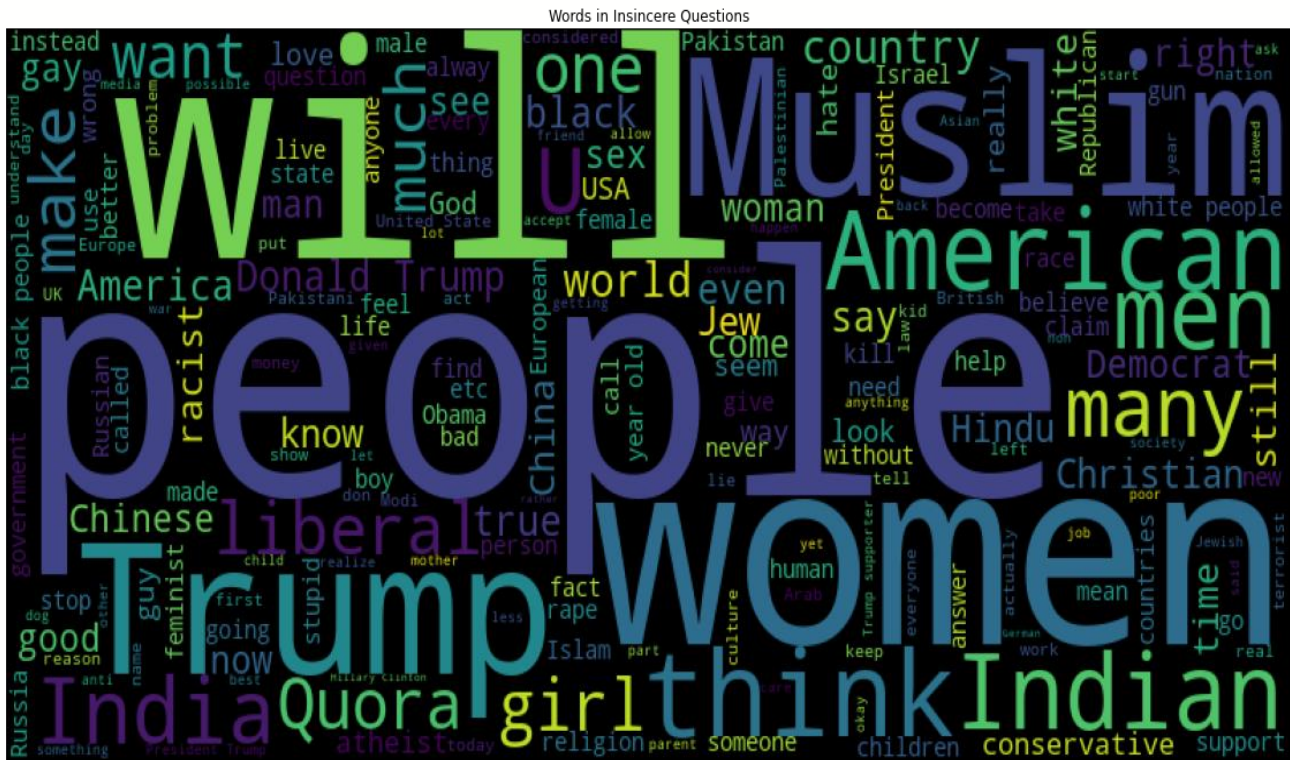
Observations:

As we see Sincere Questions words hasn't contain any abessive words.

There are **educational words** are more as compare other.

Words in Insincere Questions

```
# Wordcloud for Insincere Questions
display_wordcloud(df_train[df_train['target']==1]['question_text'], 'Insincere Questions')
```



Fig(4): WordCloud of Insincere Questions

Observations:

- As we can see insincere questions contain many of the offensive words.
- Most of the questions are related to People, Muslim, Women, Trump, etc.

3. Top words in both sincere and insincere questions.

```
from collections import Counter

def plot_word_freq(data, title, bar_color):

    top_words = Counter(data).most_common(25) #top 25 words

    df_top = pd.DataFrame(top_words, columns=['word', 'count']).sort_values('count') # storing in df

    plt.barh(df_top['word'].values, df_top['count'].values, orientation='horizontal', color=bar_color) # plot
    plt.title(f'Top words in {title}')

def get_unigrams(data):
    unigrams = []
    for sent in data:
        unigrams.extend([w for w in sent.lower().split() if w not in STOPWORDS])
    return unigrams

# Unigrams
unigrams_sincere = get_unigrams(df_train[df_train['target']==0]['question_text'])
unigrams_insincere = get_unigrams(df_train[df_train['target']==1]['question_text'])
```

Importing Counter from collections to get collections of 25 unigrams.

Filters. (linguistics) An n-gram consisting of a single item from a sequence.

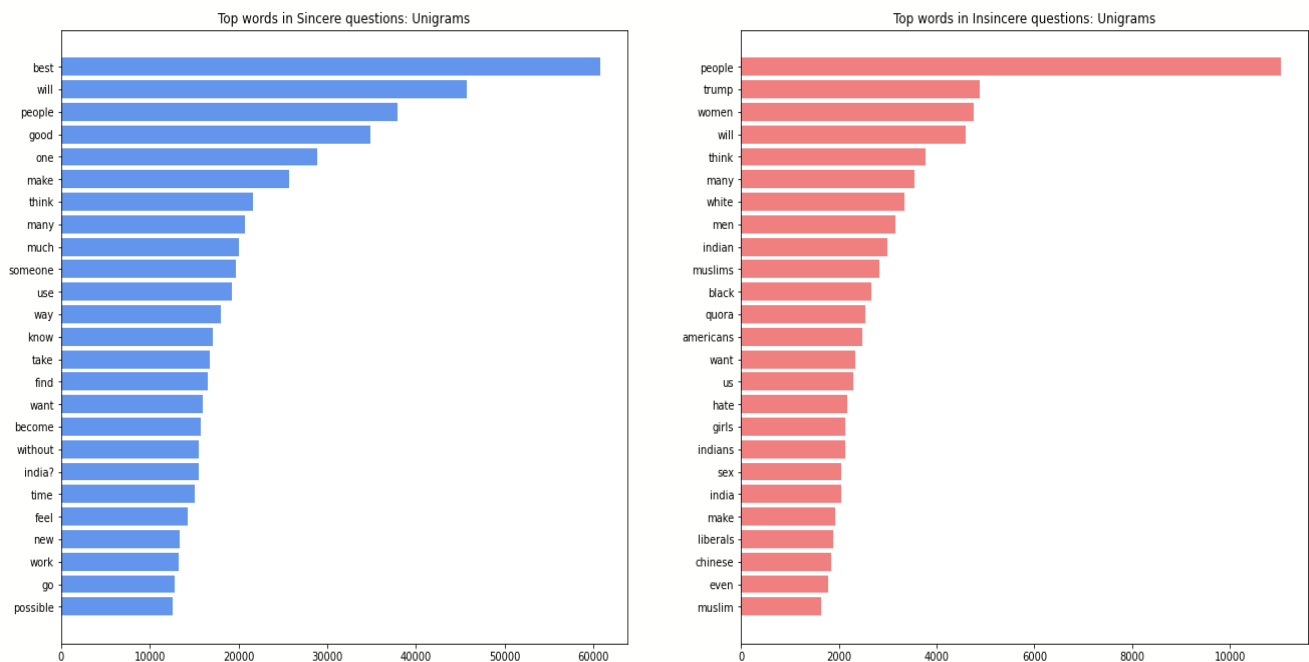
Unigram – Take single word from sequence

Bigram – Take Two Consistent words from sequence.

Trigram – Take Two Consistent words from sequence.

```
# Unigrams Sincere words
plt.subplot(1, 2, 1)
plot_word_freq(unigrams_sincere, 'Sincere questions: Unigrams', 'cornflowerblue')

# Unigrams Insincere words
plt.subplot(1, 2, 2)
plot_word_freq(unigrams_insincere, 'Insincere questions: Unigrams', 'lightcoral')
plt.subplots_adjust(right=3.0)
plt.subplots_adjust(top=2.0)
plt.show()
```



Fig(5): Top Words in Sincere & Insincere questions

Observations:

In Sincere words Best, people, good, such like these words has more distribution.

In Insincere words people, trump, women, etc. words has more distribution.

4. Basic Feature Extraction

Let's construct few basic features like:

- Number of words
- Number of capital_letters
- Number of special characters
- Number of unique words
- Number of numerics
- Number of characters
- Number of stopwords

```
# Number of words
df_train['num_words'] = df_train['question_text'].apply(lambda x: len(str(x).split()))
df_test['num_words'] = df_test['question_text'].apply(lambda x: len(str(x).split()))

# Number of capital_letters
df_train['num_capital_let'] = df_train['question_text'].apply(lambda x: len([c for c in str(x) if c.isupper()]))
df_test['num_capital_let'] = df_test['question_text'].apply(lambda x: len([c for c in str(x) if c.isupper()]))

# Number of special characters
df_train['num_special_char'] = df_train['question_text'].str.findall(r'^a-zA-Z0-9 ').str.len()
df_test['num_special_char'] = df_test['question_text'].str.findall(r'^a-zA-Z0-9 ').str.len()

# Number of unique words
df_train['num_unique_words'] = df_train['question_text'].apply(lambda x: len(set(str(x).split())))
df_test['num_unique_words'] = df_test['question_text'].apply(lambda x: len(set(str(x).split())))

# Number of numerics
df_train['num_numerics'] = df_train['question_text'].apply(lambda x: sum(c.isdigit() for c in x))
df_test['num_numerics'] = df_test['question_text'].apply(lambda x: sum(c.isdigit() for c in x))

# Number of characters
df_train['num_char'] = df_train['question_text'].apply(lambda x: len(str(x)))
df_test['num_char'] = df_test['question_text'].apply(lambda x: len(str(x)))

# Number of stopwords
df_train['num_stopwords'] = df_train['question_text'].apply(lambda x: len([c for c in str(x).lower().split() if c in STOPWORDS]))
df_test['num_stopwords'] = df_test['question_text'].apply(lambda x: len([c for c in str(x).lower().split() if c in STOPWORDS]))

df_train.head()
```

I used Regular Expression for Special Characters.

	qid	question_text	target	num_words	num_capital_let	num_special_char	num_unique_words	num_numerics	num_char	num_stopwords
0	00002165364db923c7e6	How did Quebec nationalists see their province...	0	13	2	1	13	4	72	7
1	000032939017120e6e44	Do you have an adopted dog, how would you enco...	0	16	1	2	15	0	81	10
2	0000412ca6e4628ce2cf	Why does velocity affect time? Does velocity a...	0	10	2	2	8	0	67	3
3	000042bf85aa498cd78e	How did Otto von Guericke used the Magdeburg h...	0	9	4	1	9	0	57	3
4	0000455dfa3e01eae3af	Can I convert montra helicon D to a mountain b...	0	15	3	1	15	0	77	7

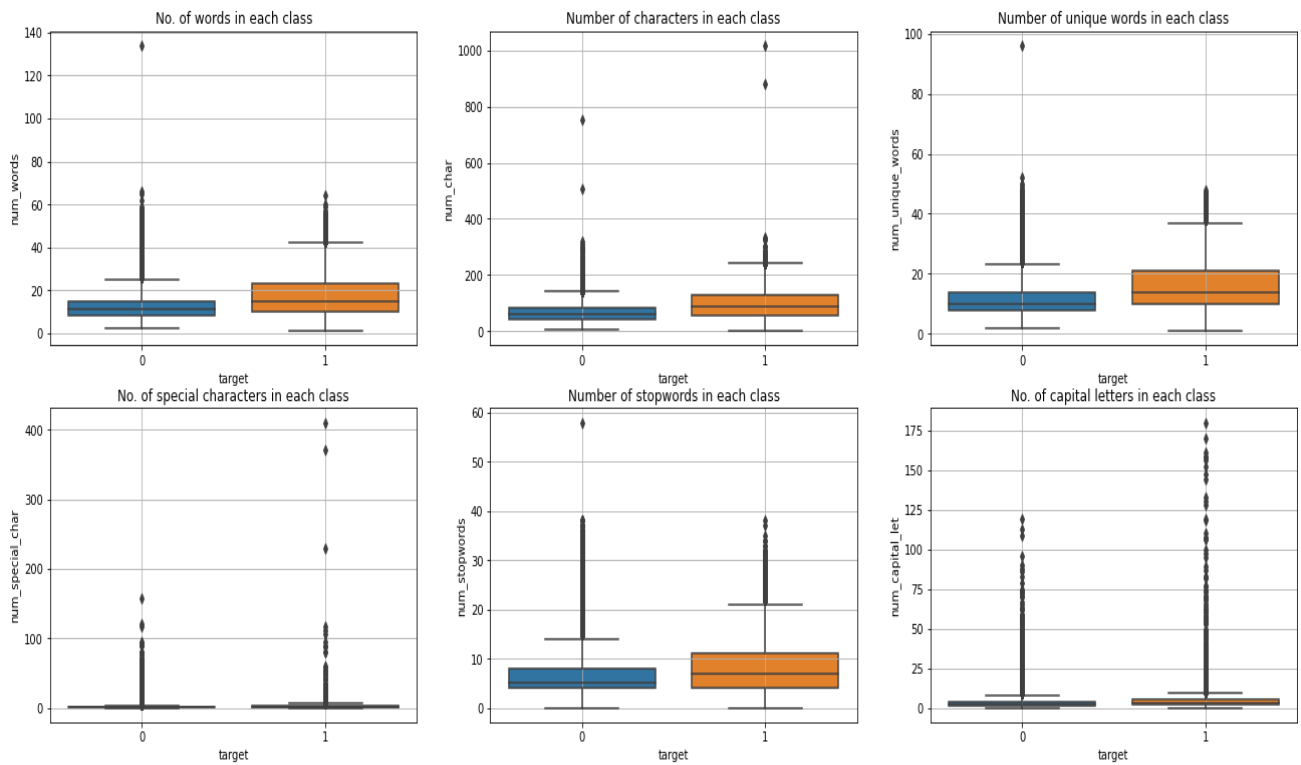
5. Analysis on extracted features.

```
print("Minimum length of a question:", min(df_train['num_words']))  
print("Maximum length of a question:", max(df_train['num_words']))
```

```
Minimum length of a question: 1  
Maximum length of a question: 134
```

```
def display_boxplot(_x, _y, _data, _title):  
    sns.boxplot(x=_x, y=_y, data=_data)  
    plt.grid(True)  
    plt.title(_title)
```

```
# Boxplot: Number of words  
plt.subplot(2, 3, 1)  
display_boxplot('target', 'num_words', df_train, 'No. of words in each class')  
  
# Boxplot: Number of chars  
plt.subplot(2, 3, 2)  
display_boxplot('target', 'num_char', df_train, 'Number of characters in each class')  
  
# Boxplot: Number of unique words  
plt.subplot(2, 3, 3)  
display_boxplot('target', 'num_unique_words', df_train, 'Number of unique words in each class')  
  
# Boxplot: Number of special characters  
plt.subplot(2, 3, 4)  
display_boxplot('target', 'num_special_char', df_train, 'No. of special characters in each class')  
  
# Boxplot: Number of stopwords  
plt.subplot(2, 3, 5)  
display_boxplot('target', 'num_stopwords', df_train, 'Number of stopwords in each class')  
  
# Boxplot: Number of capital letters  
plt.subplot(2, 3, 6)  
display_boxplot('target', 'num_capital_let', df_train, 'No. of capital letters in each class')  
  
plt.subplots_adjust(right=3.0)  
plt.subplots_adjust(top=2.0)  
plt.show()
```



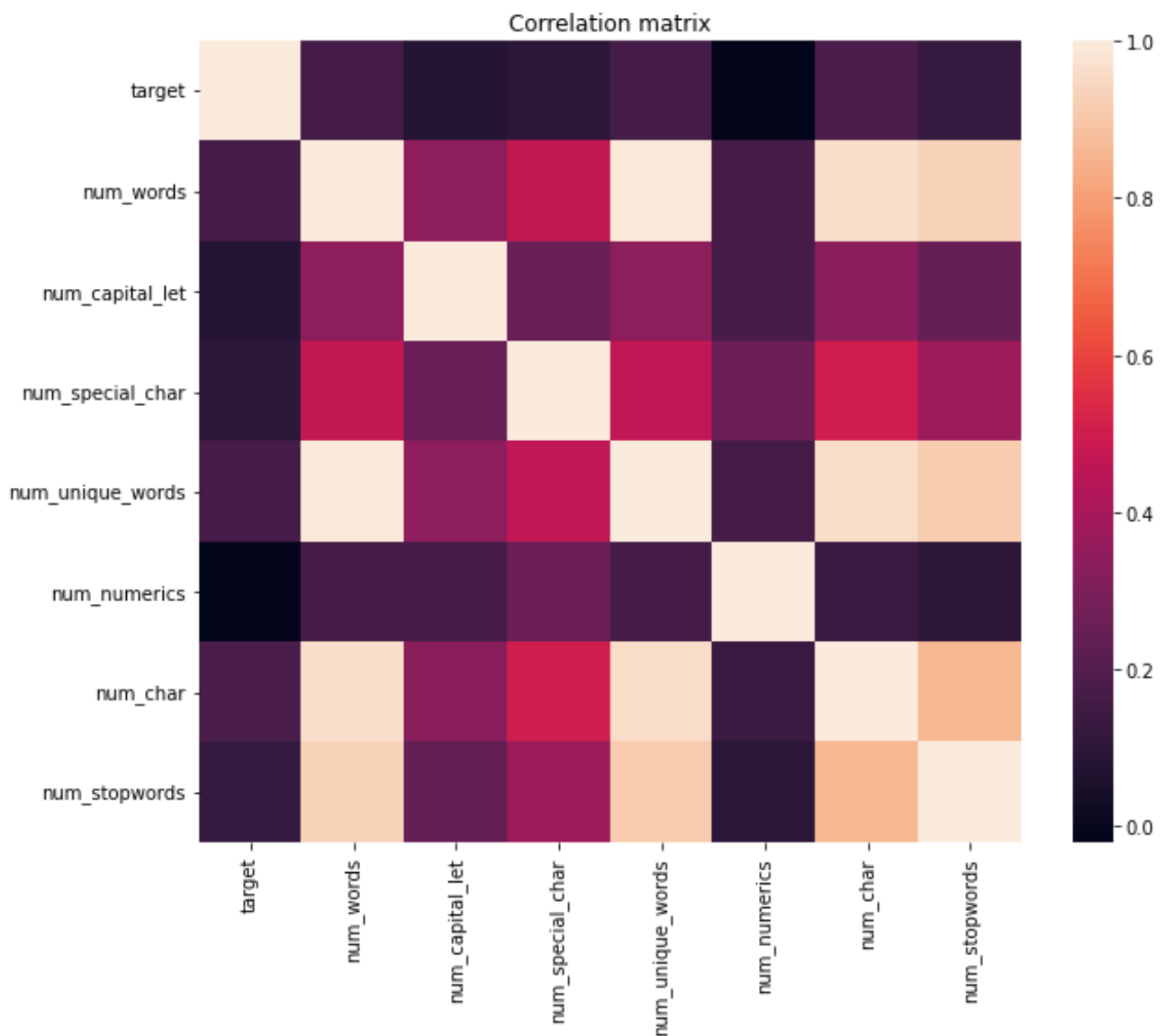
Fig(6): Analysis of Extracted features

Observations:

- Insecure questions seems to have more words and characters.
- Insecure questions also have more unique words compare to sincere questions.

6. Correlation Matrix

```
# Correlation matrix
f, ax = plt.subplots(figsize=(10, 8))
corr = df_train.corr()
sns.heatmap(corr, ax=ax)
plt.title("Correlation matrix")
plt.show()
```



Fig(7) Correlation Matrix

7. Question with the greatest number of Non- Numeric Characters.

```
# Questions with most number of non-alphanumeric characters.
# Insincere questions are printed in red color.
```

```
qids = df_train.sort_values('num_special_char', ascending=False)['qid'].head(20).values
for id in qids:
    row = df_train[df_train['qid'] == id]
    if row['target'].values[0] == 1:
        color = '\033[31m'
    else:
        color = '\033[0m'
    print(color, row['question_text'].values[0], '\n')
```

What is $\frac{\int_{-5}^5 \tan(\tan(\boxed{1x^0^{1x^2}} \sum_{\varpi=1}^\infty \int_{-3}^{-2x^2} \sum_{\alpha=7}^\infty \underbrace{\sqrt[2]{1x^5}}_{\text{Read carefully.}} - 3x^{-1} \div 1x^5 + \sqrt[3]{2x^{-3}})^{1x^0}) + \vec{\nu}}$

What is $\overbrace{\sum_{\vartheta=8}^\infty \vec{\nu} \circ \sum_{\kappa=7}^\infty \overbrace{1x^0}^{\text{Read carefully.}}} - 3x^{-1} \div 1x^5 + \sqrt[3]{2x^{-3}})^{1x^0} + \vec{\nu}$

What is the answer to $\frac{422}{2262} \mid 5501 \int_{846}^{} \frac{\Omega}{256} \int_{2551}^{5942} \wedge^{4366+1736} \frac{993+548}{1491} \mid 8813 \int (\int_{}^{} 6x + 35) ^{8613-4985} \frac{re}{fr}$

Do You think the republic of China still exist? If yes, Where do You think the capital of the republic of China? Nanjing or Taipei? 中国大陆的同胞们，此问题只是针对外国友人的一次统计，并无

What does this mean? "あからさまに機嫌悪いオーラ出してるのに話しかけてくるし、からかってくる男に、今そう言う冗談で笑える気分じゃないから一人にしてって言ったら何があったの？話聞くよって言われたんだけど"

$\frac{\text{x}\{\text{d}\}}{\text{x}\{\text{t}\}} = ax$, $\text{x}^{-1} = A \text{t}$ [math], [math]\{\text{now taking transpose on both sides}\} [math] \int_C x \text{x}^{-T} \{\text{x}\} = i

How do you show that $\sum_{m \geq 3} \left(4 \binom{1}{2} m + \binom{1}{2} m - 1 \right) \frac{1}{\Gamma(m - \frac{1}{2})} \lim_{\epsilon \rightarrow \infty} \int_{-\epsilon}^{\epsilon} x^{m-3/2} dx$

Which of the options listed below would most accurately fit $(f \circ g)(x), f(x) = \frac{1}{x+4}, g(x) = \frac{7}{x}$? -A. $(f \circ g)(x) = \frac{7x+28}{x}$ -B. $(f \circ g)(x) = \frac{7}{x+4}$

If $f_1 = f_2 = \dots = f_n = f_{n-1} + f_{n-2}$ \$ how I prove that $\sum_{k=1}^n f_k = \sum_{k=1}^n f_n \cdot \frac{1}{f_{n-1}}$

```
def get_all_symbols(data):  
    """  
  
    Returns SET: special symbols in corpus  
    """  
  
    symbols = []  
    for text in data:  
        s = re.findall(r'\W+', str(text))  
        symbols.extend(s)  
  
    return set(symbols)
```

```
extracted_symbols = get_all_symbols(df_train['question_text'].values)  
print(extracted_symbols)
```

```
{':', '"', '$', '/', '(', ')', '*', '+', '!', '}', '?', '=', '-', '\\\\', '^"', '?', '  
^', ')))', '.', 'p?', ' ', "--", "'", ')?"?', ',', '+', '+', '+', '+', '$', ',', '"', ':', '\\\\:', ']', ' ', '= {{ {{ {{, " '&?'", '\\\\'?", '?...', '-/-?', '<
```

Observations:

- Looks like there are some math questions (most of them are classified as insincere) in the data which contains more special chars and numbers.
- Some questions also contain emojis and non-English characters.
- Presence of punctuations in may add more value to ML models.

3.Data Preprocessing & Cleaning

1.Handle Imbalance data

To handle imbalance text data, we have to increase the size of the minority data. Following are the methods,

1. Text Augmentation: Here we generate duplicate texts. Using Pyaug library.
2. Resampling: There are two methods Under sampling and Over Sampling.
3. Copy minority data: Copy Minority data for how many times we want.

>In my research I firstly used **Copy Minority data** method to increase the size of minority of data. And I haven't use Text Augmentation, because it needs to High GPU Ram. So it is taking so much time to augment texts.

>Minority data size is **80810**.

>I copied this data 7 times and the overall size I got is **565670**.

>**df_os** is the new minority data.

```
df_os = pd.read_csv('/content/sample.csv', encoding='ISO-8859-1')
```

```
df_os.head()
```

	id	question_text	target
0	0000e91571b60c2fb487	Has the United States become the largest dicta...	1
1	00013ceca3f624b09f42	Which babies are more sweeter to their parents...	1
2	0004a7fcb2bf73076489	If blacks support school choice and mandatory ...	1
3	00052793eaa287aff1e1	I am gay boy and I love my cousin (boy). He is...	1
4	000537213b01fd77b58a	Which races have the smallest penis?	1

>We append this data in df_train data.

```
df_app = df_train.append(df_os)
```

>Now the dataset size has become $1306122 + 565670 = 1759540$

```
df_app.shape
```

```
(1759540, 4)
```

>Now the data looks balance dataset.

```
df_app['target'].value_counts()
```

```
0    1225312
1     534228
Name: target, dtype: int64
```

For model building I will use this append df_app dataset.

2)Lower Casing

```
df_train["question_text"] = df_train["question_text"].str.lower()
df_app["question_text"] = df_app["question_text"].str.lower()
df_app.head()
```

	qid	question_text	target
0	00002165364db923c7e6	how did quebec nationalists see their province...	0
1	000032939017120e6e44	do you have an adopted dog, how would you enco...	0
2	0000412ca6e4628ce2cf	why does velocity affect time? does velocity a...	0
3	000042bf85aa498cd78e	how did otto von guericke used the magdeburg h...	0
4	0000455dfa3e01eae3af	can i convert montra helicon d to a mountain b...	0

Whole Sentence will be in small letter.

3)Contractions

```
! pip install contractions
import contractions
df_train["question_text"] = df_train["question_text"].apply(lambda x: [contractions.fix(word) for word in x.split()])
df_app["question_text"] = df_app["question_text"].apply(lambda x: [contractions.fix(word) for word in x.split()])
df_train.head()
```

```
df_train['question_text'] = [' '.join(map(str, l)) for l in df_train['question_text']]
df_app['question_text'] = [' '.join(map(str, l)) for l in df_app['question_text']]
df_train.head()
```

	qid	question_text	target
0	00002165364db923c7e6	how did quebec nationalists see their province...	0
1	000032939017120e6e44	do you have an adopted dog, how would you enco...	0
2	0000412ca6e4628ce2cf	why does velocity affect time? does velocity a...	0
3	000042bf85aa498cd78e	how did otto von guericke used the magdeburg h...	0
4	0000455dfa3e01eae3af	can i convert montra helicon d to a mountain b...	0

There is one library called as contractions, we need to install and import that. This will remove contractions

Example;

Aren't = Are Not,

Don't = Do not

Can't = Can not

Like this model will read these words properly.

4)Removal of StopWords

```

nltk.download('stopwords')
from nltk.corpus import stopwords
", ".join(stopwords.words('english'))

```

```

STOPWORDS = set(stopwords.words('english'))
def remove_stopwords(text):
    """custom function to remove the stopwords"""
    return " ".join([word for word in str(text).split() if word not in STOPWORDS])

df_train["question_text"] = df_train["question_text"].apply(lambda text: remove_stopwords(text))
df_app["question_text"] = df_app["question_text"].apply(lambda text: remove_stopwords(text))
df_train.head()

```

	qid	question_text	target
0	00002165364db923c7e6	quebec nationalists see province nation 1960s?	0
1	000032939017120e6e44	adopted dog, would encourage people adopt shop?	0
2	0000412ca6e4628ce2cf	velocity affect time? velocity affect space ge...	0
3	000042bf85aa498cd78e	otto von guericke used magdeburg hemispheres?	0
4	0000455dfa3e01eae3af	convert montra helicon mountain bike changing ...	0

These are some examples of StopWords.

[i, me, my, myself, we, our, ours, ourselves, you, you're, you've, you'll, you'd, your, yours, yourself, yourselves, he, him, his, himself, she, she's, her, hers, herself, it, it's, its, itself, they, them, their, theirs, themselves, what, which, who, whom, this, that, that'll, these, those, am, is, are, was, were, be, been, being, have, has, had, having, do, does, did, doing, a, an, the, and, but, if].

And we removed these StopWords from the both datasets.

5)Remove Integers from Texts

```
df_train['question_text'] = df_train['question_text'].str.replace('\d+', '')
df_app['question_text'] = df_app['question_text'].str.replace('\d+', '')
df_train.head()
```

In the questions we analyse the keywords or particular text so we don't need any integer in the data.

6)Stemming

The process of developing morphological variants of a root/base word is known as stemming. Stemming algorithms or stemmers are terms used to describe stemming programmes. The phrases "chocolates," "chocolatey," and "choco" are reduced to the root word "chocolate," and "retrieval," "retrieved," and "retrieves" are reduced to the stem "retrieve."

In natural language processing, stemming is an integral aspect of the pipelining process. Tokenized words are fed into the stemmer.

```
stemmer = PorterStemmer()
def stem_words(text):
    return " ".join([stemmer.stem(word) for word in text.split()])

df_train["question_text"] = df_train["question_text"].apply(lambda text: stem_words(text))
df_app["question_text"] = df_app["question_text"].apply(lambda text: stem_words(text))
df_train.head()
```

There are 2 libraries to do stemming PorterStemmer, LancasterStemmer.

After this,

7)Lemmatization

Lemmatization is the process of combining a word's several inflected forms into a single item that may be studied. Lemmatization is similar to stemming, but it gives the words context. As a result, it connects words with similar meanings into a single term.

Stemming and lemmatization are both part of text preprocessing. Many individuals are confused by these two terms. Some people confuse these two. In fact, lemmatization is preferable than stemming since it does morphological analysis on the words.

Examples of lemmatization:

- > rocks : rock
- > corpora : corpus

-> better : good

```
nltk.download('wordnet')
from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()
def lemmatize_words(text):
    return " ".join([lemmatizer.lemmatize(word) for word in text.split()])

df_train["question_text"] = df_train["question_text"].apply(lambda text: lemmatize_words(text))
df_app["question_text"] = df_app["question_text"].apply(lambda text: lemmatize_words(text))
df_train.head()
```

8)Remove URL from the texts

```
df_train["question_text"] = df_train["question_text"].str.replace(r's*https?://S+(s+|$)', ' ').str.strip()
df_app["question_text"] = df_app["question_text"].str.replace(r's*https?://S+(s+|$)', ' ').str.strip()
```

Some of the questions contain URL also so we remove them.

9)Remove HTML Tags

```
def remove_tags(string):
    result = re.sub('<.*?>', '', string)
    return result

df_train['preprocessed_question_text'] = df_train['question_text'].apply(lambda cw : remove_tags(cw))
df_app['preprocessed_question_text'] = df_app['question_text'].apply(lambda cw : remove_tags(cw))
df_train.head()
```

Also, we have to remove HTML Tags.

Now, the data cleaned.

And also I create new variable **preprocessed_question_text** and I will use this variable for model building.

4.Modeling

1.Import dependencies

```
#Libraries used for modeling
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import Pipeline
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
from mlxtend.plotting import plot_confusion_matrix
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn import metrics
```

Fig(8): Dependencies

Train_test_split – To split data into train and test

CountVectorizer & TfidfTransformer – To extract features into integer.

Pipeline – To do Vecrorizer, algorithm and all in one.

Logistic Regression, Naïve Bayes - Algorithm

Model Evaluation Libraries – Metrics, confusion_metrics,
plot_confusion_metrics

Bias & Variance trade -off

- 1) Under Fit – We have High Bias and High Variance.
- 2) Over Fit – We have Low Bias and High Variance.
- 3) Best Fit – We have Low Bias and High Variance.

2.First of all we build model on imbalance data and check the accuracy.

These two models we will use

- 1)Multinomial Naïve Bayes
- 2)Logistic Regression

3.Cross Validation

Train- Test Split

We split the dataset into training and test data. The training set contains a known output and is used for learning. The test dataset is used to test the model's prediction. We do this using the method `train_test_split` provided by the Scikit-learn library. We split the data set in 75:25 ratio, 75% for the training and 25% for testing. Also, the `random_state` is **2000**.

Then Initialized X as `preprocessed_question_text` and Y as target independent variable.

```
X = df_train['preprocessed_question_text']  
Y = df_train['target']
```



```
#Shuffling the data and splitting the data into train and test sets (75:25)
```

```
X_train, X_test, Y_train, Y_test= train_test_split(X, Y, test_size=0.25, random_state = 2000, shuffle= True)
```

4.Feature Extraction Methods:

CountVectorizer:

CountVectorizer is a Python tool supplied by the sklearn or scikit-learn package that provides application programming interfaces (APIs) to assist in the construction of a BoW model.

It turns a collection of text documents into a matrix, with each entry corresponding to the count of a specific token in the corresponding sentences.

CountVectorizer provides a lot of versatility when it comes to using a prebuilt dictionary of words rather than developing one from scratch. It also has tools for tokenizing text and eliminating stopwords. Using the CountVectorizer module's ngram range attribute, you may accomplish this without having to write any code.

It is used to convert a text into a vector based on the frequency (count) of each word in the text. This is useful when dealing with a large number of such texts and converting each word into a vector (for using in further text analysis).

The BoW model gives a method for numerically encoding text data. It does, however, have certain restrictions. The approach is based solely on the number of phrases in a document. This might work for some activities or use cases with a restricted vocabulary, but it wouldn't scale well to huge vocabularies.

The BoW approach also includes options for removing or lowering the relevance of tokens or words that appear infrequently. These sentences may appear in a small number of papers, but they can have a significant impact

on how those documents are represented. Such alternatives are not supported by the BoW model.

For a big text corpus, the BoW model can get exceedingly enormous in terms of vocabulary. These models do not take into account the semantics or meanings of a token or words in a document. It overlooks the possibility of capturing elements in the vicinity of a phrase that can provide insight into the context in which a word or phrase is utilised. As a result, it completely disregards the context.

TF-IDF:

The TF-IDF method uses a weight scheme that is based on the frequency of terms in a text corpus.

The TF-IDF method is by far the most popular method for weighing words. It can be found in a variety of applications, including search engines, information retrieval, and text mining systems. The TF-IDF method for vectorizing text and extracting features is likewise based on occurrences. It is made up of two terms that are defined as follows:

$$TF(w) = \frac{\text{Number of times the word } w \text{ occurs in a document}}{\text{Total number of words in the document}}$$

$$IDF(w) = \log \frac{\text{Total number of documents}}{\text{Number of documents containing word } w}$$

$$weight(w, d) = TF(w, d) \times IDF(w)$$

The weight of word w in document d is a product of the TF of word w in document d and the IDF of word w across the text corpus

"We are reading about Natural Language Processing Here".

"Natural Language Processing making computers comprehend language data".

"The field of Natural Language Processing is evolving everyday"

```
['comprehend', 'computers', 'data', 'everyday', 'evolve', 'field',  
'language', 'make', 'natural', 'process', 'read']
```

```
[[0. 0. 0. 0. 0. 0. 0. 0.41285857 0. 0.41285857 0.41285857 0.69903033]  
 [0.40512186 0.40512186 0.40512186 0. 0. 0. 0.478543 0.40512186 0.2392715  
 0.2392715 0. ] [0. 0. 0. 0.49711994 0.49711994 0.49711994 0.29360705 0.  
 0.29360705 0.29360705 0. ]]
```

5.Algorithms:

1.Multinomial Naïve Bayes

The Naive Bayes Classifier Algorithm is a collection of probabilistic algorithms based on Bayes' theorem and the "naive" assumption of conditional independence between each pair of features.

The Bayes theorem determines the probability $P(c|x)$, where c is the class of probable outcomes and x is the supplied case to be identified, which represents some specific characteristics.

$$P(x|c) * P(c) / P(c) = P(x|c) (x)$$

Natural language processing (NLP) challenges are where naive Bayes are most commonly utilised. Naive Bayes predicts a text's tag. They calculate each tag's probability for a given text and output the tag with the highest probability.

We classify whether the sentence "generally enjoyed the film" is a good or negative review.

$P(\text{positive} | \text{overall enjoyed the movie})$ — the probability that a sentence's tag is positive provided that the sentence is "overall loved the movie" — must be calculated.

$P(\text{negative} | \text{overall enjoyed the movie})$ — the chance that a sentence's tag is negative when the sentence's content is "overall liked the movie."

2.Logistic Regression:

This article covers the fundamentals of Logistic Regression and how to use it in Python. A supervised classification algorithm, logistic regression is. For a given collection of features (or inputs), X , the target variable (or output), y , can only take discrete values in a classification issue.

Logistic regression, contrary to popular assumption, is a regression model. The model creates a regression model to forecast the likelihood that a given data entry belongs to the "1" category. Logistic regression models the data using the sigmoid function, just like linear regression assumes that the data follows a linear distribution.

When a decision criterion is introduced, logistic regression transforms into a classification procedure. Setting the threshold value is a crucial part of logistic regression, and it is determined by the classification problem. The precision and recall levels have a significant influence on the threshold value determination. Both precision and recall should ideally be 1, but this is rarely the case.

The following arguments are used to determine the threshold in a Precision-Recall tradeoff: -

1. High recall/low precision:

We choose a decision value with a low Precision or a high Recall in applications where we wish to lower the number of false negatives without necessarily reducing the number of false positives. In a cancer diagnosis application, for example, we don't want any affected patients to be labelled as unaffected without considering whether or not they are being unjustly diagnosed with cancer. This is because the lack of cancer can be recognised by other medical conditions, but the existence of cancer cannot be detected in a candidate who has previously been rejected.

2. High Precision/Low Recall:

We choose a decision value with a high Precision or a low Recall in applications where we wish to lower the number of false positives without necessarily reducing the number of false negatives. For example, if we're predicting whether a consumer would react positively or negatively to a personalised advertisement, we want to be completely certain that the customer will respond positively because a negative reaction could result in a loss of prospective sales.

6.Evaluation Metrics:

Evaluation Metrics Accuracy: The ratio of correct predictions over the total predictions.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision: The number of true positives divided by all positive predictions. It is a measure of a classifier's exactness. It tells us how often the classifier is correct when it predicts positive.

Low precision means that there is a high number of false positives.

$$precision = \frac{TP}{TP + FP}$$

Recall: The number of true positives divided by the number of positive values in the test data. It is also known as Sensitivity or the True Positive Rate.

It is a measure of a classifier's completeness. It tells us how often the classifier is correct for all positive instances.

Low recall means that there is a high number of false negatives.

$$recall = \frac{TP}{TP + FN}$$

F1-Score: It is the harmonic mean of precision and recall.

$$F1\ score = 2 * \frac{precision * recall}{precision + recall}$$

Confusion Matrix: It is a table that is used to describe the performance of a classifier on the test data for which the true values are known

	<i>Predicted YES</i>	<i>Predicted NO</i>
<i>Actual YES</i>	True Positives (TP)	False Negatives (FN)
<i>Actual NO</i>	False Positives (FP)	True Negatives (TN)

ROC – AUC Curve:

The AUC - ROC curve is a performance metric for classification issues at different threshold levels. AUC represents the degree or measure of separability, whereas ROC is a probability curve. It indicates how well the model can distinguish between classes. The AUC indicates how well the model predicts 0 courses as 0 and 1 classes as 1. By analogy, the higher the AUC, the better the model distinguishes between people who have the condition and those who do not.

The ROC curve is plotted with TPR on the y-axis and FPR on the x-axis, with TPR on the y-axis and FPR on the x-axis.

6.Model Building

1.Multi-Nomial Naïve Bayes on imbalance dataset.

Here I used both CountVectorizer and TfidfTransformer at once.

```
[113] pipeline=Pipeline([('bow',CountVectorizer(dtype=np.float32,
                                                strip_accents='unicode',
                                                analyzer='word',
                                                token_pattern=r'\w{1,}',
                                                ngram_range=(1,3),
                                                min_df=3)), #Minimum Document Frequency
                        ('tfidf',TfidfTransformer()),
                        ('classifier',MultinomialNB())
                        ])
```

```
[114] pipeline.fit(X_train, Y_train)
      pred=pipeline.predict(X_test)
```

In Countvectorizer the datatype is always in float and min_df(Minimum Document Frequency) is 3.

Min_df means the words which occur less than or equal to 3 times in dataset which we will neglect.

Then we fit the **X_train** and **Y_train** and using predict function predict on basis of **X_test**.

```
print('Test Accuracy using MultiNomial Naive Bayes Classifier: ',accuracy_score(Y_test,pred))
```

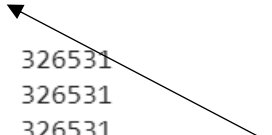
```
Test Accuracy using MultiNomial Naive Bayes Classifier: 0.9429211927810836
```

>Accuracy Score of the Multinomial Naïve Bayes model is **94.29%**.

>Classification Report

```
print(classification_report(Y_test,pred))
```

	precision	recall	f1-score	support
0	0.94	1.00	0.97	306417
1	0.75	0.11	0.19	20114
accuracy			0.94	326531
macro avg	0.85	0.55	0.58	326531
weighted avg	0.93	0.94	0.92	326531



Fig(9): Classification Report for Imbalance NB

Conclusion:

Here we can see f1- score and recall is low for “1” minority data.

Recall means True Positive Rate is not classifying very well, so whenever we will predict the model, the insincere question will not get correct prediction because of misclassification problem.

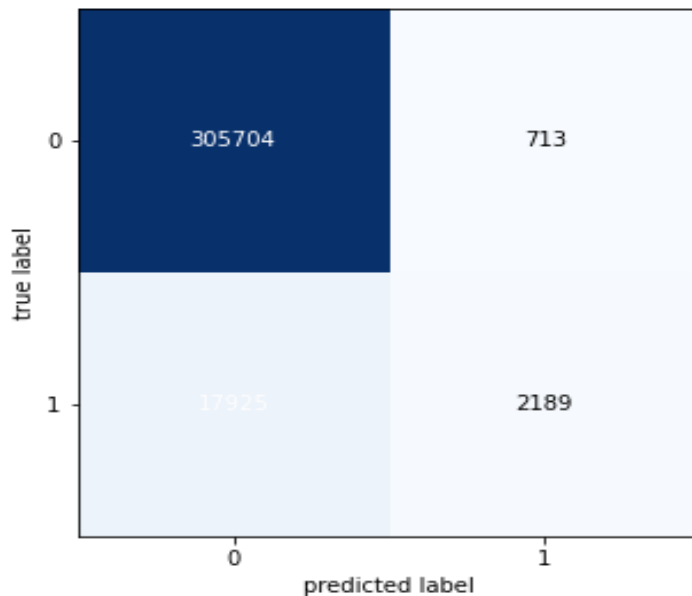
In this model Biasness towards Sincere question is high and therefore the model is underfitting and not giving correct predictions.

>Confusion Matrix

```
print(confusion_matrix(Y_test, pred))
cm_nb = confusion_matrix(Y_test, pred)

fig, ax = plot_confusion_matrix(conf_mat=cm_nb , figsize=(5, 5))
plt.show()
```

```
[[305704   713]
 [ 17925  2189]]
```

Fig(9): Confusion Matrix Imbalance NB

TP = 305704

FN = 713

FP = 17925

TN = 2189

Conclusion:

True Positive has high value therefore this will correctly predict and True Negative has low value then it will not correctly classify values.

It will misclassify the values.

>Correctly Classify:

```
print(f"Correctly classified sincere questions: {round(cm_nb[0][0]/(cm_nb[0][0] + cm_nb[0][1])*100, 2)}%")
print(f"Correctly classified insincere questions: {round(cm_nb[1][1]/(cm_nb[1][0] + cm_nb[1][1])*100, 2)}%")
```

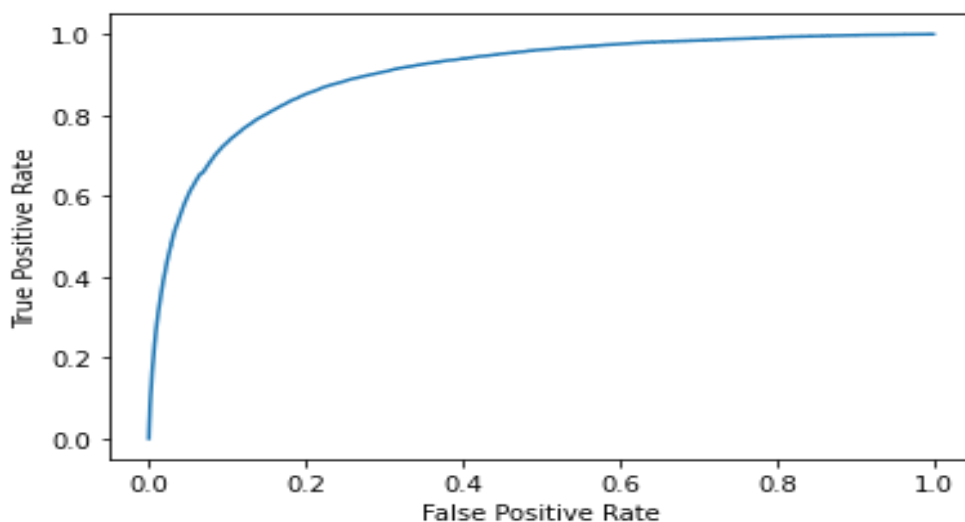
```
Correctly classified sincere questions: 99.77%
Correctly classified insincere questions: 10.88%
```

We can see the misclassification of Insincere questions on this.

>ROC – AUC Curve

```
y_pred_proba = pipeline.predict_proba(X_test)[::,1]  
fpr, tpr, _ = metrics.roc_curve(Y_test, y_pred_proba)
```

```
#create ROC curve  
plt.plot(fpr,tpr)  
plt.ylabel('True Positive Rate')  
plt.xlabel('False Positive Rate')  
plt.show()
```



Fig(11)ROC – AUC Curve Imbalance NB

Conclusion:

While the classifier has a high accuracy, it fails to classify even a single example from the minority class

2. Logistic Regression on Imbalance data.

Here I used both CountVectorizer and TfidfTransformer at once.

```
pipeline2 =Pipeline([('bow',CountVectorizer(dtype=np.float32,  
                                             strip_accents='unicode',  
                                             analyzer='word',  
                                             token_pattern=r'\w{1,}',  
                                             ngram_range=(1,3),  
                                             min_df=3)),  
                    ('tfidf',TfidfTransformer()),  
                    ('classifier',LogisticRegression())  
                    ])
```

```
pipeline2.fit(X_train, Y_train)  
pred2=pipeline2.predict(X_test)
```

In Countvectorizer the datatype is always in float and min_df(Minimum Document Frequency) is 3.

Min_df means the words which occur less than or equal to 3 times in dataset which we will neglect.

Then we fit the **X_train** and **Y_train** and using predict function predict on basis of **X_test**.

```
print('Test Accuracy using Logistics Regression Classifier: ',accuracy_score(Y_test,pred2))  
  
Test Accuracy using Logistics Regression Classifier:  0.9512603703783101
```

Accuracy Score on Logistic Regression Classifier model is **95.12%**.

>Classification Report

```
print(classification_report(Y_test, pred2))
```

	precision	recall	f1-score	support
0	0.96	0.99	0.97	306417
1	0.68	0.40	0.50	20114
accuracy			0.95	326531
macro avg	0.82	0.69	0.74	326531
weighted avg	0.94	0.95	0.95	326531

Fig(12): Classification Report Imbalance LR

Conclusion:

Here is also the same f1- score and recall is low means 50% and this is not enough to correctly classify for “1” minority data.

Recall means True Positive Rate is not classifying very well, so whenever we will predict the model, the insincere question will not get correct prediction because of misclassification problem.

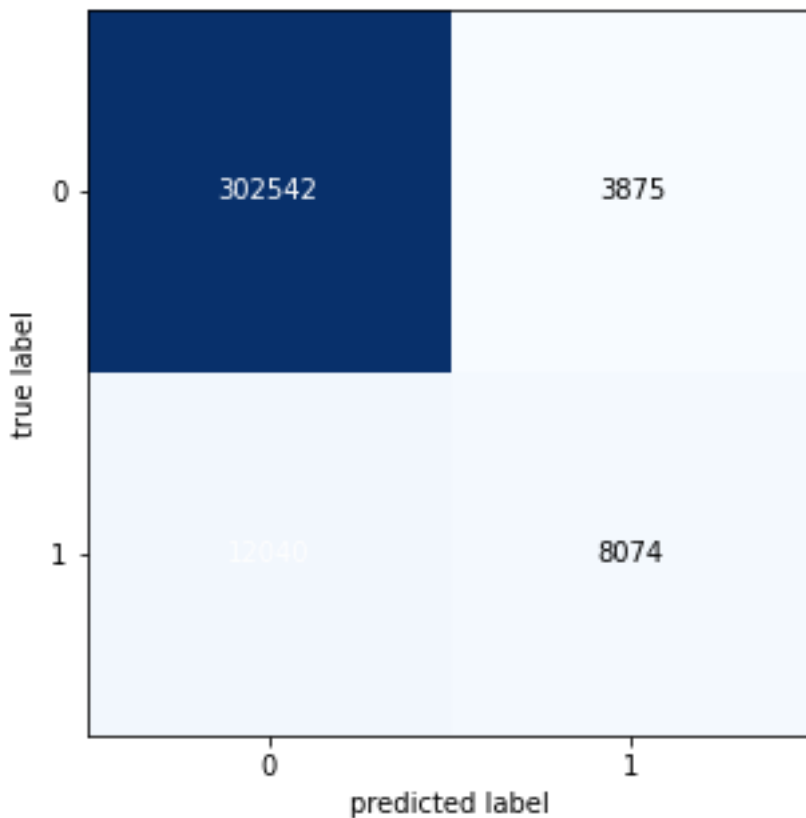
In this model Biasness towards Sincere question is high and therefore the model is underfitting and not giving correct predictions.

>Confusion Matrix

```
print(confusion_matrix(Y_test, pred2))
cm_lr = confusion_matrix(Y_test, pred2)

fig, ax = plot_confusion_matrix(conf_mat=cm_lr ,  figsize=(5, 5))
plt.show()
```

```
[[ 302542   3875]
 [ 12040   8074]]
```



Fig(13): Confusion Matrix Imbalance LR

TP = 302542

FN = 3875

FP = 12040

TN = 8074

Conclusion:

True Positive has high value therefore this will correctly predict and True Negative has low value then it will not correctly classify values.

It will misclassify the values.

>Correctly Classify

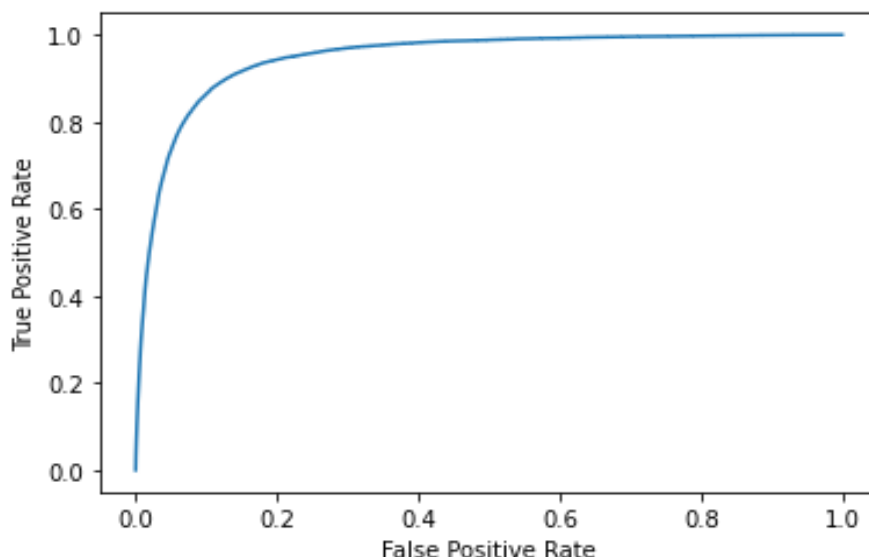
```
print(f"Correctly classified sincere questions: {round(cm_lr[0][0]/(cm_lr[0][0] + cm_lr[0][1])*100, 2)}%")  
print(f"Correctly classified insincere questions: {round(cm_lr[1][1]/(cm_lr[1][0] + cm_lr[1][1])*100, 2)}%")
```

Correctly classified sincere questions: 98.74%
Correctly classified insincere questions: 40.14%

We can see the misclassification of Insincere questions on this

>ROC – AUC Curve

```
y_pred_proba2 = pipeline2.predict_proba(X_test)[::,1]  
fpr, tpr, _ = metrics.roc_curve(Y_test, y_pred_proba2)  
  
#create ROC curve  
plt.plot(fpr,tpr)  
plt.ylabel('True Positive Rate')  
plt.xlabel('False Positive Rate')  
plt.show()
```



Conclusion:

Same as here also, While the classifier has a high accuracy, it fails to classify even a single example from the minority class.

Therefore, we can balance data and will see the accuracy.

To balance the data there are these two methods are also use but I haven't used.

Word Embedding:

Google's Word2vec is a sophisticated tool released in 2013^{5,24}. In high-dimensional vector space, it efficiently computes word vector representations.

Word vectors are mapped near each other in the vector space by words that have similar semantics and have common contexts. Similarity of word representations receive semantic features in addition to syntactic information, therefore semantic linkages are frequently preserved in vectors.

Vector of (King) + vector of (Woman) - vector of (Man),
for example, is similar to vector of (Queen).

In NLP applications such as text clustering and classification, as well as sentiment analysis, word vector representations have proven to be an effective and successful technique. The continuous bag-of-words (CBOW) and skip-gram neural network designs are used by Word2vec. These architectures have similar algorithms.

CBOW architecture, on the other hand, is taught to predict a word based on its context, whereas skip-gram predicts a word's context.

SMOTE:

SMOTE was utilised to solve the problem of an unbalanced dataset. SMOTE11 adds synthetic samples to the minority classes based on feature-space similarities between existing minority cases. It evaluates its k -nearest neighbours for each case x_i , where $x_i \in \text{min}$ (the minority class). SMOTE doubles the difference between the sample under consideration and its nearest neighbour by a random number between 0 and 1. The number of nearest neighbours and the type of estimators are two of the most essential SMOTE parameters. We employed the SVM estimator and five nearest neighbours in our trials.

7.Resampling

>Over sampling

```
# Class count
count_target_0, count_target_1 = df_train.target.value_counts()

# Divide by class
df_train_0 = df_train[df_train['target'] == 0]
df_train_1 = df_train[df_train['target'] == 1]

# Oversample 1-class and concat the DataFrames of both classes
df_train_1_over = df_train_1.sample(count_target_0, replace=True)
df_test_over = pd.concat([df_train_0, df_train_1_over], axis=0)

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

Random over-sampling:
0    1225312
1    1225312
Name: target, dtype: int64
```

Fig(14): Over Sampling

So, we over sample the minority data.

The minority data size has become equal to majority data.

>Dataset size

```
df_test_over.shape

(2450624, 11)
```

>Split the data into Train-test split

```
x = df_test_over['preprocessed_question_text']
y = df_test_over['target']

xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size=0.2, random_state=15)
```

>Here I used both CountVectorizer and TfidfTransformer at once.

```
pipeline3=Pipeline([('bow',CountVectorizer(dtype=np.float32,
                                             strip_accents='unicode',
                                             analyzer='word',
                                             token_pattern=r'\w{1,}',
                                             ngram_range=(1,3),
                                             min_df=3)),
                    ('tfidf',TfidfTransformer()),
                    ('classifier',LogisticRegression())
                    ])
```

Using Logistic Regression, I fit the model.

```
pipeline3.fit(xtrain, ytrain)
pred=pipeline3.predict(xtest)

print('Test Accuracy using Logistics Regression Classifier: ',accuracy_score(ytest,pred))

print(classification_report(ytest,pred))

print(confusion_matrix(ytest, pred))
CM = confusion_matrix(ytest, pred)

fig, ax = plot_confusion_matrix(conf_mat=CM ,  figsize=(5, 5))
plt.show()
```

Predict the model and then accuracy score, classification report and confusion matrix I plot.

Test Accuracy using Logistics Regression Classifier: 0.9644458046416731					
	precision	recall	f1-score	support	
0	1.00	0.93	0.96	245154	
1	0.94	1.00	0.97	244971	
accuracy			0.96	490125	
macro avg	0.97	0.96	0.96	490125	
weighted avg	0.97	0.96	0.96	490125	
[[228363 16791]					
[635 244336]]					

Fig(15) Classification Report LR

Here we can see for both classes model has over fitted because of high variance and recall is 100%.

So, for predictions we don't need this model.

2.Copy Minority data method:

We have already created the new data set which has contains balance classes.

Df_app data we will used for modeling and Predictions.

```
df_app['target'].value_counts()
```

```
0    1225312  
1     534228  
Name: target, dtype: int64
```

We can see the shape of the dataset is fairly balance to train the model.

1)Train – test Split

```
X = df_app['preprocessed_question_text']  
Y = df_app['target']
```

```
Xtrain, Xtest, Ytrain, Ytest= train_test_split(X, Y, test_size=0.25, random_state= 2000, shuffle= True)
```

2)Vecorization

There are two methods of **Vectorizerzation**

1) TfidfVectorizer

2) CountVectorizer

We will use only **TfidfVectorizer**.

```

from sklearn.feature_extraction.text import TfidfVectorizer

tfv = TfidfVectorizer(dtype=np.float32, min_df=3, max_features=None,
                      strip_accents='unicode', analyzer='word', token_pattern=r'\w{1,}',
                      ngram_range=(1, 3), use_idf=1, smooth_idf=1, sublinear_tf=1,
                      stop_words = 'english')
xv_train = tfv.fit_transform(Xtrain)
xv_test = tfv.transform(Xtest)

```

Fig(16): Vectorizer

Import TfidfVectorizer

Data type is in float

Minimum document frequency is 3

Use inverse document frequency is 1

Smooth inverse document frequency is 1

Stopwords is English.

Machine is not identifying the data in Text format so we have to transform them into numerical format.

That's why I used fit_transform on Xtrain

```

print("Number of data points in train data :",xv_train.shape, Ytrain.shape)
print("Number of data points in test data :",xv_test.shape, Ytest.shape)

```

```

Number of data points in train data : (1319655, 1017272) (1319655,)
Number of data points in test data : (439885, 1017272) (439885,)

```

Logistic Regression

Fit the Logistic Regression model.

```
LR = LogisticRegression()  
LR.fit(xv_train,Ytrain)
```

Predicted values

```
#Prediction  
pred_lr=LR.predict(xv_test)
```

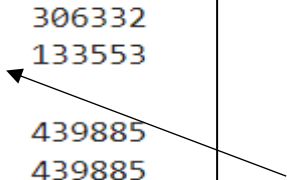
```
print('Test Accuracy using Logistics Regression Classifier: ',accuracy_score(Ytest,pred_lr))
```

Test Accuracy using Logistics Regression Classifier: 0.9505643520465576

Accuracy Score using Logistic Regression Classifier is **95.05%**.

```
print(classification_report(Ytest,pred_lr))
```

	precision	recall	f1-score	support
0	0.98	0.95	0.96	306332
1	0.90	0.95	0.92	133553
accuracy			0.95	439885
macro avg	0.94	0.95	0.94	439885
weighted avg	0.95	0.95	0.95	439885



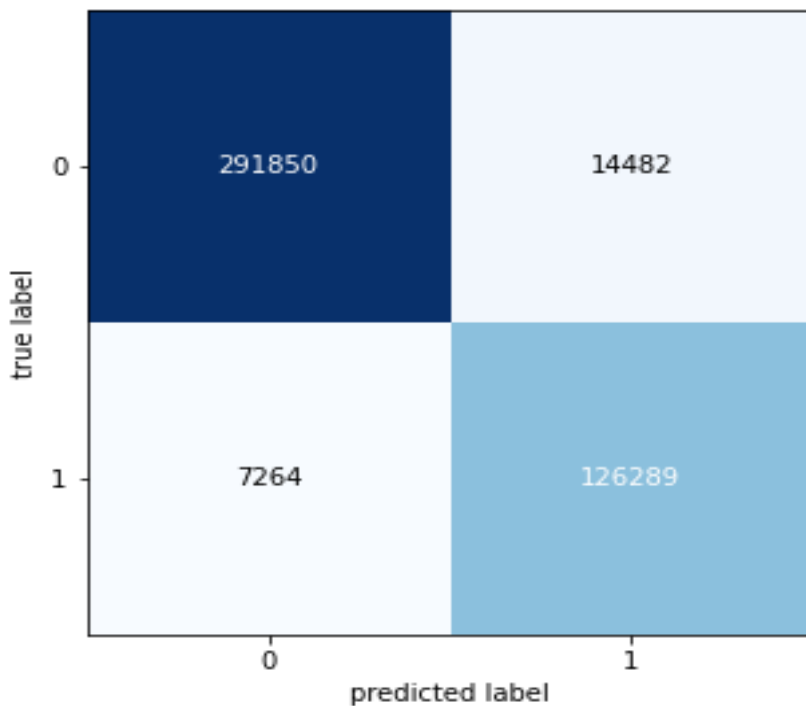
Fig(17): CR for LR

Here, the Presicion, recall and f1-score values for both classes are accurate.

```
print(confusion_matrix(Ytest,pred_lr))
CM_lr = confusion_matrix(Ytest,pred_lr)

fig, ax = plot_confusion_matrix(conf_mat=CM_lr ,  figsize=(5, 5)) #Plotting Confusion Matrix using Heatmap
plt.show()

[[291850  14482]
 [  7264 126289]]
```



Fig(18): Confusion Matrix

TP = 291850

FN = 14482

FP = 7264

TN = 126289

So, the True Positive and True Negative are classified in very good count.

This Model is now best fitted.

```
print(f"Correctly classified sincere questions: {round(CM_lr[0][0]/(CM_lr[0][0] + CM_lr[0][1])*100, 2)}%")
print(f"Correctly classified insincere questions: {round(CM_lr[1][1]/(CM_lr[1][0] + CM_lr[1][1])*100, 2)}%")
```

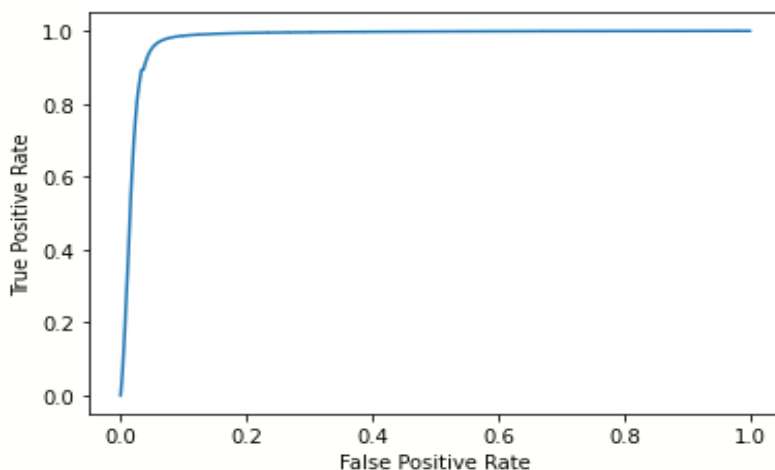
Correctly classified sincere questions: 95.27%
Correctly classified insincere questions: 94.56%

Correctly Classified both classes are in more accurate.

>ROC – AUC Curve

```
y_pred_proba_LR = LR.predict_proba(xv_test)[::,1]
fpr, tpr, _ = metrics.roc_curve(Ytest, y_pred_proba_LR)
```

```
#create ROC curve
plt.plot(fpr,tpr)
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```



Fig(18): ROC – AUC curve LR

Conclusion:

This Logistic Regression model is correctly classifying the both sincere and insincere questions.

We can use this model for our Sincere and Insincere question classification.

Multinomial- Naïve Bayes

```
NB = MultinomialNB()  
NB.fit(xv_train, Ytrain)
```

```
MultinomialNB()
```

Fit the Multinomial-Naïve Bayes model.

```
#Prediction  
pred_nb=NB.predict(xv_test)
```

Predicted values

```
print('Test Accuracy using Multinomial Naive-Bayes Classifier: ',accuracy_score(Ytest,pred_nb))
```

```
Test Accuracy using Multinomial Naive-Bayes Classifier: 0.9665389817793287
```

Accuracy Score using Multinomial Naïve Bayes Classifier is **96.65%**.

```
print(classification_report(Ytest,pred_nb))
```

```
print(confusion_matrix(Ytest,pred_nb))
```

	precision	recall	f1-score	support
0	0.97	0.98	0.98	306332
1	0.95	0.94	0.94	133553
accuracy			0.97	439885
macro avg	0.96	0.96	0.96	439885
weighted avg	0.97	0.97	0.97	439885

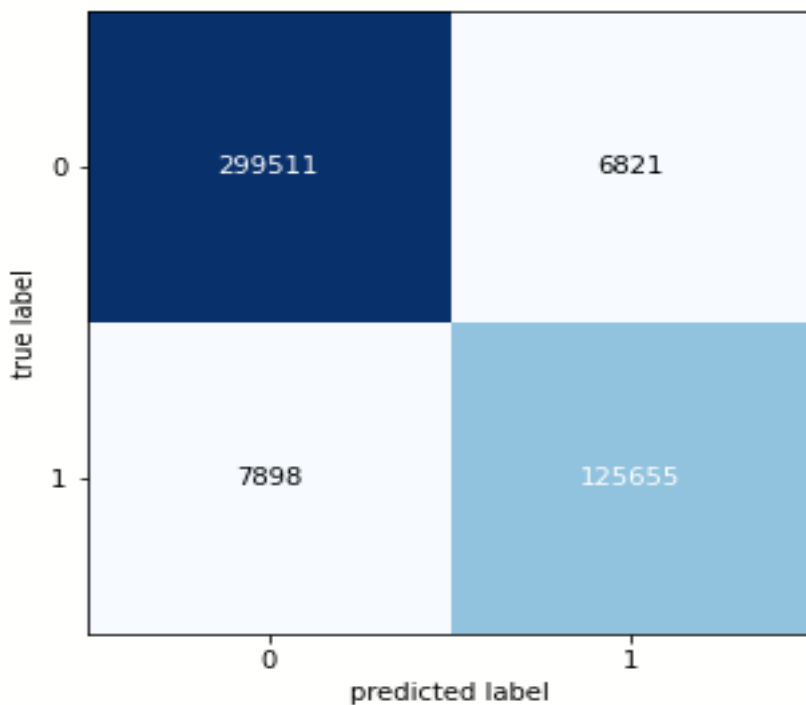
```
[[299511  6821]  
 [ 7898 125655]]
```

Here, the Presicion, recall and f1-score values for both classes are accurate.

```
print(confusion_matrix(Ytest,pred_nb))
CM_nb = confusion_matrix(Ytest,pred_nb)

fig, ax = plot_confusion_matrix(conf_mat=CM_nb , figsize=(5, 5)) #Plotting Confusion Matrix using Heatmap
plt.show()
```

```
[[299511  6821]
 [ 7898 125655]]
```



TP = 299511

FN = 6821

FP = 7898

TN = 125655

So, the True Positive and True Negative are classified in very good count.

This Model is now best fitted.

```
print(f"Correctly classified sincere questions: {round(CM_nb[0][0]/(CM_nb[0][0] + CM_nb[0][1])*100, 2)}%")
print(f"Correctly classified insincere questions: {round(CM_nb[1][1]/(CM_nb[1][0] + CM_nb[1][1])*100, 2)}%")
```

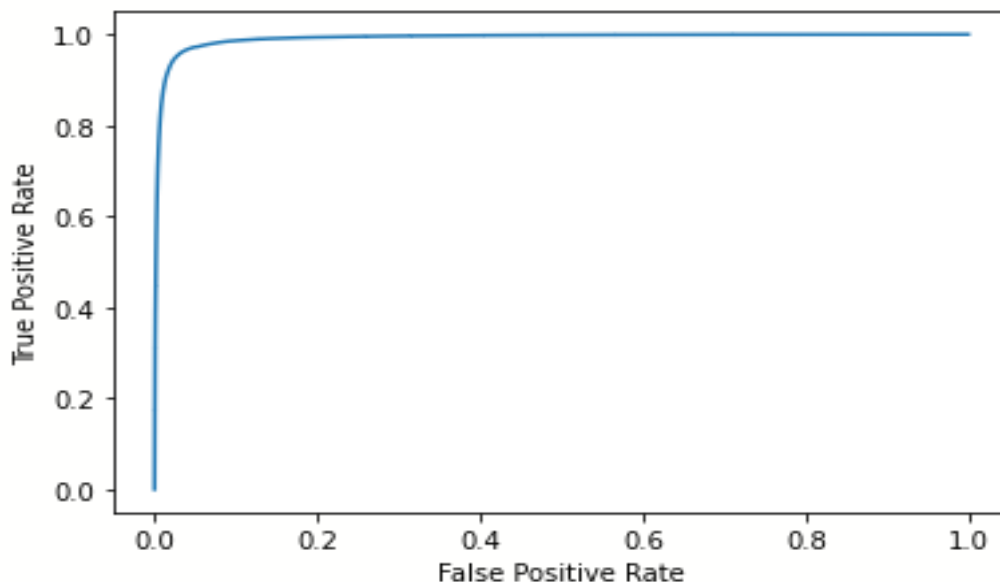
Correctly classified sincere questions: 97.77%
Correctly classified insincere questions: 94.09%

Correctly Classified both classes are in more accurate.

>ROC – AUC Curve

```
y_pred_proba_NB = NB.predict_proba(xv_test)[::,1]
fpr, tpr, _ = metrics.roc_curve(Ytest, y_pred_proba_NB)

#create ROC curve
plt.plot(fpr,tpr)
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```



Conclusion:

This Multinomial – Naïve Bayes model is correctly classifying the both sincere and insincere questions.

We can use this model for our Sincere and Insincere question classification.

After model building some Sincere and Insincere words.

```
!pip install regex eli5 emoji
import eli5

eli5.show_weights(LR, vec=tfv, top=50, feature_filter=lambda x:x != '<BIAS>')
```

Weight?	Feature
+16.472	castrat
+14.980	ia m
+12.844	ita
+11.385	dona
+11.289	dona t
+11.268	muslim
+11.243	trumpa
+11.141	castrated
+10.891	liber
+9.447	fuck
+8.945	whata
+8.870	democrat
+8.780	jew
+8.762	rape
+8.596	cana t
+8.512	cana
+8.381	moron
+8.072	black peopl
+7.903	trump
+7.616	doesna t
+7.616	doesna
+7.588	hillari
+7.574	castration
+7.456	feminist
+7.451	white peopl
+7.283	isna
+7.283	isna t
+7.186	homosexu
+7.105	gay
+6.906	crap
+6.853	obama
+6.815	theya

+6.815	theya
+6.794	leftist
+6.691	asshol
+6.638	stupid
+6.621	woman
+6.574	illiter
... 709794 more positive ...	
... 307428 more negative ...	
-6.670	import
-6.780	learn
-6.794	affect
-6.885	scope
-7.021	mean
-7.217	novel
-7.333	exampl
-7.364	effect
-7.467	studi
-7.691	happen
-8.022	book
-10.397	differ
-10.984	best

Fig(20):After Model Building words

8.Prediction System

```
def output_lable(n):
    if n == 0:
        return "Yes!! This question is Sincere :)"
    elif n == 1:
        return "This Question Looks like an Insincere :("

def manual_testing(question):
    testing_question = {"text":[question]}
    new_def_test = pd.DataFrame(testing_question)
    new_x_test = new_def_test["text"]
    new_xv_test = tfv.transform(new_x_test)
    pred_LR = LR.predict(new_xv_test)
    pred_NB = NB.predict(new_xv_test)

    return print("LR Prediction:{} \nNB Prediction: {}".format(output_lable(pred_LR[0]),
                                                                output_lable(pred_NB[0])))
```

Fig(21): Prediction System

This is the Input Cell

```
question = str(input())
manual_testing(question)
```

we have to enter our question here.

Some Sincere and Insincere questions predictions as below.

Insincere Questions:

```
question = str(input())
manual_testing(question)
```

```
Why is Jinping the biggest serial religion murderer and rapist leader of the world after Mao?
LR Prediction:This Question Looks like an Insincere :(
NB Prediction: This Question Looks like an Insincere :(
```

```
question = str(input())
manual_testing(question)
```

Why was Nehru a serial Ra-apist and "pimp" in India?
LR Prediction:This Question Looks like an Insincere :(
NB Prediction: This Question Looks like an Insincere :(

```
question = str(input())
manual_testing(question)
```

Who will win in video game battle: Adam d angelo vs Kim kardasian?
LR Prediction:Yes!! This question is Sincere :)
NB Prediction: Yes!! This question is Sincere :)

```
question = str(input())
manual_testing(question)
```

Was AIDS a punishment from God for gay people?
LR Prediction:This Question Looks like an Insincere :(
NB Prediction: This Question Looks like an Insincere :(

Sincere Questions:

```
question = str(input())
manual_testing(question)
```

How can I top CBSE in 6 months?
LR Prediction:Yes!! This question is Sincere :)
NB Prediction: Yes!! This question is Sincere :)

```
question = str(input())
manual_testing(question)
```

What is the writing style of the book "How to Resist Prince Charming" by Linda Kage?
LR Prediction:Yes!! This question is Sincere :)
NB Prediction: Yes!! This question is Sincere :)

```
question = str(input())
manual_testing(question)
```

Someone breaks into your house you shoot and kill them they were armed with only a knife what happens now?
LR Prediction:Yes!! This question is Sincere :)
NB Prediction: Yes!! This question is Sincere :)

```
question = str(input())
manual_testing(question)
```

What is the best way to propose a girl without annoying her?
LR Prediction:Yes!! This question is Sincere :)
NB Prediction: Yes!! This question is Sincere :)

```
question = str(input())
manual_testing(question)
```

Is there any woman (other than Muslim) who is happy after marrying a Muslim guy, as there are a lot of answers on Quora about love jihad, and all of them suffered a lot after marrying
LR Prediction:This Question Looks like an Insincere :(
NB Prediction: This Question Looks like an Insincere :(

```
question = str(input())
manual_testing(question)
```

How can you solve this equation? $\sin(a*t) - b*t = 0$?
LR Prediction:Yes!! This question is Sincere :)
NB Prediction: Yes!! This question is Sincere :)

```
question = str(input())
manual_testing(question)
```

Are you going to kill me with laughfter?
LR Prediction:Yes!! This question is Sincere :)
NB Prediction: Yes!! This question is Sincere :)

```
question = str(input())
manual_testing(question)
```

How to kill someone?
LR Prediction:This Question Looks like an Insincere :(
NB Prediction: Yes!! This question is Sincere :)

```
question = str(input())
manual_testing(question)
```

How do I train my dogs to kill raccoons?
LR Prediction:Yes!! This question is Sincere :)
NB Prediction: Yes!! This question is Sincere :)

```
question = str(input())  
manual_testing(question)
```

```
i do not like apples?  
LR Prediction:Yes!! This question is Sincere :)  
NB Prediction: Yes!! This question is Sincere :)
```

Fig(22): Prediction Output

9.Model Deployment

To deploy a model I used Spyder Python Notebook.

```
xtrain, xtest, ytrain, ytest= train_test_split(X, Y, test_size=0.25, random_state= 2000, shuffle= True)

from sklearn.pipeline import Pipeline

pipe_nb = Pipeline(steps=[('bow', TfidfVectorizer()), ('nb', MultinomialNB())])

pipe_nb.fit(xtrain, ytrain)

Pipeline(steps=[('bow', TfidfVectorizer()), ('nb', MultinomialNB())])

pipe_nb.score(xtest, ytest)

0.8827579935665003
```

Fig(23)Model Deployment

Accuracy we are getting by using Naïve Bayes is 88.27%.

```
#Prediction
ques = "Please, Enter your question"
```

```
pipe_nb.predict([ques])

array([0])
```

```
pipe_nb.predict_proba([ques])

array([[0.52955237, 0.47044763]])
```

```
pipe_nb.classes_

array([0, 1])
```

```
#Save model & Pipeline
import joblib
pipeline_file = open("question_model.pkl", "wb")
joblib.dump(pipe_nb, pipeline_file)
pipeline_file.close()
```

For model deployment I've created one new prediction function.

For model deployment first of all I have to create one new model and save that in same repository. This model will use for building.

Model I have saved using joblib library.

To save model I used Google Colab Notebook & for deployment Spyder Notebook have been used.

Function code for Multi-nomial Naïve bayes model deployment.

```
import streamlit as st
```

```
import numpy as np
```

```
import pandas as pd
```

```
import joblib
```

```
# Fxn
```

```
pipe_nb = joblib.load(open("D:/Arvind IMP/BSE Sem 2/Black  
Book/Streamlit/question_model.pkl", "rb"))
```

```
def predict_emotions(docx):
```

```
    results = pipe_nb.predict([docx])
```

```
    return results[0]
```

```
def get_prediction_proba(docx):
```

```
    results = pipe_nb.predict_proba([docx])
```

```
return results
```

```
def main():  
    st.title("Insincere Question Classifier App")  
    menu = ["Home", "Monitor", "About"]  
    choice = st.sidebar.selectbox("Menu", menu)  
  
    if choice == "Home":  
        st.subheader("Home-Question in text")  
  
        with st.form(key='question_clf'):  
            raw_text = st.text_area("Type Here")  
            submit_text = st.form_submit_button(label='Submit')  
  
        if submit_text:  
            col1, col2 = st.columns(2)  
  
            #Apply func here  
            prediction = predict_emotions(raw_text)  
            probability = get_prediction_proba(raw_text)  
  
            with col1:  
                st.success("Original Text")
```

```
st.write(raw_text)
```

```
st.success("Prediction")
```

```
st.write(prediction)
```

```
st.write("Confidence: { }".format(np.max(probability)))
```

```
if (prediction == [0]):
```

```
    st.write("Yes!! This question is Sincere :)")
```

```
else:
```

```
    st.write("This Question Looks like an Insincere :)")
```

```
with col2:
```

```
    st.success("Prediction Probability")
```

```
    st.write(probability)
```

```
    proba_df = pd.DataFrame(probability, columns=pipe_nb.classes_)
```

```
    st.write(proba_df.T)
```

```
    proba_df_clean = proba_df.T.reset_index()
```

```
    proba_df_clean.columns = ["emotions", probability]
```

```
elif choice == "Monitor":
```

```
    st.subheader("Monitor App")
```

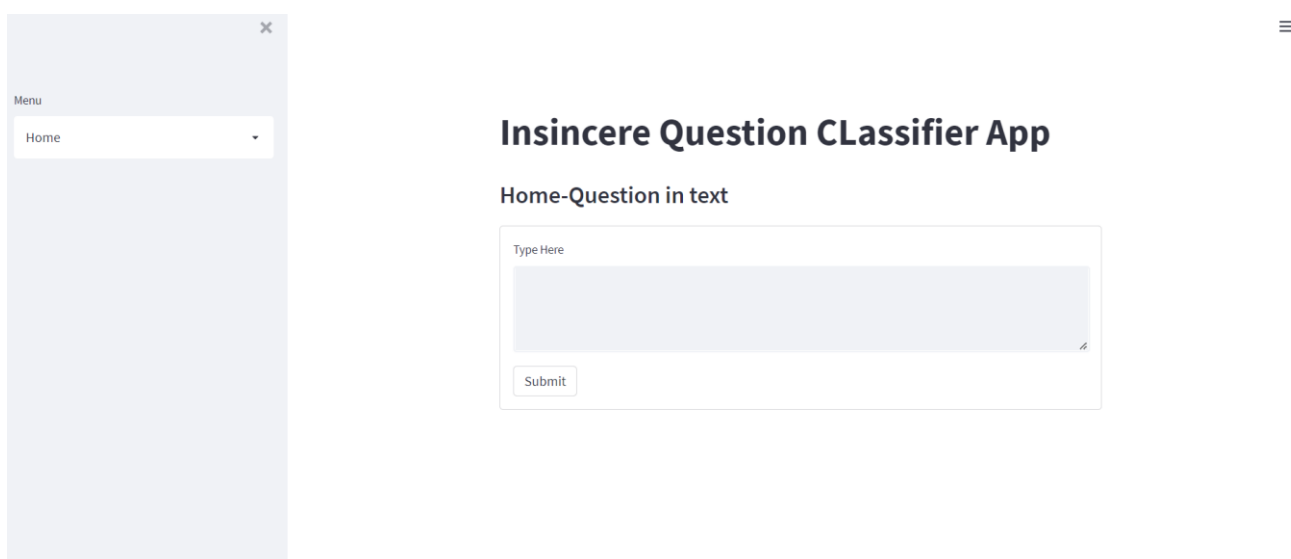
```
else:
```

```
    st.subheader("About")
```

```
if __name__ == '__main__':
```

```
    main()
```

Interface of the streamlit deployment.



Fig(24): Streamlit Interface

Insincere Question Classifier App

Home-Question in text

Type Here

Why are Americans, British, Canadians, Australians and New Zealanders considered to be separate nations even when they all speak the same language?

Submit

Original Text

Why are Americans, British, Canadians, Australians and New Zealanders considered to be separate nations even when they all speak the same language?

Prediction

Prediction Probability

	0	1
0	0.1108	0.8892

	0
0	0.1108
1	0.8892

1

Confidence:0.8891506521457421

This Question Looks like an Insincere :(

Fig(25):Insincere question output

Insincere Question Classifier App

Home-Question in text

Type Here

Can we use our external hard disk as a OS as well as for data storage.will the data be affected?

Submit

Original Text

Can we use our external hard disk as a OS as well as for data storage.will the data be affected?

Prediction Probability

	0	1
0	0.9483	0.0517

Prediction

0

Confidence:0.9482586016589405

Yes!! This question is Sincere :)

	0
0	0.9483
1	0.0517

Fig(26): Sincere question Output

- 1)We can see first the Input text box where we can enter question.
- 2)Original Text
- 3)Prediction Probability
- 4)Prediction

Chapter 7 – Conclusion

We were able to analyse the ‘Quora insincere Questions Classification’ dataset to classify the questions as sincere and insincere. We processed the dataset using various word embedding model such as TF-IDF. We tried to handle the dataset imbalance using Text Augmentation, Resampling method, Random Over Sampling, and Copy paste the minority data. We trained two models such as-Naïve Bayes, Logistic Regression.

We observed that Machine learning algorithms (Logistic Regression and Naïve Bayes) can achieve high accuracy for majority of data classifying insincere questions without handling the imbalance, i.e., they misclassify the Insincere questions. The best results were obtained after Balancing the data. Using Random Over Sampling the model becomes over fitted. So, I used copy paste minority data method.

When we have the given data, then the data has **high variance** and **high bias** towards Sincere Questions, that’s why the model was underfitted.

Now the model is best fit with Multinomial Naïve Bayes classifier is giving us best **96.56%** accuracy.

Chapter 8 - Recommendations

1. It is important to Understand the problem statement and business understanding.
2. We need to improve our analyse power.
3. Then we need to check Bias and Variance of the dataset.
4. Data Should be clean.

Chapter 9 - Bibliography

<https://www.kaggle.com/competitions/quora-insincere-questions-classification/overview>

<https://towardsdatascience.com/quora-insincere-questions-classification-d5a655370c47>

<https://medium.com/analytics-vidhya/quora-insincere-question-classification-eda-41add82a2d0b>

https://www.researchgate.net/publication/334549103_Quora_Insincere_Questions_Classification

http://cs229.stanford.edu/proj2019aut/data/assignment_308832_raw/26647500.pdf

<https://www.quora.com/What-are-your-thoughts-on-the-Quora-insincere-questions-classification-challenge-on-Kaggle>

<https://streamlit.io/>

Chapter 10- Annexure

SMOTE - Synthetic Minority Oversampling Technique.

TF-IDF – Term Frequency Inverse Document Frequency.

ROC – Receiver Operating Characteristic

AUC – Area Under Curve

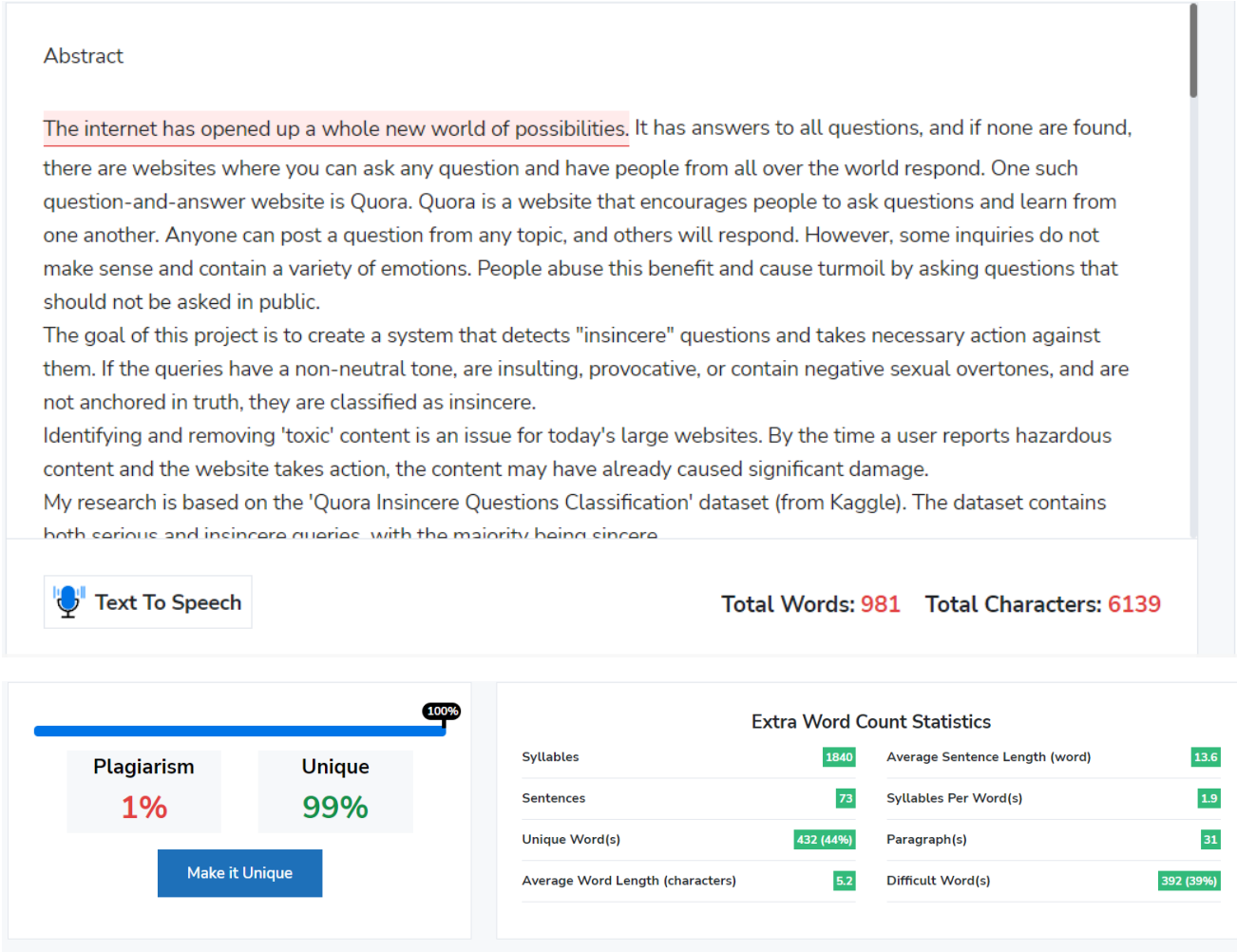
BOW – Bag of Words

CM – Confusion Matrix

Cm_lr – Confusion matrix for Logistic regression model.

Cm_nb – Confusion matrix for naïve bayes.

Chapter 11- Plagiarism report



Fig(27):Plagiarism Report

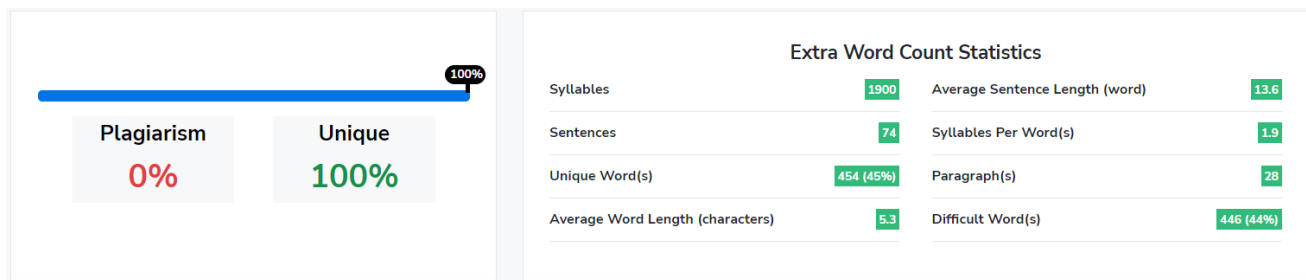
Chapter 2 – Statement of the problem

Quora is a prominent online platform where users may ask questions and (usually) receive thoughtful responses. While the majority of questions are posed in good faith, a small minority of bad actors offer questions that are either disingenuous or problematic (for example, questions based on misleading premises or questions that are just intended to make a statement).

In order to improve the overall community experience, a task is to create a classifier that will take a user-generated question as input and automatically categorise it as Sincere or Insincere.

Chapter 3 – Literature Review

In any fraud detection or more of the binary classification dataset imbalance data problem we face in this kind of data sets, and we have to use resampling, SMOTE library, or embedding methods to balance the dataset. In our problem I used resampling method.



3.Data Preprocessing & Cleaning

1.Handle Imbalance data

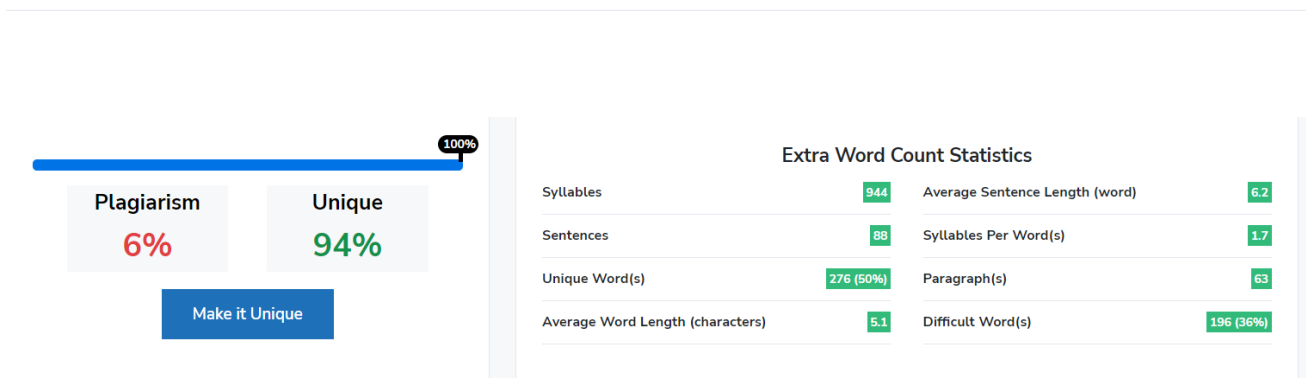
To handle imbalance text data, we have to increase the size of the minority data. Following are the methods,

1. Text Augmentation: Here we generate duplicate texts. Using Pyaug library.
2. Resampling: There are two methods Under sampling and Over Sampling.
3. Copy minority data: Copy Minority data for how many times we want.

>In my research I firstly used Copy Minority data method to increase the size of minority of data. And I haven't use Text Augmentation, because it needs to High GPU Ram. So it is taking so much time to augment texts.

>Minority data size is 80810.

>I copied this data 7 times and the overall size I got is 565670.



Mentor Approval

Project approval Inbox x

chirag jhumkhawala

to me ▼

You have approval for your dissertation project.

Great, thanks for letting me know!

Thanks a lot.

Thank you!