Alma Mater Studiorum
University of Bologna

Duplicate Question Pair

Course: Machine Learning Mini Project

Presented by Priti Kumari Gupta

Contents

1 Introduction

In today's digital age, where information is readily available at our fingertips, question-and answer platforms have become increasingly popular. Quora is one of the popular platforms where users can ask questions and receive answers from a diverse community. Over 100 million people visit Quora every month, so it's no surprise that people ask similar-worded questions. Multiple questions with the same intent can cause seekers to spend more time finding the best answer to their question, and make writers feel they need to answer multiple versions of the same question. Quora uploaded a competition on Kaggle where users were challenged to tackle the problem of duplicate questions having different threads.

This project aims to detect duplicate question pairs on Quora by utilizing the power of machine learning techniques. The main aim of the project is to develop a model capable of accurately identifying and classifying duplicate question pairs, automating the process, and improving the overall user experience on Quora. At present, Quora utilizes a Random Forest model for detecting duplicate questions. Following a similar approach, we initially implemented a Random Forest model to evaluate its performance on the dataset. Later implement xgboost, Logistic Regression, Naive Bayes to check the performance.

2 Data

After Importing the Library and dataset, we performed a preliminary Exploratory Data Analysis to gather insights and understand its characteristics. Quora provided two datasets, the train and test datasets. The training dataset contained approximately 400,000 pairs of questions, while the test dataset comprised a set of 1 million question pairs. The training dataset contains real-time data whereas the test data contains computer-generated data. The primary focus has been on the train set as those are the real-time values. Quora, as mentioned on Kaggle, has acknowledged that the labeling of the 'is_duplicate' column is performed by humans, resulting in a certain degree of inconsistency. It is important to consider, the ground truth labels on this dataset as being knowledgeable but not entirely accurate, as they may contain instances of incorrect labeling. More over, it was also indicated that the labels, on the whole, represent reasonable agreement, this may frequently do not hold true on a case-by-case basis for individual items in the dataset. The training dataset has five columns, including an ID column, two columns representing the ID of the question set 1 and question set 2, two columns containing the text of question 1 and

question 2 respectively, and 2 fields indicating duplicates. The dataset consists of the following columns:

- **Id**: This column represents a unique identifier for each question pair.
- **qid1**: This column contains the ID of the first question in the pair.
- **qid2**: This column contains the ID of the second question in the pair.
- **question1,question2**: This column includes the text of the first question in the pair. question2: This column includes the text of the second question in the pair.
- **is_duplicate:**: This column indicates whether the question pair is classified as a duplicate or not. It typically contains binary values (0 or 1) where 1 denotes a duplicate pair and 0 denotes a non-duplicate pair.

```
[4] df.head()
```

	id	qid1	qid2	question1	question2	is_duplicate
0	0	1	2	What is the step by step guide to invest in sh...	What is the step by step guide to invest in sh...	0
1	1	3	4	What is the story of Kohinoor (Koh-i-Noor) Dia...	What would happen if the Indian government sto...	0
2	2	5	6	How can I increase the speed of my internet co...	How can Internet speed be increased by hacking...	0
3	3	7	8	Why am I mentally very lonely? How can I solve...	Find the remainder when [math]23^{24}[/math] i...	0
4	4	9	10	Which one dissolve in water quikly sugar, salt...	Which fish would survive in salt water?	0

Figure 1: Distribution

The main objective of the project is to identify and assess the level of duplication between the two below-given questions. From the above analysis, it can be concluded that a binary classification problem is present in which we will be presented with two questions and our objective is to determine whether they are duplicates or not. After analyzing the dataset, it was found that there are few null values. One value was missing in the question 1 column whereas two more values were missing in the question 2 column.

The null values in the dataset were considered negligible in comparison to its overall dataset size, hence we dropped the rows completely. further,it was also checked if there were more than one row exactly similar to each other, and found that there were none like this. Additionally, a few more analyses were conducted to check how

```
#to check the null value
df.isnull().sum()
```

```
id              0
qid1            0
qid2            0
question1       1
question2       2
is_duplicate    0
dtype: int64
```

Figure 2: Null value

many question pairs were duplicates and how many were not, As a result, it was found that out of the total question pairs, 255,024 were identified as non-duplicates, while 149,263 were identified as duplicates. The percentage of non-duplicates and duplicates are 63.07 and 36.92 respectively. From Kaggle It was found the other

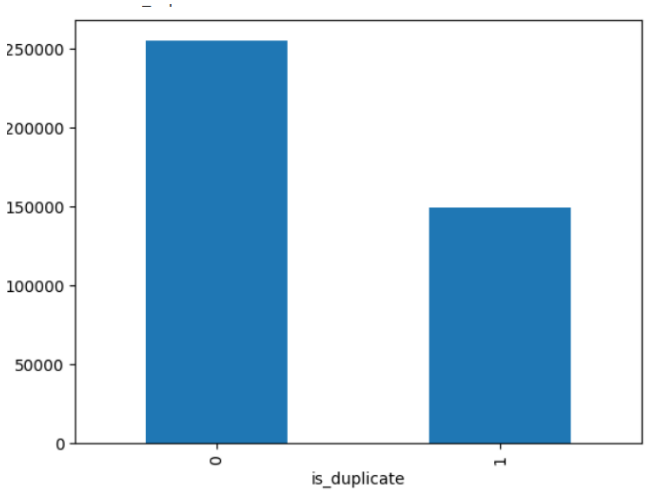

Figure 3: is_null distribution

people working on the project have identified the occurrence of the word/pattern 'MATH' approximately 900 times.

3 Preprocessing

In the preprocessing phase of the analysis, several essential steps are to clean and prepare the dataset before model evaluation:

1. **Replacement of Special Characters**: In this part replaced certain special characters with their string equivalents to ensure consistency in the text data. For example, symbols like '%', '$', ,'€', and '@' were replaced with 'percent', 'dollar', 'euro', and 'at', respectively.

2. **Pattern Replacement**: Notably, the pattern '`[math]`' appeared around 900 times throughout the dataset. To addressed this by substituting it with an empty string to remove unnecessary placeholders.

3. **Numerical Conversion**: Later converted some numerical values into their string equivalents. This included replacing large number formats with abbreviations, such as converting '1,000,000' to '1m' and '1,000' to '1k'. Though not exhaustive, this step helped standardize numeric representations in the text.

4. **Decontracting Words**: preprocessing also includes decontracted words to their full forms to avoid ambiguity and ensure uniformity. Common contractions like "can't" were expanded to "cannot", and "you're" was expanded to "you are".

5. **Removing HTML Tags**: Preprocessing also stripped HTML tags from the text data to clean the content and focus on the textual information.

6. **Removing Punctuations**: Finally, in the last steps removed punctuation marks to streamline the textual data, ensuring that only relevant words were retained for better processing by the model.

4 Feature Engineering

After preprocessing, few basic feature engineering are also conducted on the dataset. This involved extracting new features from the existing data. The features added during the feature engineering process are mentioned below with details:

- **q1_len**: char length of question1.

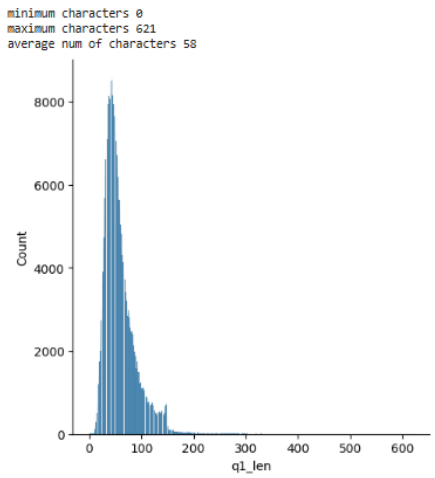

Figure 4: q1_length

- **q2_len**: char length of question2.

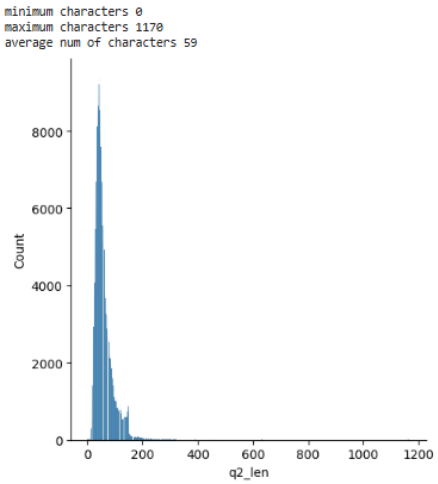

Figure 5: q2_length

- **q1_num_words**: Number of words in question1 obtained by splitting the questions in the question1 column using ” ”.

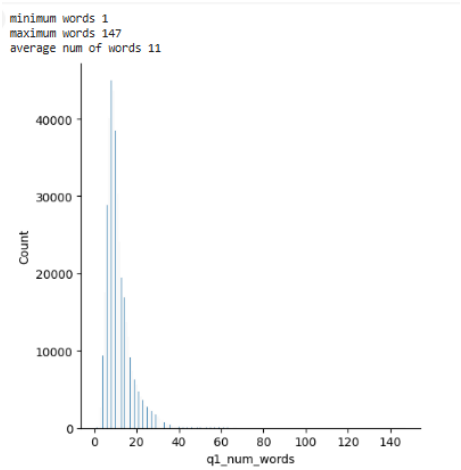

Figure 6: q1_num_word

- **q2_num_words**: Number of words in question2 obtained by splitting the questions in the question2 column using " ".

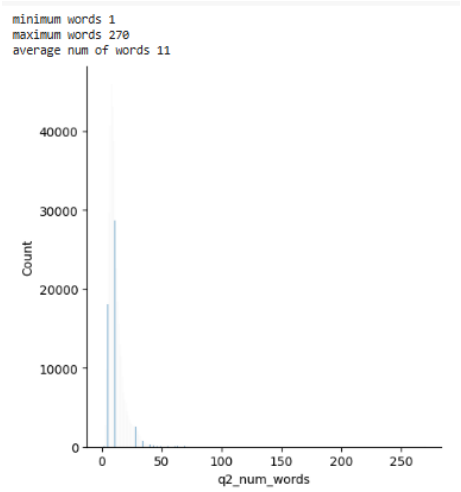

Figure 7: q2_num_word

- **word_common**: number of common unique words obtained by taking the intersection of the set of collection of all words in question1 and question2

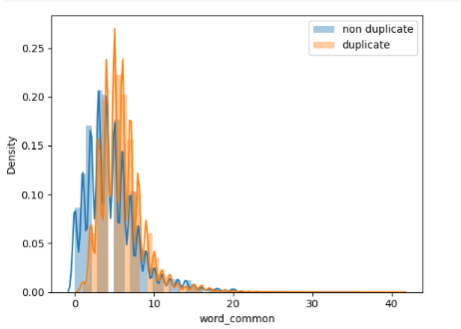

Figure 8: word_common

- **word_total**: sum of the total number of words in question1 and question2 obtained by taking the sum of a set of collections of all words in question1 and question2.

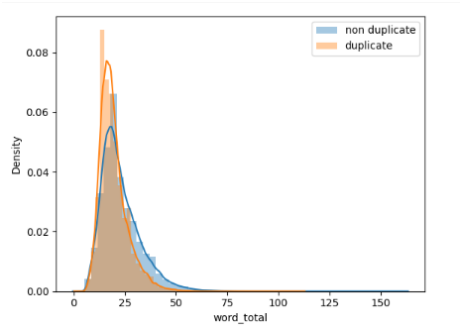

Figure 9: word_total

- **word_share**: word_common/word_total

The two columns question1 and question2 were dropped to evaluate the performance of the two models xgboost and random forest. Later the performance of the models was noted on the few features of the question pair set.

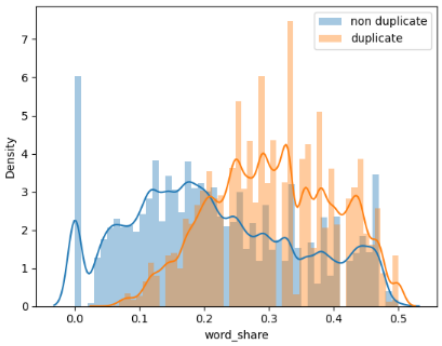

Figure 10: word_share

These are the basic feature engineering implemented further for more better results few advanced feature engineering also includes which are as below:

1. **Token Features**

   - **cwc_min**: This is the ratio of the number of common words to the length of the smaller question.
   - **cwc_max**:This is the ratio of the number of common words to the length of the larger question.
   - **csc_min**:This is the ratio of the number of common stop words to the smaller stop word count among the two questions.
   - **csc_max**: This is the ratio of the number of common stop words to the larger stop word count among the two questions.
   - **ctc_min**: This is the ratio of the number of common tokens to the smaller token count among the two questions
   - **ctc_max**: This is the ratio of the number of common tokens to the larger token count among the two questions
   - **last_word_eq**: 1 if the last word in the two questions is same, 0 otherwise
   - **first_word_eq**: 1 if the first word in the two questions is same, 0 otherwise

2. **Fuzzy Features**

- **fuzz_ratio**:fuzz_ratio score from fuzzywuzzy. This feature represents the fuzz_ratio score from fuzzywuzzy, which measures the simple edit distance between two strings.
- **fuzz_partial_ratio**:fuzz_partial_ratio from fuzzywuzzy.This feature captures the fuzz_partial_ratio from fuzzy-wuzzy, focusing on the best partial match of shorter substrings within the longer string.
- **token_sort_ratio**:token_sort_ratio from fuzzy-wuzzy, which sorts the tokens in each string and then computes the fuzz_ratio, effectively normalizing word order differences.
- **token_set_ratio**: token_set_ratio from fuzzywuzzy, which handles duplicate tokens and token order variations by considering the intersection and differences between token sets.

5 Models

5.1 RandomForest

After applying feature engineering to extract some features from the data and splitting the dataset into training and testing, We implemented our model with the Random Forest algorithm. To optimize its performance, we utilized the GridSearchCV tool from the sklearn.model_selection module to find the best set of hyperparameters. The parameter grid we used included a range of values for several important parameters:

- **n_estimators:**Number of trees in the forest, with values [300, 500].
- **criterion:**Function to measure the quality of a split, with options ['gini', 'log_loss'].
- **min_samples_split:** Minimum number of samples required to split an internal node, with values [2, 5].
- **max_features:** Number of features to consider when looking for the best split, with options ['sqrt', 'log2'].

- **bootstrap:**Whether bootstrap samples are used when building trees, with options [True, False].
- **class_weight:**Weights associated with classes, with options ['balanced', None].

After running GridSearchCV to evaluate different combinations of these parameters, the best parameters found were 'bootstrap': True, 'class_weight': 'balanced', 'criterion': 'log_loss', 'max_features': 'sqrt', 'min_samples_split': 5, 'n_estimators': 500. These optimized parameters provided the best performance for the Random Forest model in the task of detecting duplicate questions on Quora which gives an Accuracy of 76.69 and an F1 score of 69.85 respectively.

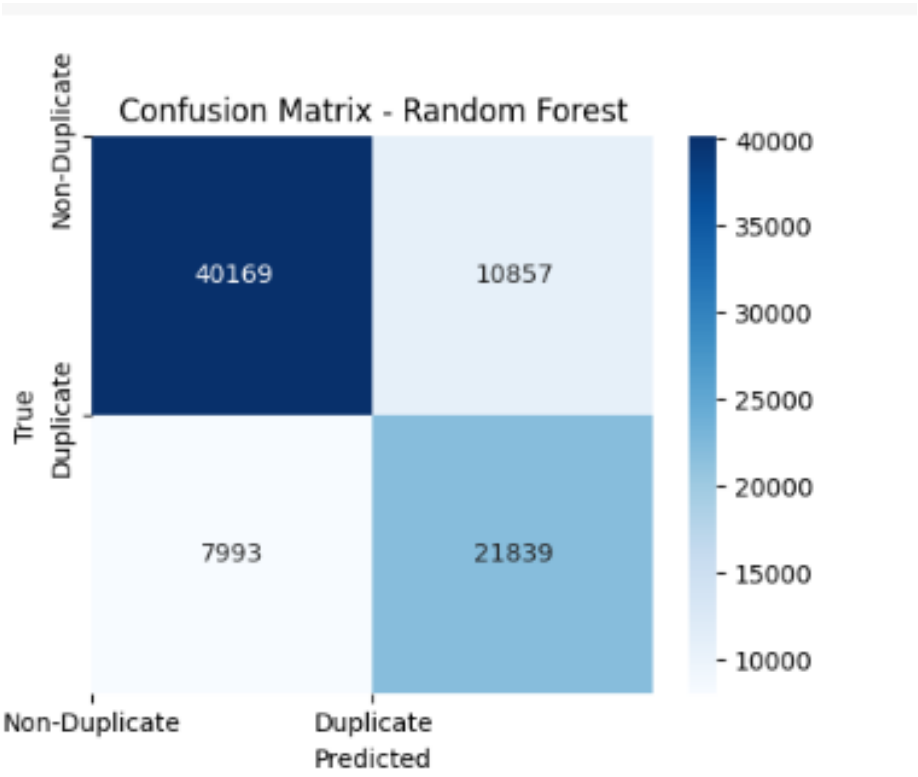

Figure 11: RandomForest Confusion Matrix

5.2 XGBoost

After the RandomForest, We chose XGBoost to implement on the model as it effectively captures interactions between the question features and identifies important

patterns contributing to question similarity. It is also known for being able to handle a large number of features and automatically learn feature interactions, making it suitable for text-based classification tasks. The parameter grid for XGBoost included a range of values for several key hyperparameters:

- **n_estimators:**Number of trees in the forest, with values [300, 500].
- **learning_rate:**value used values [0.1, 0.3].
- **max_depth:**Maximum depth of a tree, with values [3, 5, 7].
- **gamma:** Minimum loss reduction required to make a further partition on a leaf node of the tree, with values [0, 0.1].
- **scale_pos_weight:**Balancing of positive and negative weights, with values [1, 3, 5].

After conducting GridSearchCV to evaluate different combinations of these parameters, the best parameters found for the xgboost model were 'gamma': 0, 'learning_rate': 0.1, 'max_depth': 7, 'n_estimators': 500, 'scale_pos_weight': 1 The xgboost gives an Accuracy of 76.73 and an F1 score of 69.32 respectively.

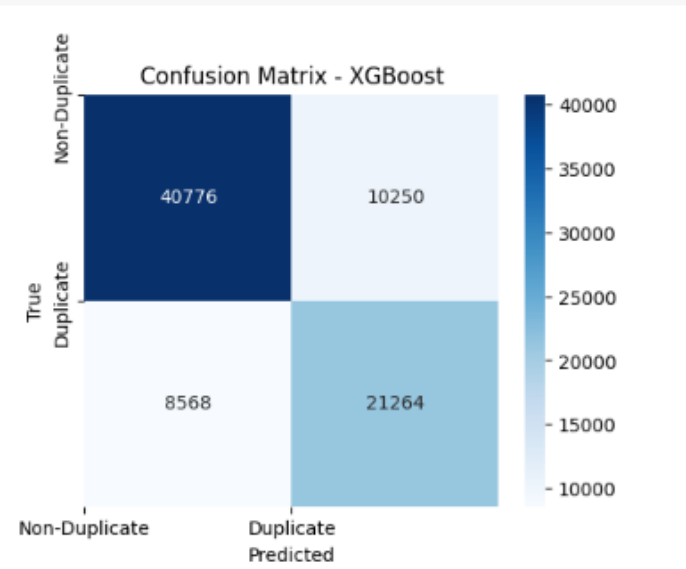

Figure 12: XGBoost Confusion Matrix

5.3 Naive bayes

We also explored the Naive Bayes algorithm, the Multinomial Naive Bayes classifier, to further enhance our duplicate question detection result. but got worse results than RandomForest and XGBoost. The parameter grid for Multinomial Naive Bayes included the following:

- **alpha:** values [0.0, 0.5, 1.0, 1.5, 2.0].
- **fit_prior:**values [True, False].

The best paramaetrs found for Naive bayes apha=0.0 and fit prior= true. The Naive Bayes gives an Accuracy and F1 score of naive Bayes are 64.31 and 60.3 respectively.

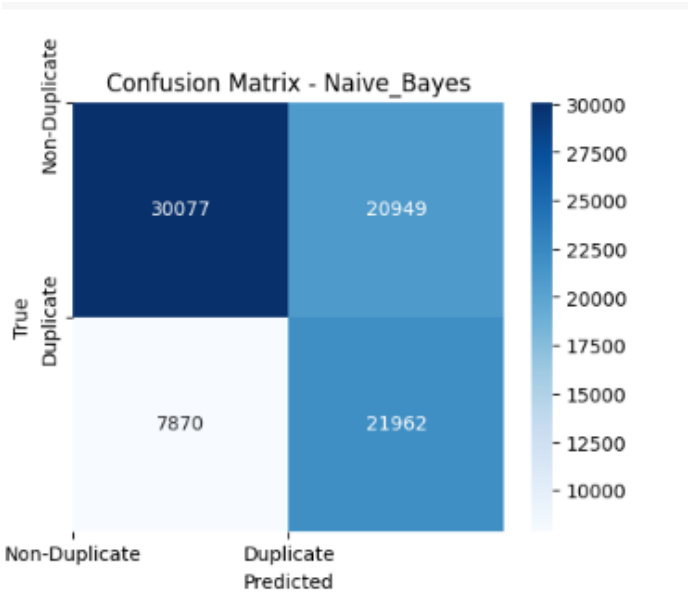

Figure 13: Naive Bayes Confusion Matrix

5.4 Logistic Regression

Last classifier used for duplicate question pair detection is Logistic regression.The parameter grid for Logistic Regression included the following:

- **C:**Inverse of regularization strength, with values [0.01, 0.1, 1
- **penalty**:Norm used in the penalization, with the option ['l2'].

- **Solver:**Algorithm to use in the optimization problem, with the option ['liblinear'].
- **class_weight:**Weights associated with classes, with options ['balanced', 'none'].

The best paramaetrs found for Logistic regression are 'C': 0.1, 'class_weight': 'balanced', 'penalty': 'l2', 'solver': 'liblinear' The Logistic regression Model gives an Accuracy of 69.46 anf F1 score of 64.47.

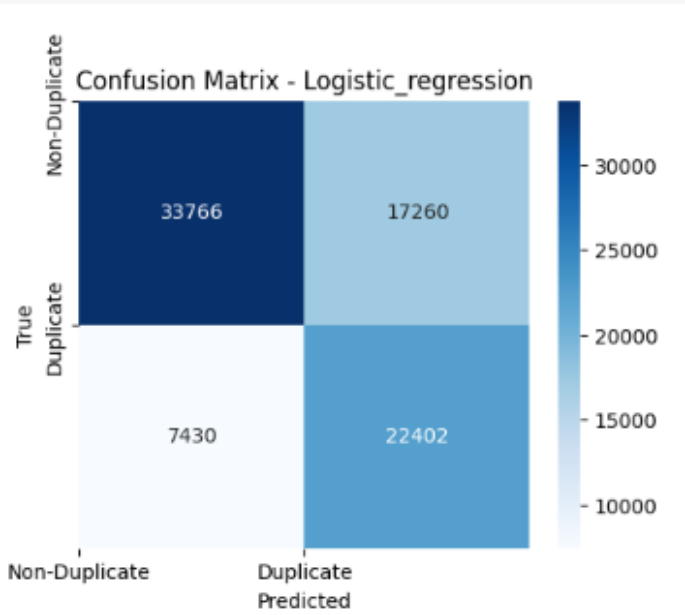

Figure 14: Logistic regression Confusion Matrix

6 Model Comparision and Conclusion

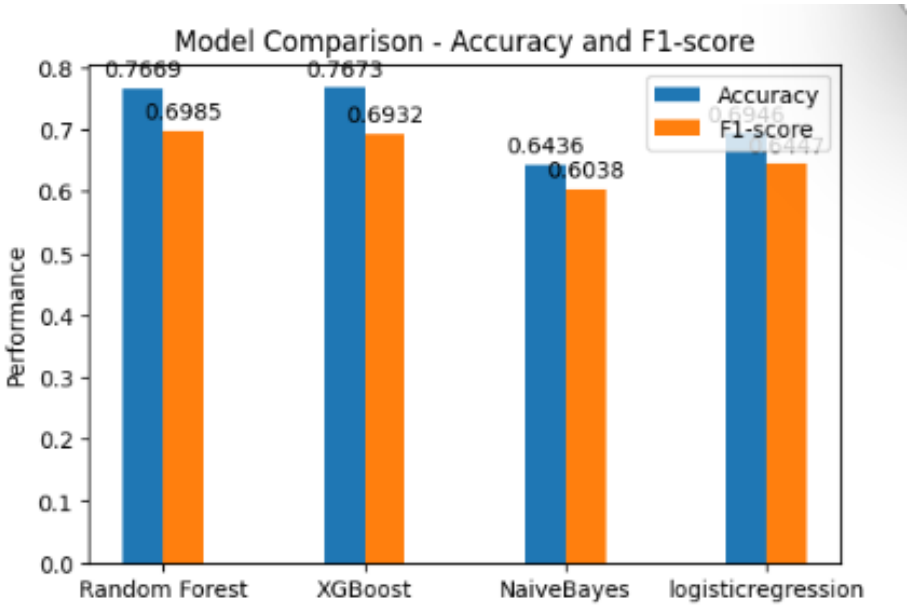

Figure 15: Models Comparision

From the above Model results it is clear that our two best models are XGboost with an accuracy of 76.73 and RandomForest with an accuracy of 76.69. According to accuracy, the XGBoost model gives a slightly better result than RandomForest. While these machine learning models provided strong baseline results, there is potential for further enhancement using advanced deep learning techniques and natural language processing (NLP) methods. Specifically, incorporating models such as Long Short-Term Memory (LSTM) networks, Bidirectional LSTM (biLSTM), Gated Recurrent Units (GRU), and transformer-based models like BERT could significantly improve the detection accuracy. These models are well-suited for capturing complex patterns and contextual information in textual data, thereby offering a more nuanced understanding of the questions' semantic similarities.