NLP PROJECT ON QUORA QUESTION PAIRS

Quora's first public dataset is related to the problem of identifying duplicate questions. At Quora, an important product principle is that there should be a single question page for each logically distinct question. For example, the queries “What is the most populous state in the USA?” and “Which state in the United States has the most people?” should not exist separately on Quora because the intent behind both is identical. Having a canonical page for each logically distinct query makes knowledge-sharing more efficient in many ways: for example, knowledge seekers can access all the answers to a question in a single location, and writers can reach a larger readership than if that audience was divided amongst several pages.

The dataset is based on actual data from Quora and will give anyone the opportunity to train and test models of semantic equivalence.

LIBRARIES USED

nltk - is a comprehensive Python library used for working with human language data (text). It provides tools to work with text processing tasks such as tokenization, stemming, tagging, parsing, and more. It also includes a variety of datasets and lexicons to help with language processing.

re - module in Python provides support for regular expressions, which allow you to search for patterns in strings. Regular expressions are extremely powerful for text processing tasks.

BeautifulSoup - for parsing HTML and XML documents.

Fuzzywuzzy - is a Python library used for string matching and comparison. It leverages Levenshtein Distance to calculate the differences between sequences and provide a similarity score, making it a popular tool for tasks like record linkage, deduplication, and fuzzy string matching.

DATA PREPROCESSING

HTML Tags are removed. The special characters are replaced by their meanings in word.
short forms are replaced by the full form.

```
Example: preprocess("I've already! wasn't <b>done</b>?")
'i have already  was not done'
```

This is done using the user-defined function ‘**preprocess’.**

FEATURE EXTRACTION

Some features such as –

1. length of each sentence (q1_len and q2_len)
2. number of words in each question (q1_num_words and q2_num_words)
3. common words in both sentences of each row
4. total words in both the sentences
5. word share = common words/total words

Distance features such as-

1. Absolute length difference

```
length_features[0] = abs(len(q1_tokens) - len(q2_tokens))
```

2. Average length of both tokens

length_features[1] = (len(q1_tokens) + len(q2_tokens))/2

3. Longest substring ratio

```
length_features[2] = len(strs[0]) / (min(len(q1), len(q2)) + 1)
```

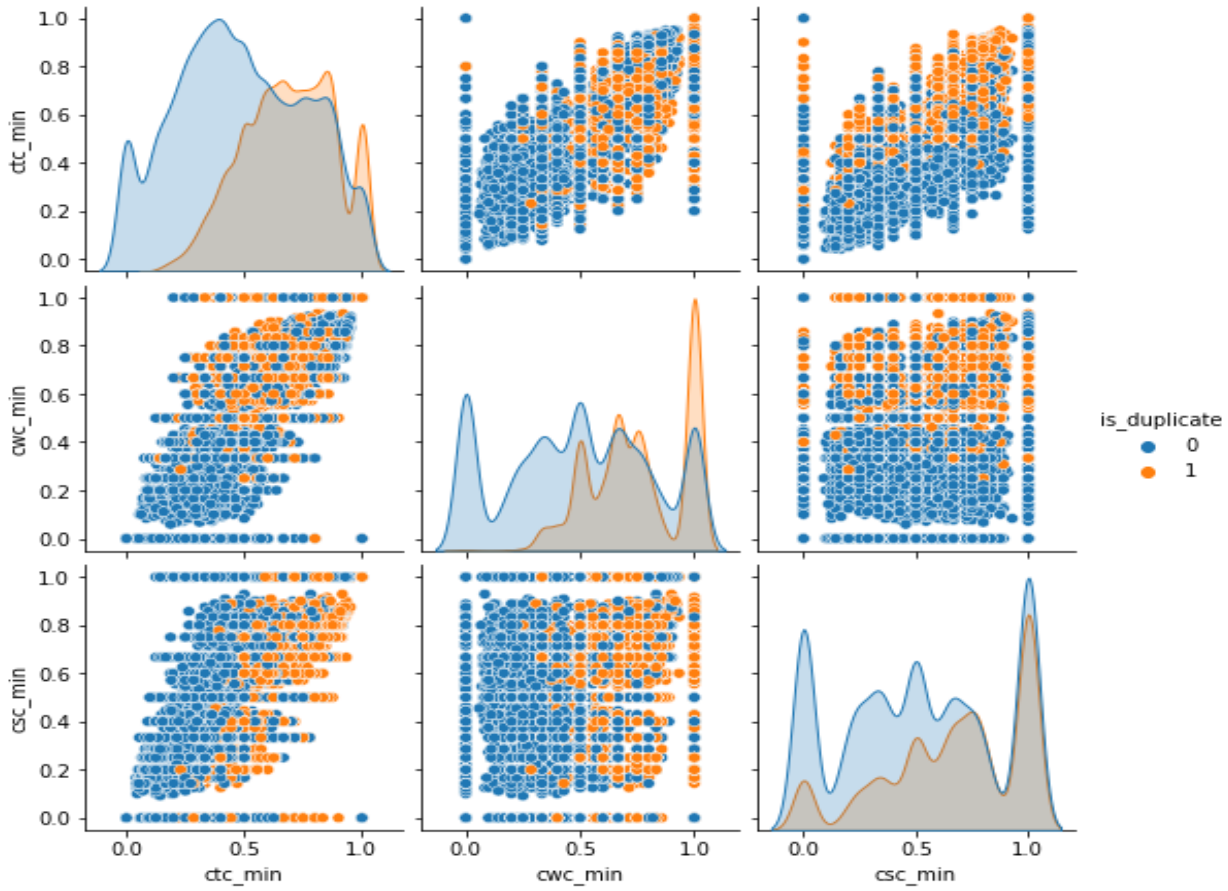
From 'fuzzywuzzy' library, 4 features are calcuted

1. The `ratio()` function calculates the similarity ratio between two strings using the Levenshtein Distance algorithm. It returns a score between 0 and 100, where 100 indicates that the two strings are identical, and a lower score indicates less similarity.
2. The `partial_ratio()` function computes the similarity score by comparing the best matching substrings between the two input strings. This method is useful when one string is significantly longer than the other, or when one string is a substring of the other.

EXPLORATORY DATA ANALYSIS

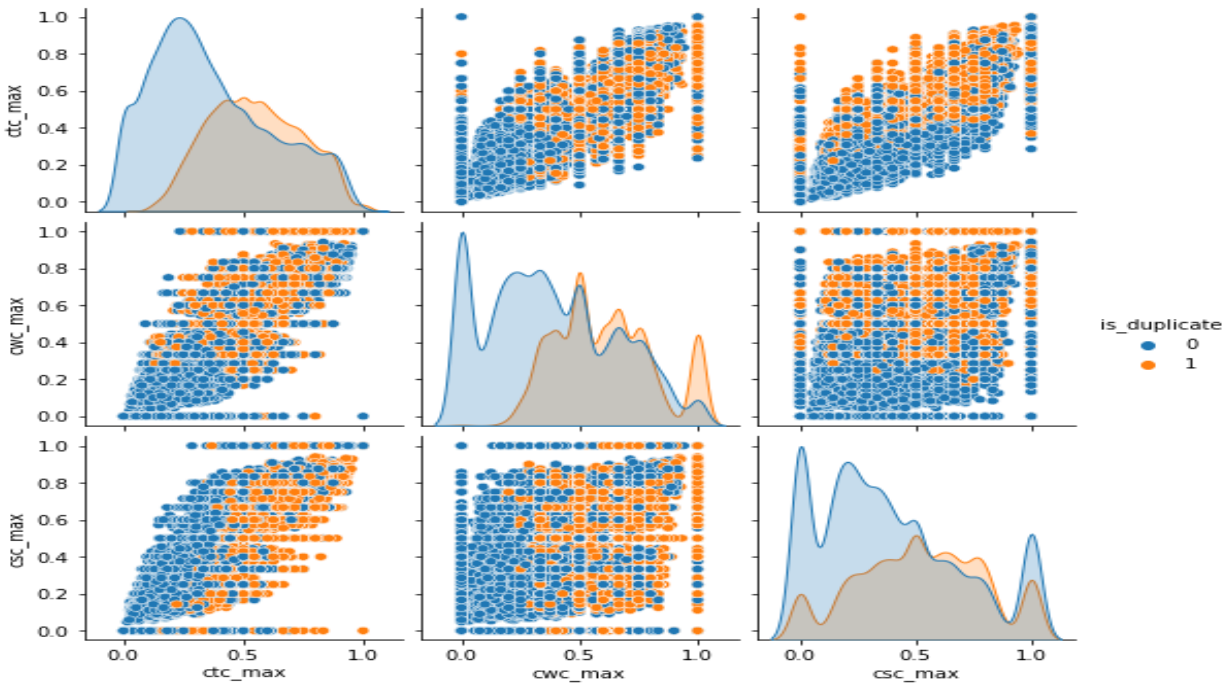
1)

```
sns.pairplot(new_df[['ctc_min', 'cwc_min', 'csc_min',
'is_duplicate']],hue='is_duplicate')
```



We are concerned with the diagonal plots. If we are able to distinguish between the two plots, we can say that the feature is significant for us.

For the above plot, all the three features can be important for us as we are able to distinguish between the two.



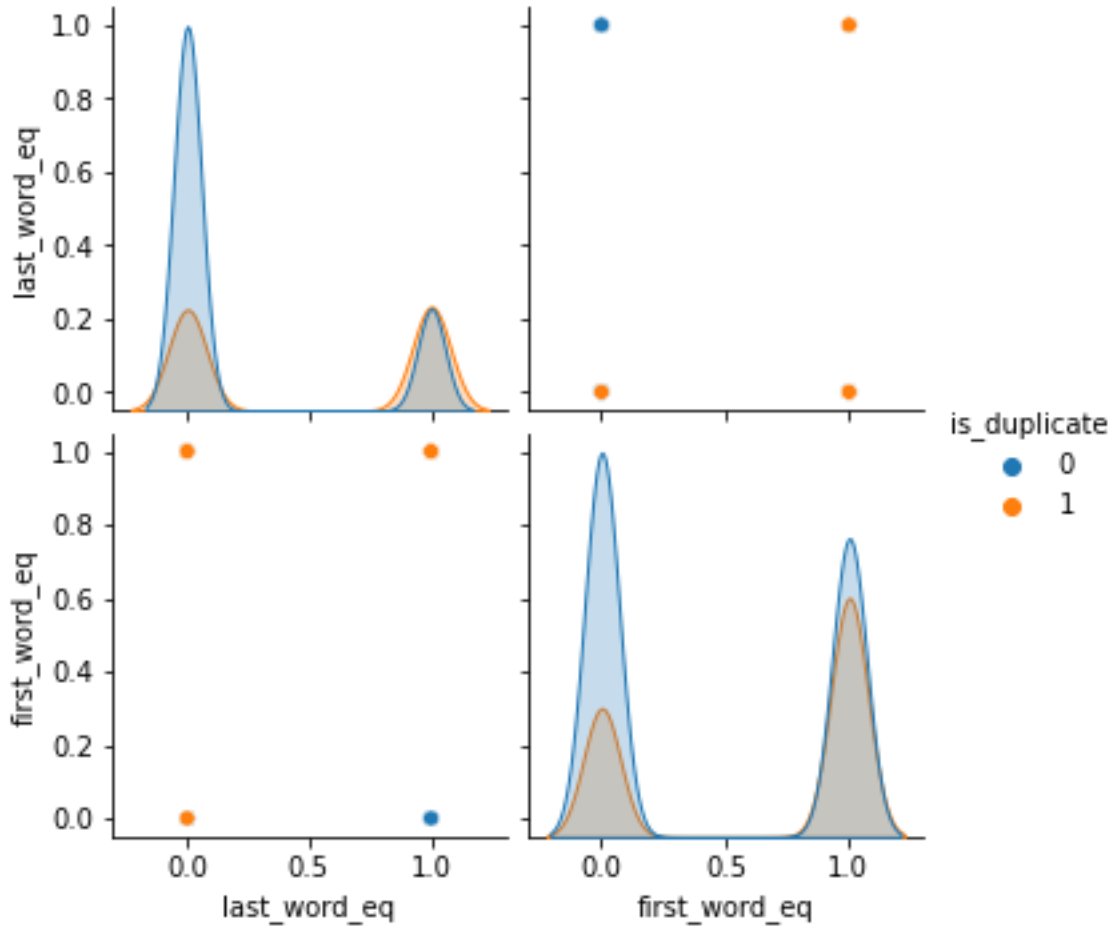
2)

```
sns.pairplot(new_df[['ctc_max', 'cwc_max', 'csc_max',
'is_duplicate']],hue='is_duplicate')
```

Similarly, we are able to distinguish between the two plots. Therefore, this is also an important feature for the data.

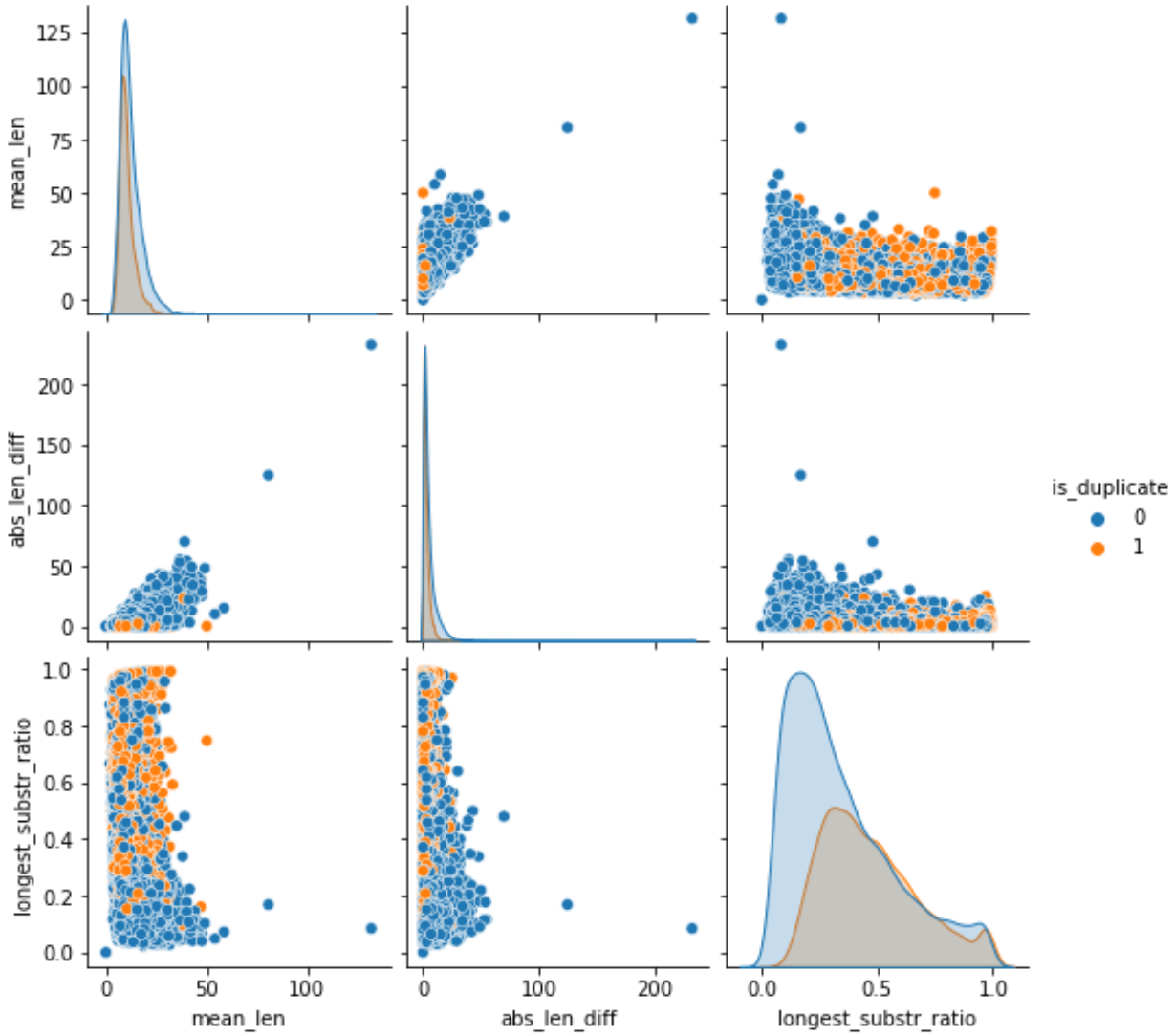
3)

```
sns.pairplot(new_df[['last_word_eq', 'first_word_eq',
'is_duplicate']],hue='is_duplicate')
```



4)

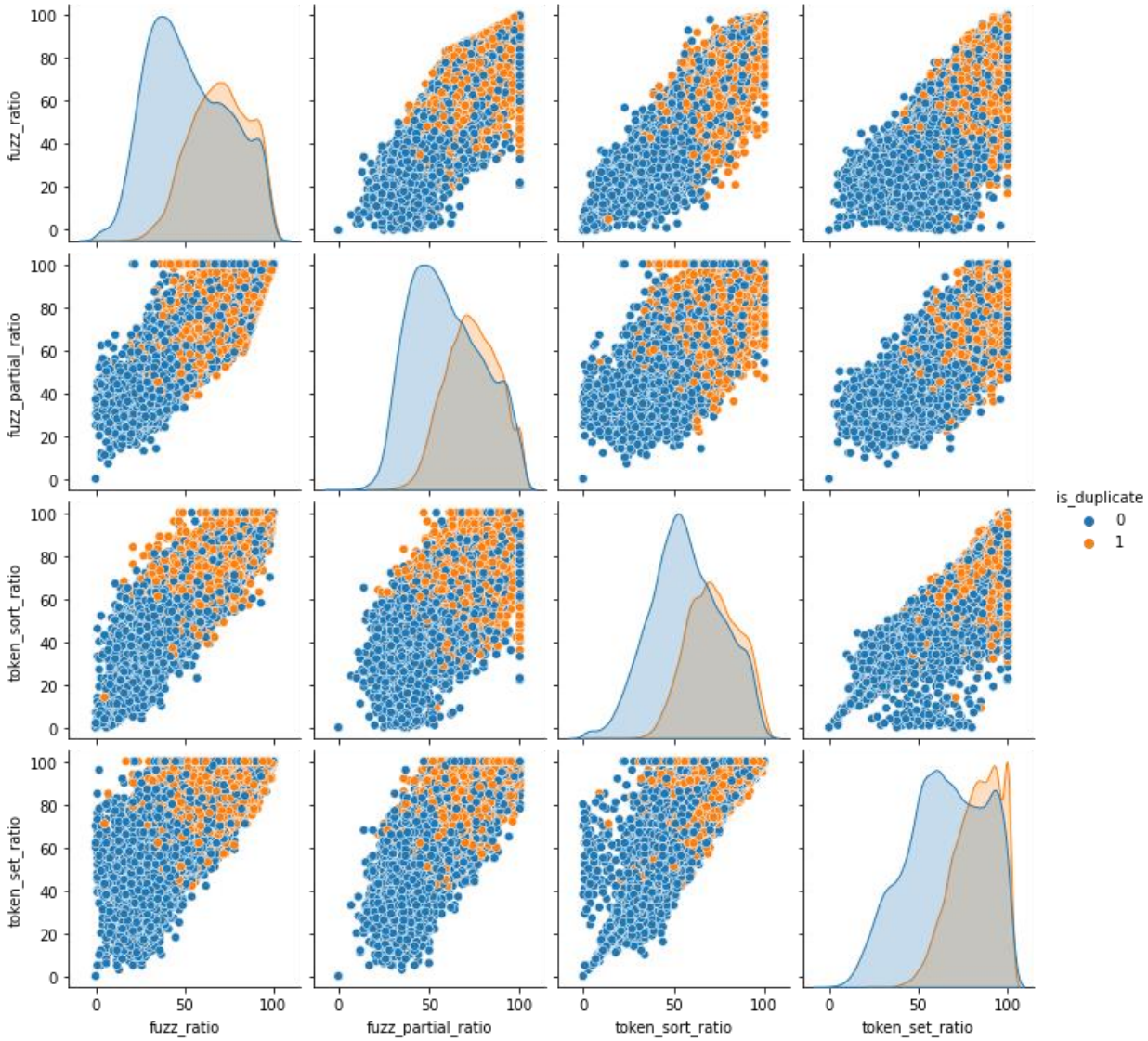
```
sns.pairplot(new_df[['mean_len', 'abs_len_diff','longest_substr_ratio',
'is_duplicate']],hue='is_duplicate')
```



In this plot, we are not able to distinguish between the red and blue one in case of 'mean_len' and 'abs_len_diff'. So, these features might not be that much significant for our model.

5)

```
sns.pairplot(new_df[['fuzz_ratio',
'fuzz_partial_ratio','token_sort_ratio','token_set_ratio',
'is_duplicate']],hue='is_duplicate')
```



Accuracy score is used to check the overall performance of the model. **Random Forest and XGBoost classifier** algorithms are used and both of them gives accuracy score of 0.74 approximately.

```
# for random forest model
confusion_matrix(y_test,y_pred)

array([[3271,  541],
       [ 751, 1437]], dtype=int64)
```

This is the confusion matrix for the random forest model.

CONCLUSION

Both the algorithms gives similar accuracy socre of 0.74 which is a good score.
whatever features we extracted from the data, we concluded that the two features might not be important which was – mean length and absolute length difference (both of them are the distance metrics).

Both the values- 0 and 1 are similarly distributed for almost all the features.