A MINOR PROJECT REPORT
ON
QUORA QUESTION PAIR

SUBMITTED IN PARTIAL FULFILLMENT FOR THE AWARD OF DEGREE OF
BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE ENGINEERING

Submitted By:

KARAN MALHOTRA (17102121)
ANWESHA SINGH(18103124)
DINI JAIN(18103130)

Under the Guidance Of
KASHAV AJMERA

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING
JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY, NOIDA (U.P.)
December, 2020

TABLE OF CONTENTS

CERTIFICATE

This is to certify that the minor project report entitled, "**QUORA QUESTION PAIR**" submitted by **KARAN MALHOTRA, ANWESHA SINGH** and **DINI JAIN** in partial fulfillment of the requirements for the award of Bachelor of Technology Degree in **Computer Science ngineering** of the Jaypee Institute of Information Technology, Noida is an authentic work carried out by them under our supervision and guidance. The matter embodied in this report is original and has not been submitted for the award of any other degree.

Signature of Supervisor:
Name of the Supervisor: Kashav Ajmera
CSE Department,
JIIT, Sec-62,
Noida

Dated: 10-12-20

DECLARATION

We hereby declare that this written submission represents our own ideas in our own words and where others' ideas or words have been included, have been adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission.

Place: New Delhi
Date: 10-12-20

Name: Karan Malhotra
Enrollment: 17102121

Name: Anwesha Singh
Enrollment: 18103124

Name: Dini Jain
Enrollment: 18103130

ACKNOWLEDGEMENTS

We wish to express our sincere appreciation to our supervisor, **Kashav Ajmera** for his timely, informative feedback and support during this effort. It has been a great pleasure to learn from him during the process of our work. He has been a generous mentor. We extend our most sincere thanks to Ankit Vidyarthi(Coordinator of Minor Project – I) for his cooperation during the project work.

Name: Karan Malhotra
Enrollment:

Name: Anwesha Singh
Enrollment: 18103124

Name: Dini Jain
Enrollment: 18103130

ABSTRACT

The purpose of this project was to learn basic machine learning using python. **Quora** is an American question-and-answer website where questions are asked, answered, followed, and edited by Internet users, either factually or in the form of opinions. The negative side of a question-answer platform is that Quora has to deal with repeated questions and answers which is a difficult task. The aim of this project is a **binary classification problem** where we need to classify duplicate and non-duplicate questions. Through the project we learned about the various models used in machine learning training. We also learned what basically is machine learning and how it can be implemented in real world problems.

We first converted our data set into numerical data set, because machine learning can only be applied on numbers, this was done using natural language processing. We used different python libraries such as NUMPY, PANDAS, SPACY, SCIPY, SCIKIT, etc, that helped us in processing the data. Then the data was processed more to find out some of its basic and advanced features using libraries and functions like, FUZZY RATIO, TOKEN RATIO, etc. After that, the extracted features were used to train our models, and models used were LOGISTIC REGRESSION, LINEAR SVM, XGBOOST. We observed that basic machine learning models like Logistic Regression and Linear SVM gave logloss 0.52 and 0.48 respectively but ensemble model like XGBoost gave the best performance. XGBoost model achieved highest accuracy with lowest logloss 0.35.

1. INTRODUCTION

1.1 Quora Question Pair Similarity

Over 100 million people visit Quora every month, so it's no surprise that many people ask similarly 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 values canonical questions because they provide a better experience to active seekers and writers, and offer more value to both of these groups in the long term. So main aim of project is to predict whether pair of questions are similar or not. This could be useful to instantly provide answers to questions that have already been answered. With the growing repository of the knowledge base, there is a need for Quora to preserve the trust of the users, maintain the content quality, by discarding the junk, duplicate and insincere information.

1.2 Motivation Behind the project:

Identifying semantically identical questions on, Question and Answering(Q&A) social media platforms like Quora is exceptionally significant to ensure that the quality and the quantity of content are presented to users, based on the intent of the question and thus enriching overall user experience.

Detecting duplicate questions is a challenging problem because natural language is very expressive, and a unique intent can be conveyed using different words, phrases, and sentence structuring.

Due to the difficulty of this task, we think the Quora dataset poses an interesting problem.

1.3 Problem Statement :

- Identify the questions on Quora that are duplicates of questions that have been asked before..
- We are having the problem of binary classification by predicting whether a pair of questions are duplicates or not.

1.4 Real world/Business Objectives and Constraints :

- The cost of a mis-classification can be very high.
- You would want a probability of a pair of questions to be duplicates so that you can choose any threshold of choice.
- No strict latency concerns.
- Interpretability is partially important.

2. Critical Analysis of Research Papers read:

I. "Classification of Semantically Similar Question Pairs using Machine Learning" by Nikhar Desai, Anand Mahendran, Int. J. Com. Dig. Sys. 9, No.3 (May-2020)

-In this paper, feature extraction is done of some basic, advance, fuzzy, distance and TFIDF weighted average word2vec features. Machine learning models like Logistic Regression and Linear SVM gave decent accuracy of 74 and 73 percent respectively. But, XGBoost model achieved highest accuracy of 83.7 %.

II. "Semantic string similarity for quora question pairs", Shashank Pathak, Ayush Sharma, Shashank Shekhar Shukla. IJASEAT, Volume-6, Issue-4, Oct.-2018 .

- In their paper, they used some basic and fuzzy features along with several word2vec based distance features. Word2vec takes as its input a large corpus of text and produces a vector space, typically of several hundred dimensions, with each unique word in the corpus being assigned a corresponding vector in the space. They have used combination of different feature sets as part of feature engineering.

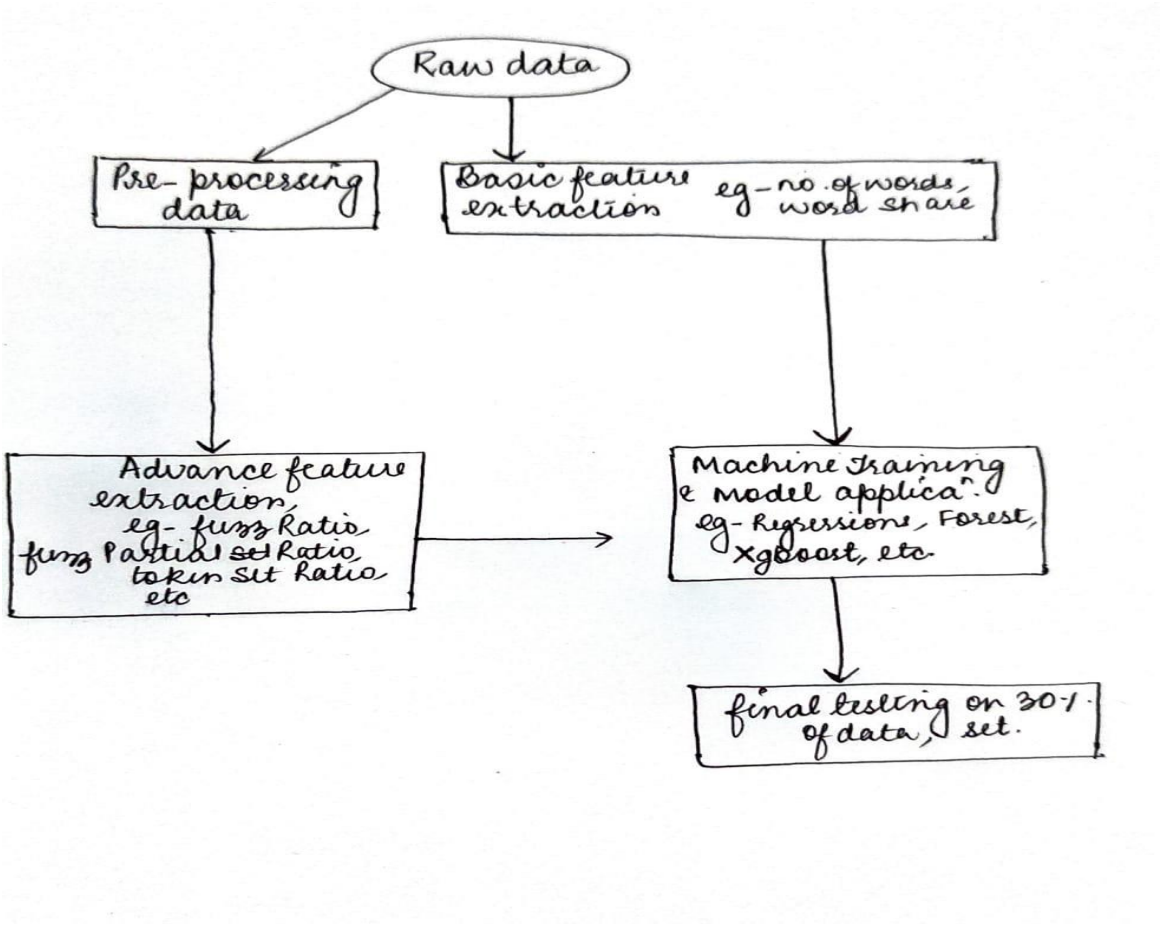
III. "Identifying Semantically Duplicate Questions Using Data Science Approach: A Quora Case Study", Navedanjum Ansari,Rajesh Sharma

-They proposed some machine learning models that gives higher accuracy than some baseline deep learning models like LSTM and Siamese LSTM. They also experimented with deep learning models that gives 85.82 percent accuracy. Their best machine learning model gives accuracy of 82.44 percent based on character level TFIDF features. They have used Glove embedding model to convert word to vector.

IV. "Natural Language Understanding with the Quora Question Pairs Dataset" by Lakshay Sharma

-They experimented with unigram, bigram and trigram out of which trigram achieved higher accuracy then unigram and bigram. Their best performing model is a Continuous Bag of Words neural network.

3. Overall Design of project:



Performance Metric:

Will imply metrics:

- log-loss
- Binary Confusion Matrix, Precision Matrix, Recall Matrix

4. Data Overview:

Our dataset consists of **404290 data points**. Train.csv contains 6 columns:

Id : Unique identifier of question pair set

qid1 : Question-1 Unique ID

qid2 : Question-2 Unique ID

question1 : Question1 is the actual question string

question2 : Question2 is the actual question string

is_duplicate : Binary value used to describe if the question pair is duplicate or not (1 or 0)

Splitted data into train and test with 70% and 30%.

DataSet:

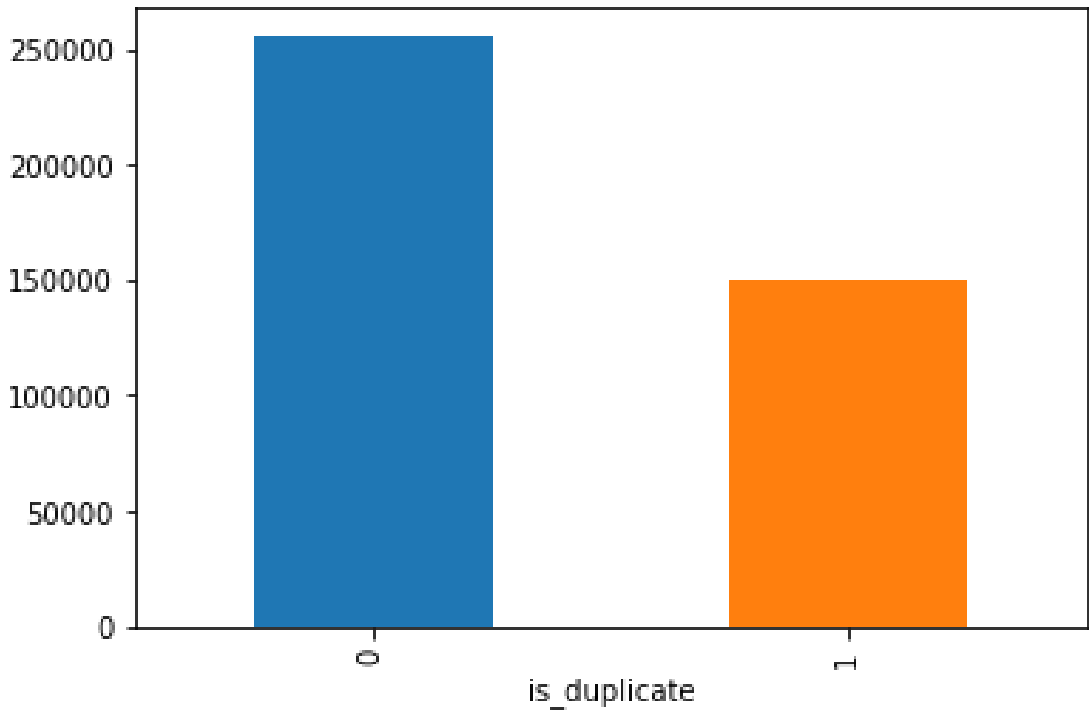
Number of Total Data Points : 404290

	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

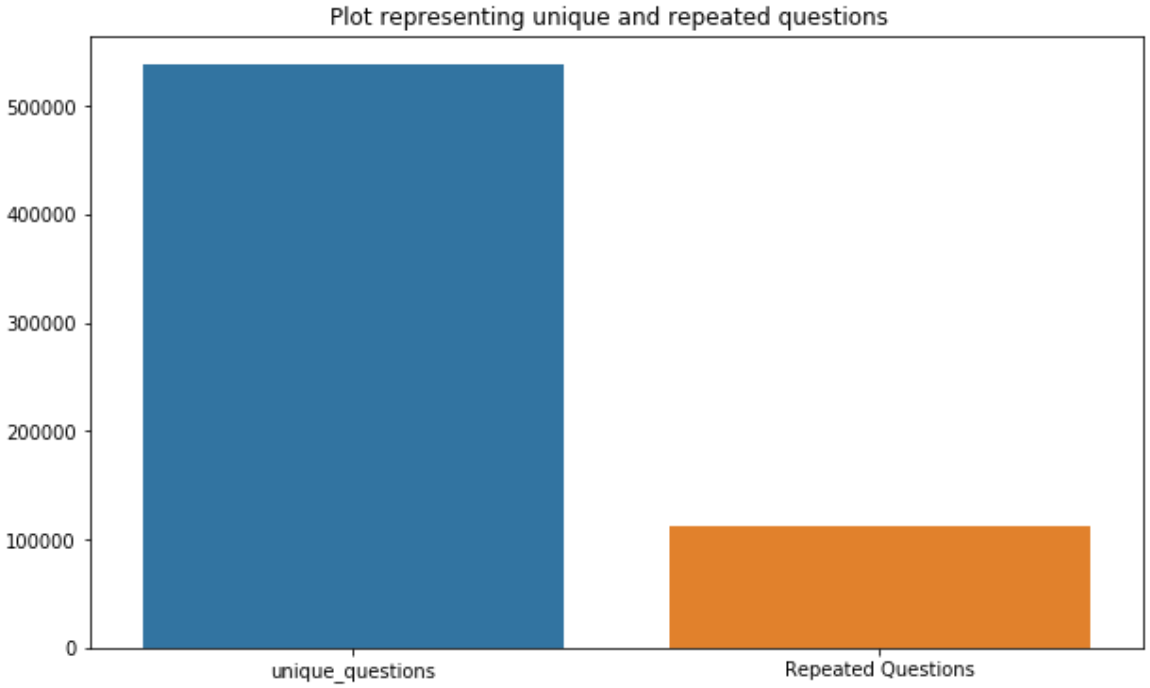
4.1 Some Analysis on Dataset done by us:

Analysing our dataset before making any observations from it was extremely important, since we did not want to train our models on a biased dataset:

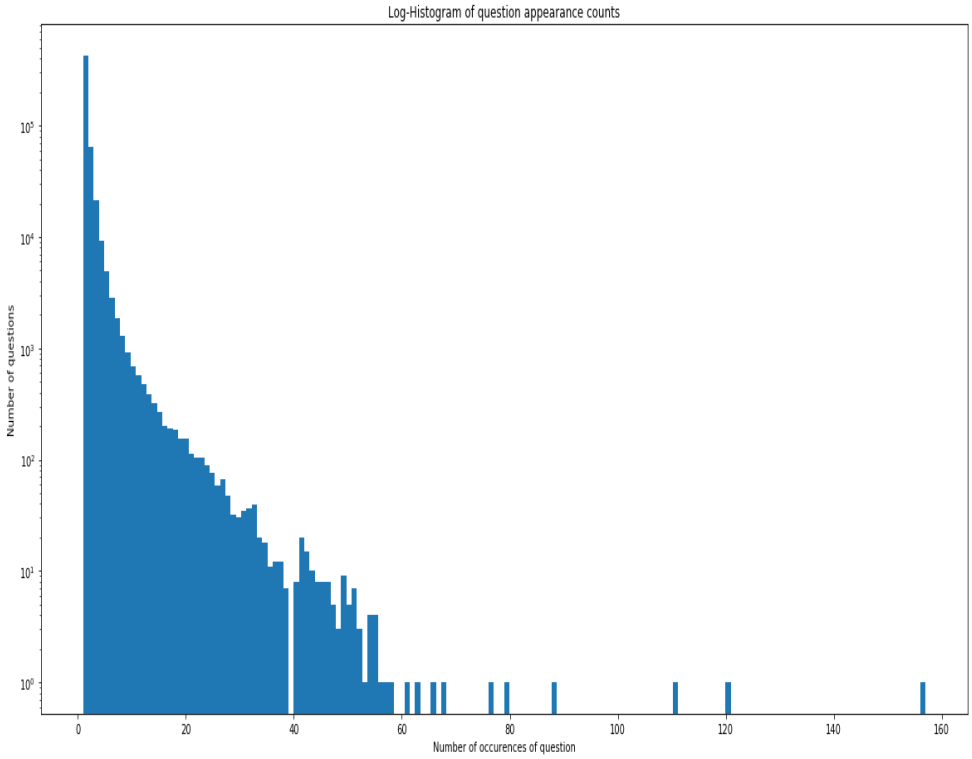
- **Distribution of data points among output classes**



- **Number of unique questions**

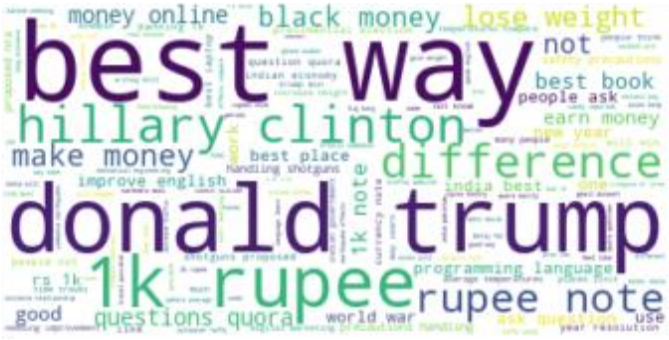


- **Number of occurrences of each question**



 There is no duplicate entry in our dataset. It had only 2 Null values, which were filled with space.

 Wordcloud for similar questions



 Wordcloud for dissimilar questions

4.2 Preprocessing:

First of all, we replaced the different representations of a same term in different questions to bring the dataset on a common platform. For instance, we replaced numerical terms like 1000 to 1k and so on. We also replaced special characters like '$' and '%' with their standard names, and contracted words having apostrophe with their uncontracted equivalents. We used Porter Stemmer to map the group of stems on to the same stem. It is based on the idea that the suffixes in the English language are made up of a combination of smaller and simpler suffixes. BeautifulSoup was used to remove the html tags like head,body,title,bold etc.

5. Feature Extraction:

We derived some features from questions like no of common words, word share and some distances between questions with the help of word vectors. Will discuss those below.

5.1 Basic Features - Extracted some features before cleaning of data as below.

- **freq_qid1** = Frequency of qid1's
- **freq_qid2** = Frequency of qid2's
- **q1len** = Length of q1
- **q2len** = Length of q2
- **q1_n_words** = Number of words in Question 1
- **q2_n_words** = Number of words in Question 2
- **word_Common** = (Number of common unique words in Question 1 and Question 2)
- **word_Total** =(Total num of words in Question 1 + Total num of words in Question 2)

- **word_share** = (word_common)/(word_Total)
- **freq_q1+freq_q2** = sum total of frequency of qid1 and qid2
- **freq_q1-freq_q2** = absolute difference of frequency of qid1 and qid2

freq_qid1	freq_qid2	q1_len	q2_len	q1_n_words	q2_n_words	word_common	word_total	word_share	freq_q1+freq_q2
1	1	66	57	14	12	10.0	23.0	0.434783	2
4	1	51	88	8	13	4.0	20.0	0.200000	5
1	1	73	59	14	10	4.0	24.0	0.166667	2
1	1	50	65	11	9	0.0	19.0	0.000000	2
3	1	76	39	13	7	2.0	20.0	0.100000	4

5.2 Advanced Features – We did some preprocessing of texts and extracted some other features. Some of the definitions used are mentioned below.

Token- You get a token by splitting sentence by space ,

Stop_Word - stop words as per NLTK,

Word -A token that is not a stop_word.

- **cwc_min** = common_word_count / (min(len(q1_words), len(q2_words))
- **cwc_max** = common_word_count / (max(len(q1_words), len(q2_words))
- **csc_min** = common_stop_count / (min(len(q1_stops), len(q2_stops))
- **csc_max** = common_stop_count / (max(len(q1_stops), len(q2_stops))
- **ctc_min** = common_token_count / (min(len(q1_tokens), len(q2_tokens))
- **ctc_max** = common_token_count / (max(len(q1_tokens), len(q2_tokens))

- **last_word_eq** = Check if Last word of both questions is equal or not (int(q1_tokens[-1] == q2_tokens[-1]))
- **first_word_eq** = Check if First word of both questions is equal or not (int(q1_tokens[0] == q2_tokens[0]))
- **abs_len_diff** = abs(len(q1_tokens) - len(q2_tokens))
- **mean_len** = (len(q1_tokens) + len(q2_tokens))/2
- **fuzz_ratio** = How much percentage these two strings are similar, measured with edit distance.
- **fuzz_partial_ratio** = if two strings are of noticeably different lengths, we are getting the score of the best matching lowest length substring.
- **token_sort_ratio** = sorting the tokens in string and then scoring fuzz_ratio.
- **longest_substr_ratio** = len(longest common substring) / (min(len(q1_tokens), len(q2_tokens))

cwc_min	cwc_max	csc_min	csc_max	...	ctc_max	last_word_eq	first_word_eq	abs_len_diff	mean_len	token_set_ratio
0.999980	0.833319	0.999983	0.999983	...	0.785709	0.0	1.0	2.0	1.0	100
0.799984	0.399996	0.749981	0.599988	...	0.466664	0.0	1.0	5.0	1.0	86

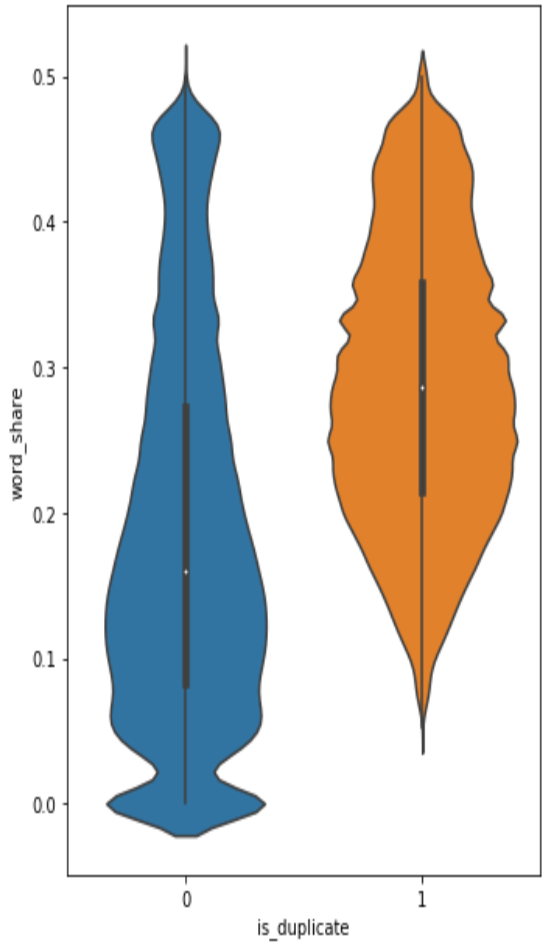
token_sort_ratio	fuzz_ratio	fuzz_partial_ratio	longest_substr_ratio
93	93	100	0.982759
63	66	75	0.596154

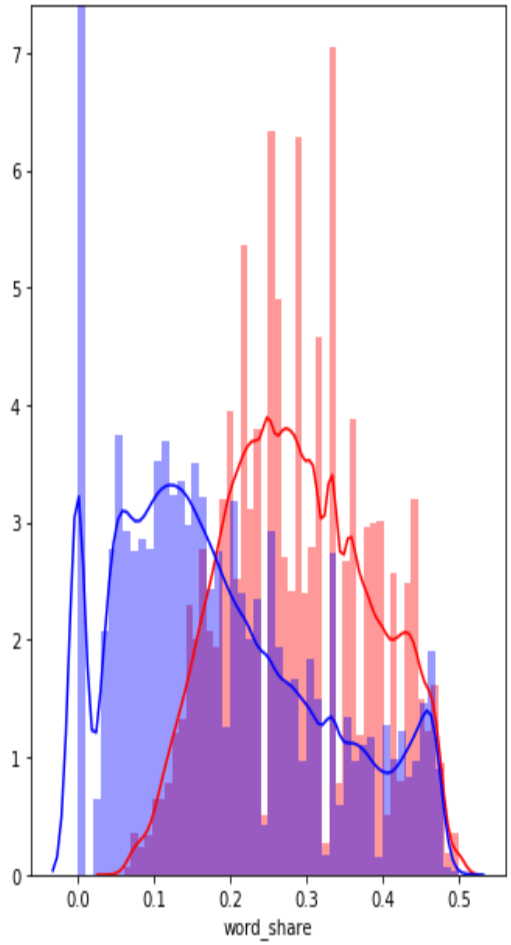
5.3 Extracted Tf-Idf features for this combined question1 and question2 and got 1,2,3 gram features with Train data. Transformed test data into same vector space.

	id	q1_feat_tf	q2_feat_tf
0	0	[73.5751650184393, 81.45066496543586, 72.65540...	[65.52385912835598, 58.07504297234118, 61.1038...
1	1	[96.52657270431519, 53.84101563692093, -1.8918...	[98.48333293199539, 57.39058792591095, 107.548...
2	2	[60.647512156516314, 68.21433615684509, 10.571...	[28.084512442350388, -0.5073451995849609, -8.7...
3	3	[32.649299710989, -39.84746631979942, 50.47549...	[23.353686332702637, -19.681684017181396, 18.3...
4	4	[128.0620268881321, 178.51437878608704, 137.34...	[1.5357239246368408, 27.935986399650574, 62.42...

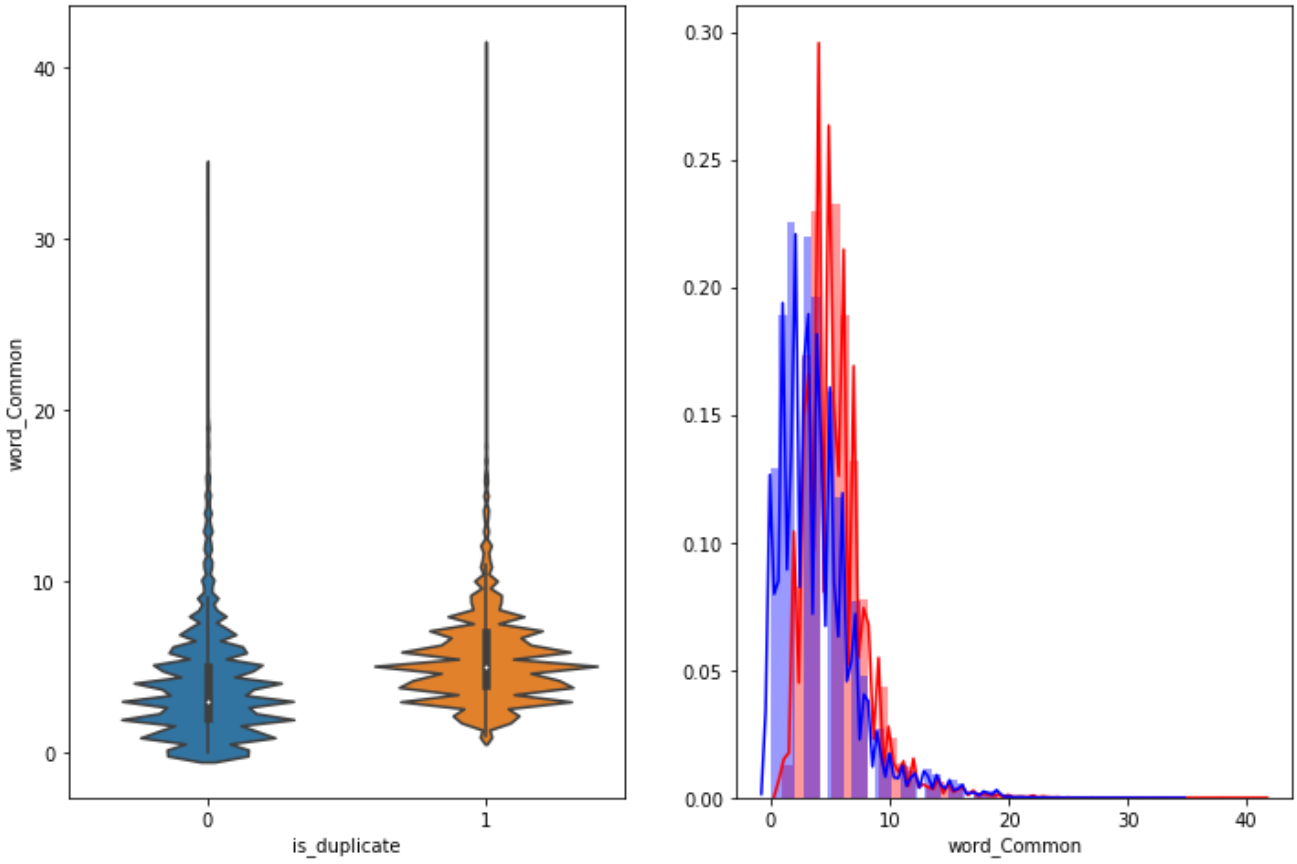
5.4 Some Features' analysis and visualizations:

 word_share - We can check from below that it is overlaping a bit, but it is giving some classifiable score for dissimilar questions.

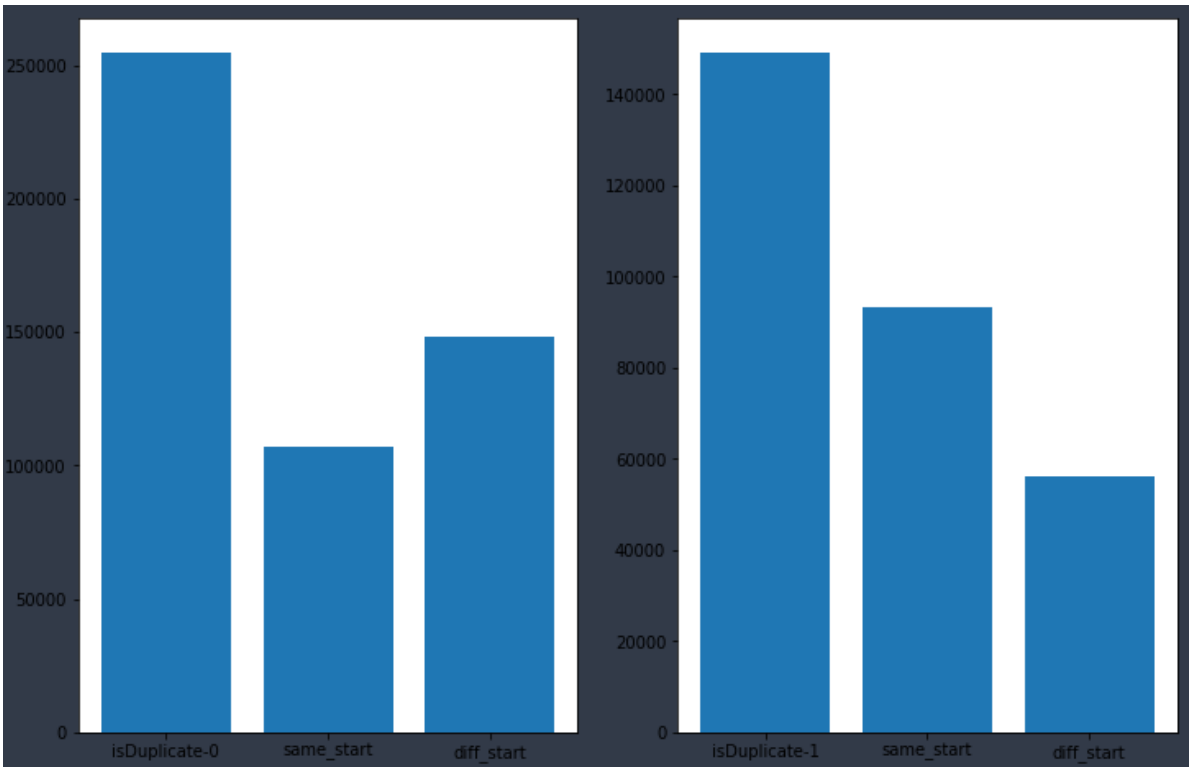




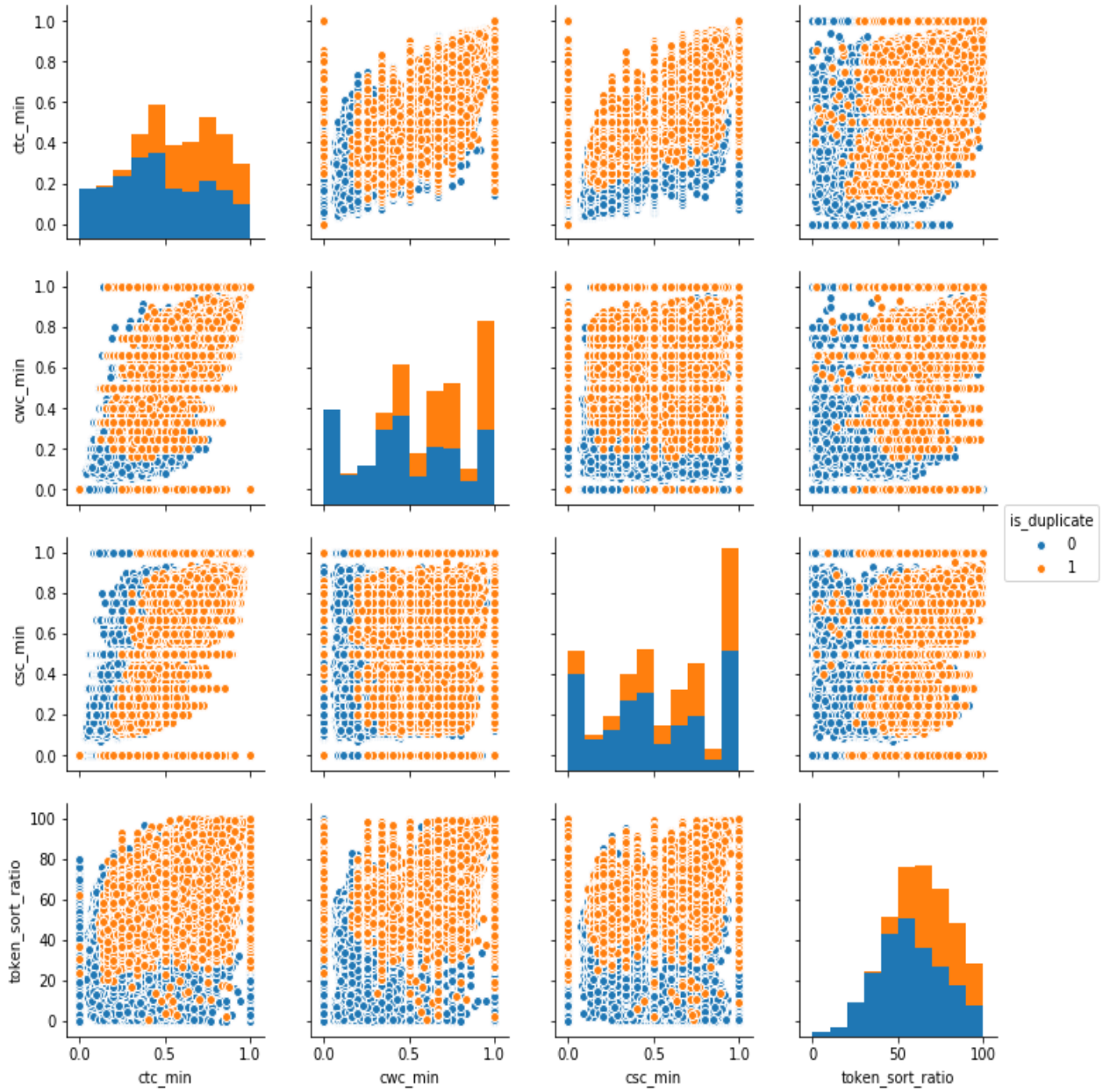
 Word Common - It is almost overlapping and therefore it doesn't provide any understanding.



 First Word Common: Analysing the graph below, the first word common feature cannot be used for any prediction.



 Bivariate analysis of features 'ctc_min', 'cwc_min', 'csc_min', 'token_sort_ratio'. We can observe that we can divide duplicate and non duplicate with some of these features with some patterns.



6 . Machine Learning Models:

Machine learning models which we used to train were:

-Logistic Regression with hyperparameter Tuning

-Linear SVM with Hyperparameter Tuning

-XGBoost

We have used confusion, precision, and recall matrix to evaluate our models. For reference, these matrix are defined as:

Confusion Matrix: It is a summary table showing how good our model is at predicting examples of various classes.

Precision — Also called Positive predictive value
The ratio of correct positive predictions to the total predicted positives.

Recall — Also called Sensitivity, Probability of Detection, True Positive Rate

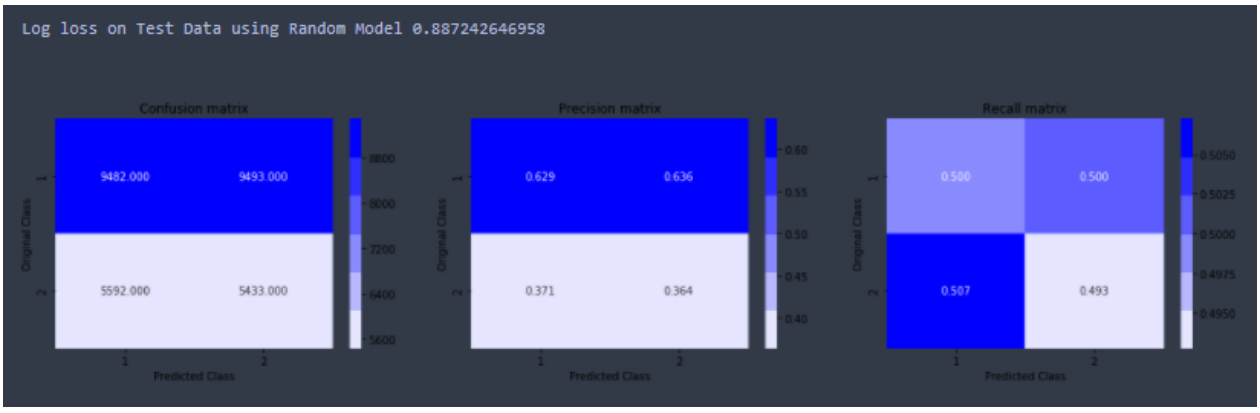
The ratio of correct positive predictions to the total positives examples.

6.1 Random Model(Finding Worst-Case Log Loss)

Code Snippet:

```
# we need to generate 9 numbers and the sum of numbers should be 1
# one solution is to genarate 9 numbers and divide each of the numbers by their sum
# ref: https://stackoverflow.com/a/18662466/4084039
# we create a output array that has exactly same size as the CV data
predicted_y = np.zeros((test_len,2))
for i in range(test_len):
    rand_probs = np.random.rand(1,2)
    predicted_y[i] = ((rand_probs/sum(sum(rand_probs)))[0])
print("Log loss on Test Data using Random Model",log_loss(y_test, predicted_y, eps=1e-15))

predicted_y =np.argmax(predicted_y, axis=1)
plot_confusion_matrix(y_test, predicted_y)
```

Trained a random model to check the worst case log loss and got log loss as 0.887242



6.2 Logistic Regression with Hyperparameter Tuning

Regression means predicting values, and logistic regression helps in predicting categorical values in the form of true/false or 0/1,which is why we used logistic regression.

Given a decision boundary or threshold value, any value above it will result in 1 and values lower it will result in 0.

We performed gradient descent algorithm to minimise the loss with the help of learning rate(alpha) using hypertuning logistic regression.

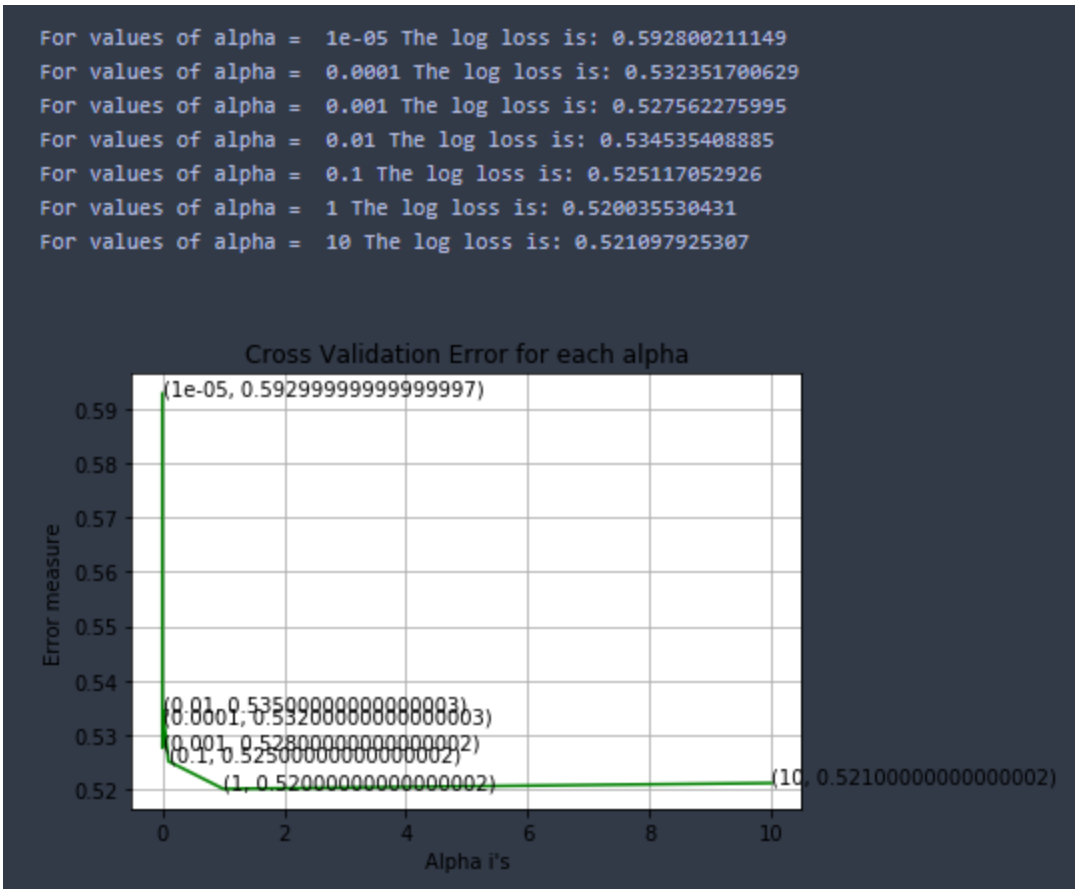
Below are some results and code snippets.

Code Snippet:

```
alpha = [10 ** x for x in range(-5, 2)] # hyperparam for SGD classifier
```

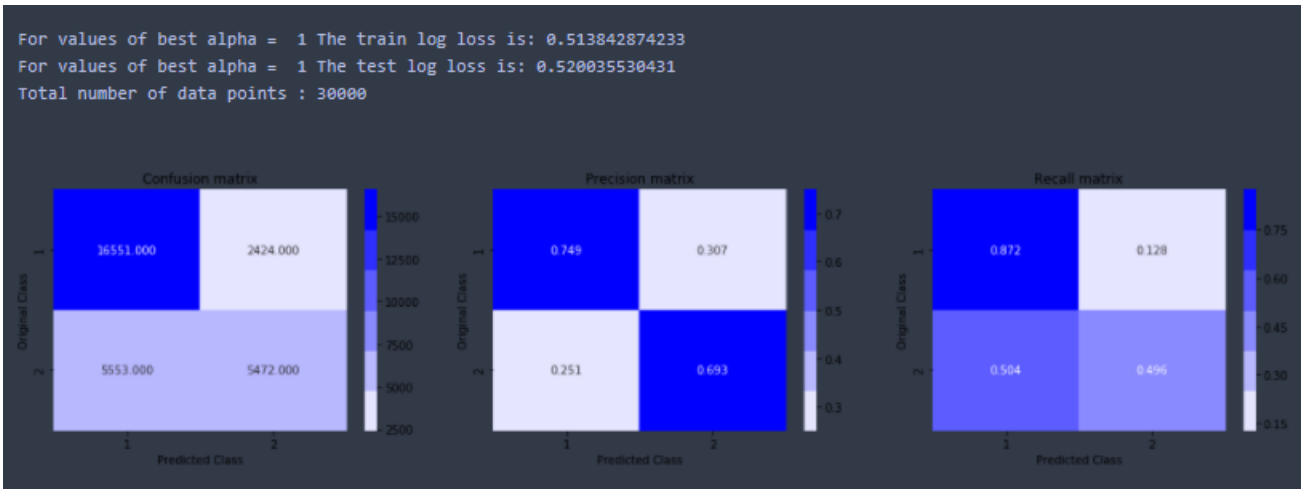
```
log_error_array=[]
for i in alpha:
    clf = SGDClassifier(alpha=i, penalty='l2', loss='log', random_state=42)
    clf.fit(X_train, y_train)
    sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
    sig_clf.fit(X_train, y_train)
    predict_y = sig_clf.predict_proba(X_test)
    log_error_array.append(log_loss(y_test, predict_y, labels=clf.classes_, eps=1e-15))
    print('For values of alpha = ', i, "The log loss is:",log_loss(y_test, predict_y, labels=clf.c
lasses_, eps=1e-15))

fig, ax = plt.subplots()
ax.plot(alpha, log_error_array,c='g')
for i, txt in enumerate(np.round(log_error_array,3)):
    ax.annotate((alpha[i],np.round(txt,3)), (alpha[i],log_error_array[i]))
plt.grid()
plt.title("Cross Validation Error for each alpha")
plt.xlabel("Alpha i's")
plt.ylabel("Error measure")
plt.show()


best_alpha = np.argmin(log_error_array)
clf = SGDClassifier(alpha=alpha[best_alpha], penalty='l2', loss='log', random_state=42)Activate Win
clf.fit(X_train, y_train)                                                    Go to Settings to
sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
```

Learning Rate v/s LogLoss:



The test log loss is: 0.520035530431

Performance Metrics for logistic regression:



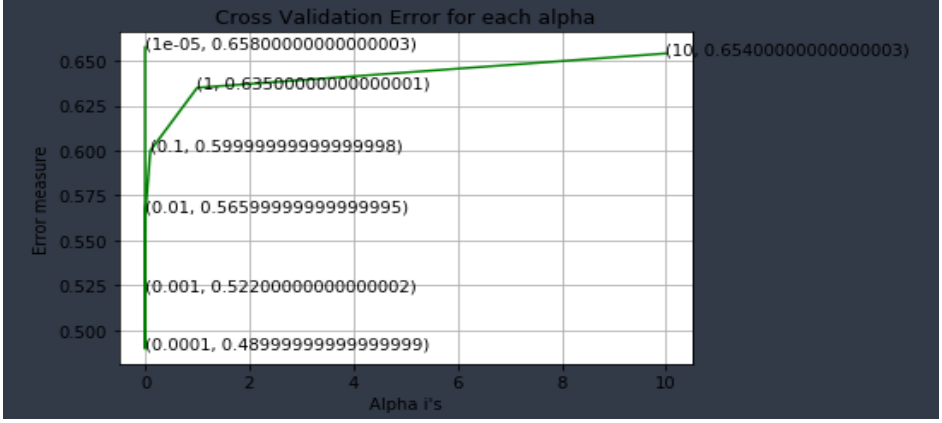
6.3 Linear SVM with Hyperparameter Tuning

Code Snippet:

```
for i in alpha:
    clf = SGDClassifier(alpha=i, penalty='l1', loss='hinge', random_state=42)
    clf.fit(X_train, y_train)
    sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
    sig_clf.fit(X_train, y_train)
    predict_y = sig_clf.predict_proba(X_test)
    log_error_array.append(log_loss(y_test, predict_y, labels=clf.classes_, eps=1e-15))
    print('For values of alpha = ', i, "The log loss is:",log_loss(y_test, predict_y, labels=clf.c
lasses_, eps=1e-15))

fig, ax = plt.subplots()
ax.plot(alpha, log_error_array,c='g')
for i, txt in enumerate(np.round(log_error_array,3)):
    ax.annotate((alpha[i],np.round(txt,3)), (alpha[i],log_error_array[i]))
plt.grid()
plt.title("Cross Validation Error for each alpha")
plt.xlabel("Alpha i's")
plt.ylabel("Error measure")
plt.show()


best_alpha = np.argmin(log_error_array)
clf = SGDClassifier(alpha=alpha[best_alpha], penalty='l1', loss='hinge', random_state=42)
clf.fit(X_train, y_train)
sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
sig_clf.fit(X_train, y_train)
```

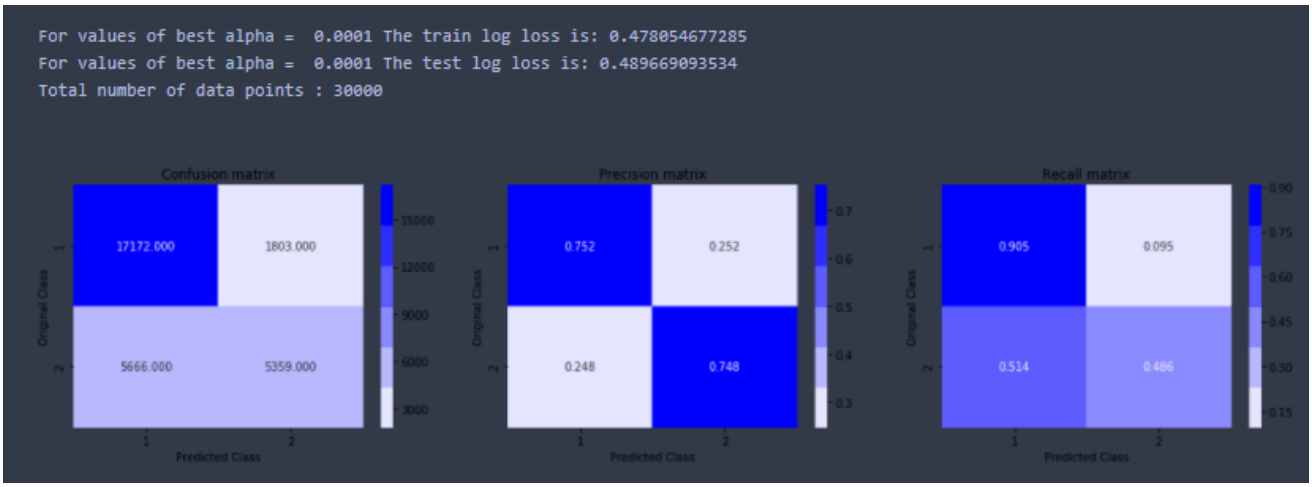
Learning Rate v/s LogLoss:

```
For values of alpha =  1e-05 The log loss is: 0.657611721261
For values of alpha =  0.0001 The log loss is: 0.489669093534
For values of alpha =  0.001 The log loss is: 0.521829068562
For values of alpha =  0.01 The log loss is: 0.566295616914
For values of alpha =  0.1 The log loss is: 0.599957866217
For values of alpha =  1 The log loss is: 0.635059427016
For values of alpha =  10 The log loss is: 0.654159467907
```



The test log loss is: 0.489669093534

Performance Metrics for linear SVM:



6.4 XGBoost

XGBoost grows the tree upto max_depth and then prune backward until the improvement in loss function is below a threshold.

XGBoost implements parallel processing and is **blazingly faster** as compared to gradient boosting machines. Boosting is sequential process but it can still be parallelized. Each tree can be built only after the previous one and each tree is built using all cores. This makes XGBoost a very fast algorithm.

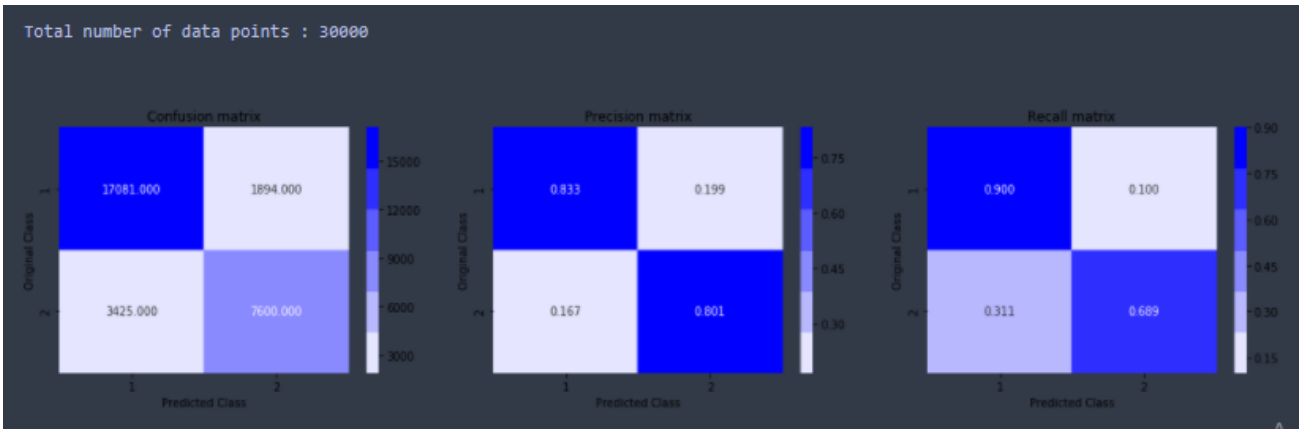
Below are some results and code snippets

Code Snippet:

```
import xgboost as xgb
params = {}
params['objective'] = 'binary:logistic'
params['eval_metric'] = 'logloss'
params['eta'] = 0.02
params['max_depth'] = 4

d_train = xgb.DMatrix(X_train, label=y_train)
d_test = xgb.DMatrix(X_test, label=y_test)

watchlist = [(d_train, 'train'), (d_test, 'valid')]

bst = xgb.train(params, d_train, 400, watchlist, early_stopping_rounds=20, verbose_eval=10)

xgdmat = xgb.DMatrix(X_train,y_train)
predict_y = bst.predict(d_test)
print("The test log loss is:",log_loss(y_test, predict_y, labels=clf.classes_, eps=1e-15))
```

The test log loss is: 0.357054433715

Performance metrics for XGBoost:



In conclusion , XGboost tend to perform much better that the linear model. This indicates that the data is not linearly separable and we need a complex non linear model like XGboost.

- For below table BF - Basic features, AF - Advanced features, DF - Distance Features including WMD.

Model	Features Used	Log Loss
Logistic Regression	BF + AF	0.5200355
Linear SVM	BF + AF	0.4896690
XGBoost	BF + AF	0.3570544

7. CONCLUSION

The basic models like Logistic Regression and SVM gave logloss of 0.52 and 0.48 respectively. But ensemble XGBoost gave lower logloss of 0.35 giving much higher accuracy . The best model in this project is XGBoost. Future work on this problem includes adding some more features to improve accuracy of XGBoost model and also experimenting with some deep learning algorithms to achieve better accuracy than ensemble model like XGBoost.

REFERENCES

[1] Shashank Pathak, Ayush Sharma, Shashank Shekhar Shukla. "Semantic string similarity for quora question pairs", IJASEAT, Volume-6, Issue-4, Oct.-2018 .
[2] Aniket Shenoy and Shashi Shankar. "Identifying Quora question pairs having the same intent".
[3] Lakshay Sharma, Laura Graesser, Nikita Nangia, Utku Evci . "Natural language understanding with the quora question pairs dataset", arXiv, 2019.
[4] Navedanjum Ansari and Rajesh Sharma, "Identifying semantically duplicate questions using data science approach: a quora case study", ACM Conference, New York, NY, USA
[5] . "Semantic string similarity for quora question pairs", Shashank Pathak, Ayush Sharma, Shashank Shekhar Shukla. IJASEAT, Volume-6, Issue-4, Oct.-2018 .