**PROJECT REPORT**

## **QUORA QUESTION PAIR SIMILARITY**

## A MACHINE LEARNING PROJECT

**AYUSH**

**BTECH/10084/20**

**AN ENGINEERING STUDENT AT**

### **BIRLA INSTITUTE OF TECHNOLOGY, MESRA**

**DESCRIPTION OF THE PROBLEM**

*(Source -* [*https://www.kaggle.com/c/quora-question-pairs*](https://www.kaggle.com/c/quora-question-pairs)*)*

Quora is a place to gain and share knowledge—about anything. It’s a platform to ask questions and connect with people who contribute unique insights and quality answers. This empowers people to learn from each other and to better understand the world.

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.

**Objective:**

Identify whether the question that is asked on Quora is duplicate of questions that have already been asked.

**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.

**DATA OVERVIEW:**

* The data is in a file named as ‘train.csv’.
* The dataset contains 5 columns: qid1, qid2, question1, question2, is\_duplicate
* There are 404,290 rows in train.csv

**Overview of the solution**

The objective of this project is to classify whether a given question is a duplicate of any existing question on Quora or not. Hence, the problem is a binary classification problem.

In machine learning we have several existing models which are capable of doing binary classification. In this project some of the major models such as Logistic Regression, Support Vector Machines, Random Forest, XGBoost, etc are used.

The whole procedure is divided into three sub problems:

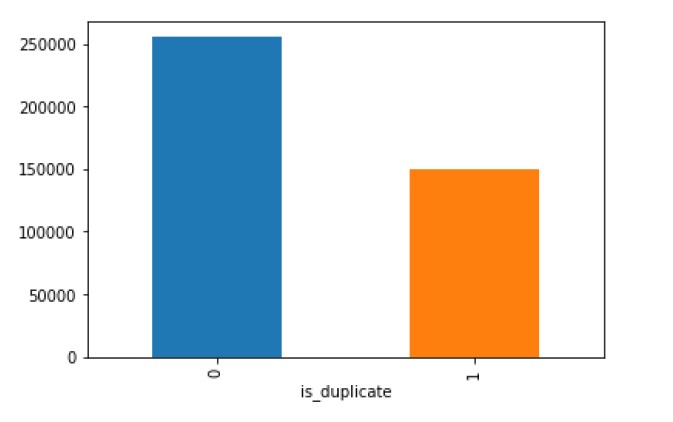
1. Analysing, cleaning and pre-processing of the data.
2. Finding correct features and training a model on them.
3. Testing the model and finding the correct hyperparameters of the respective models.

Before proceeding with anything all the libraries which will be used in this project are imported.

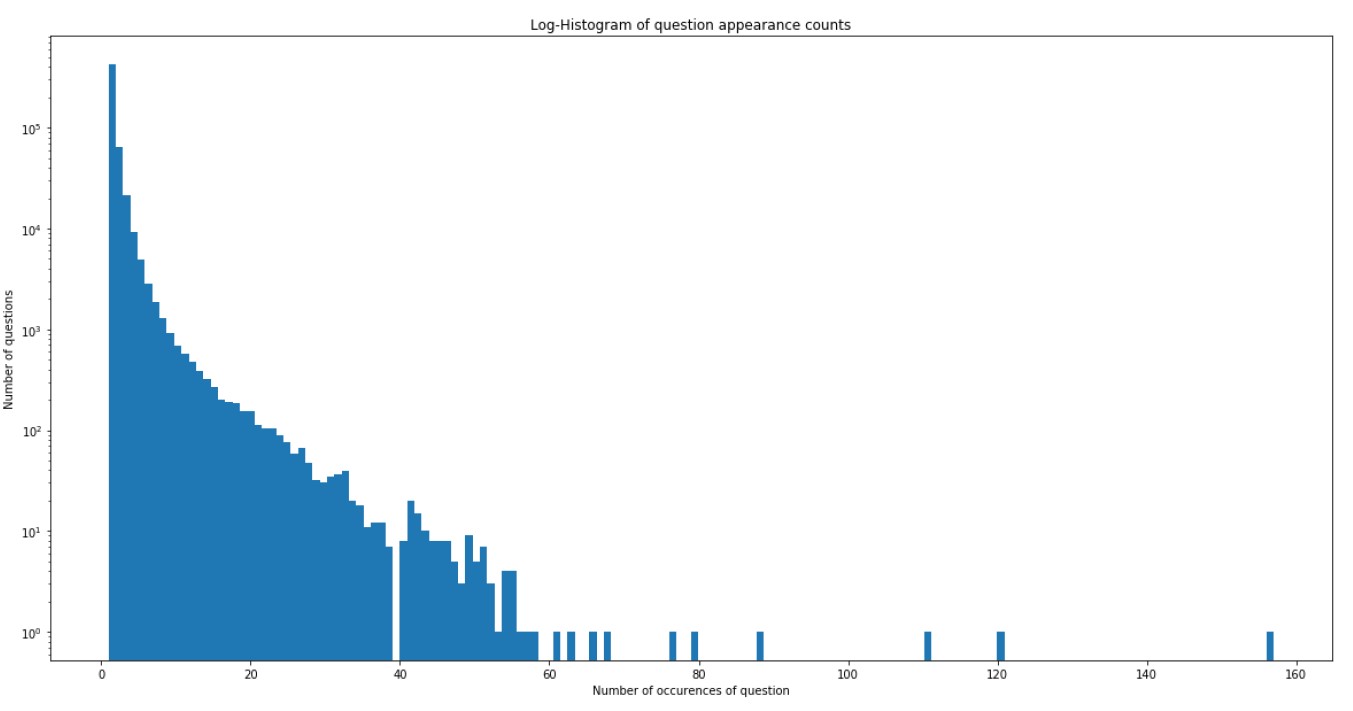
**DATA ANALYSIS**

Before deploying any machine learning model, it is very necessary to understand the nature of the dataset on which one is going to work. Following are some of the insights of the data:

* Number of data points: 404290
* Distribution of data points among output classes. *There are two output classes namely 1 (is duplicate) and 2 (not duplicate)* . 68.08% question pairs are not similar while 36.92% question pairs are similar.



* Total number of question pairs for training: 404290
* Total number of Unique Questions are: 537933
* Number of unique questions that appear more than one time: 111780 (20.77953945937505%)
* Max number of times a single question is repeated: 157
* Histogram of number of occurrence of a question:



* There are two rows with null values in question2. The null values are replaced by “ “.

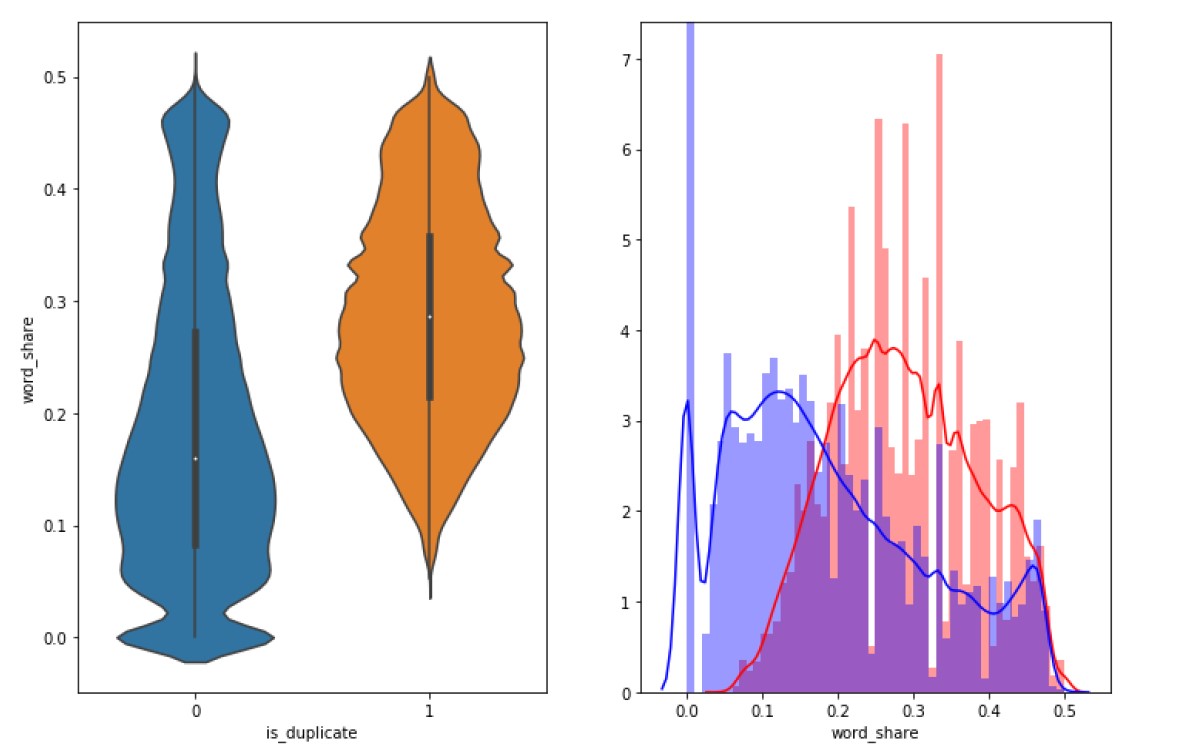
**Basic Feature Extraction:**

Following basic features are used in this project:

* **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

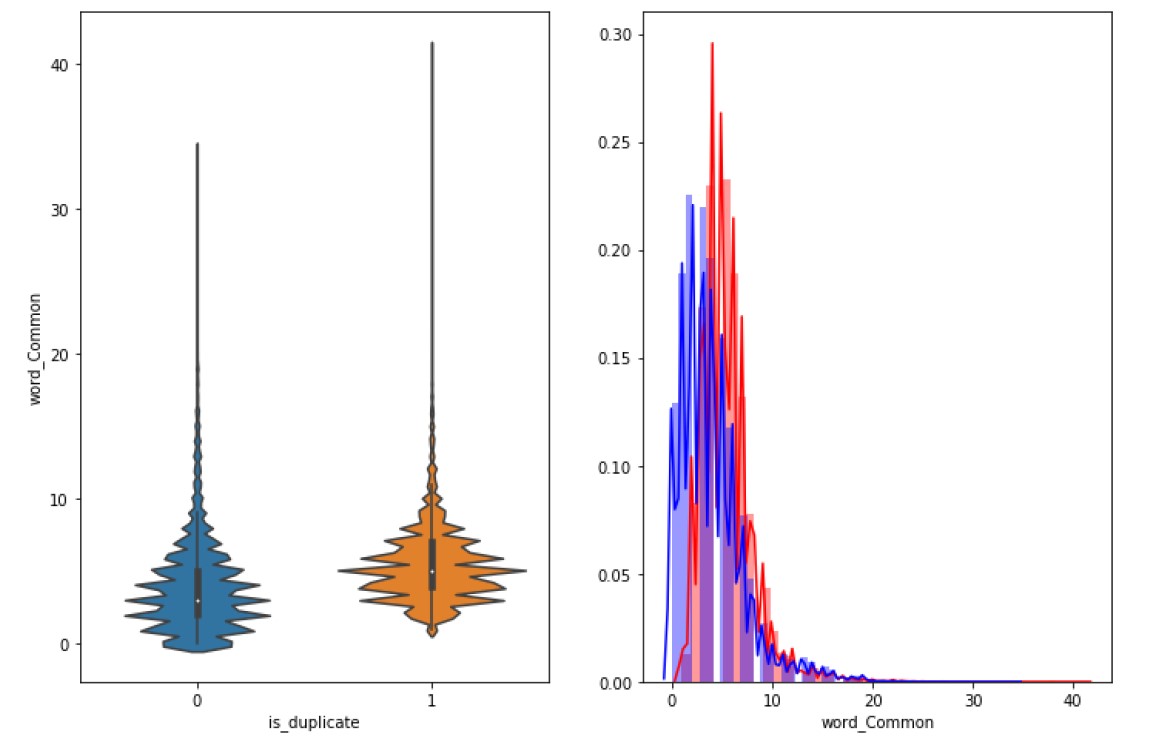
**Analysis on some of the basic features:**

* **word\_share** feature.



It is clearly visible that there is some distinction between two output classes. Although there is a significant area of overlapping but still **word\_share** helps in classifying the data up to a certain extent.

* **Word\_count** feature



Here both the output classes are overlapping a lot. It is nearly impossible to do any classification by using word\_count as the parameter. Hence this feature is alone not that useful.

**PREPROCESSING OF TEXT**

We will be using following preprocessing of text:

* Removing html tags
* Removing punctuations
* Stemming
* Removing Stopwords
* Expanding contractions etc.

**ADVANCED FEATURE EXTRACTION:**

Definition:

* **Token**: You get a token by splitting sentence a space
* **Stop\_Word** : stop words as per NLTK.
* **Word** : A token that is not a stop\_word

Features:

* **cwc\_min** : Ratio of common\_word\_count to min lenghth of word count of Q1 and Q2

cwc\_min = common\_word\_count / (min(len(q1\_words), len(q2\_words))

* **cwc\_max** : Ratio of common\_word\_count to max lenghth of word count of Q1 and Q2

cwc\_max = common\_word\_count / (max(len(q1\_words), len(q2\_words))

* **csc\_min** : Ratio of common\_stop\_count to min lenghth of stop count of Q1 and Q2

csc\_min = common\_stop\_count / (min(len(q1\_stops), len(q2\_stops))

* **csc\_max** : Ratio of common\_stop\_count to max lenghth of stop count of Q1 and Q2

csc\_max = common\_stop\_count / (max(len(q1\_stops), len(q2\_stops))

* **ctc\_min** : Ratio of common\_token\_count to min lenghth of token count of Q1 and Q2

ctc\_min = common\_token\_count / (min(len(q1\_tokens), len(q2\_tokens))

* **ctc\_max** : Ratio of common\_token\_count to max lenghth of token count of Q1 and Q2

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

last\_word\_eq = int(q1\_tokens[-1] == q2\_tokens[-1])

* **first\_word\_eq** : Check if First word of both questions is equal or not

first\_word\_eq = int(q1\_tokens[0] == q2\_tokens[0])

* **abs\_len\_diff** : Abs. length difference

abs\_len\_diff = abs(len(q1\_tokens) - len(q2\_tokens))

* **mean\_len** : Average Token Length of both Questions

mean\_len = (len(q1\_tokens) + len(q2\_tokens))/2

* **fuzz\_ratio** : https://github.com/seatgeek/fuzzywuzzy#usage http://chairnerd.seatgeek.com/fuzzywuzzy-fuzzy-string-matching-inpython/
* **fuzz\_partial\_ratio** : https://github.com/seatgeek/fuzzywuzzy#usage http://chairnerd.seatgeek.com/fuzzywuzzy-fuzzy-string-matchingin-

python/

* **token\_sort\_ratio** : https://github.com/seatgeek/fuzzywuzzy#usage http://chairnerd.seatgeek.com/fuzzywuzzy-fuzzy-string-matchingin-

python/

* **token\_set\_ratio** : https://github.com/seatgeek/fuzzywuzzy#usage http://chairnerd.seatgeek.com/fuzzywuzzy-fuzzy-string-matchingin-

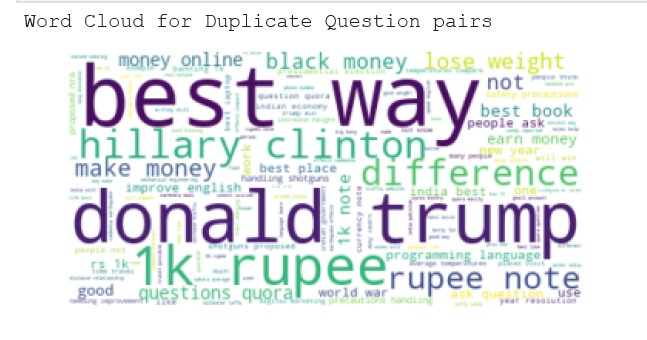
python/

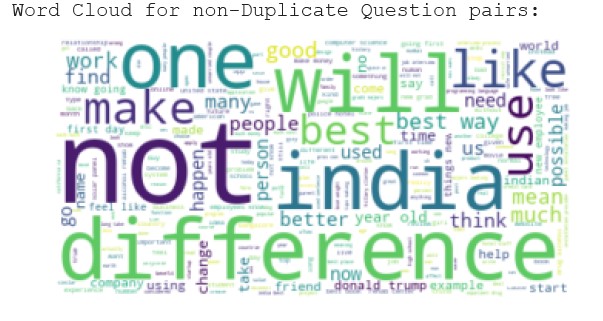
* **longest\_substr\_ratio** : Ratio of length longest common substring to min lenghth of token count of Q1 and Q2

longest\_substr\_ratio = len(longest common substring) / (min(len(q1\_tokens), len(q2\_tokens))

Like basic features it is also necessary to do some analysis on advanced features to know whether they are helpful for classification or not.

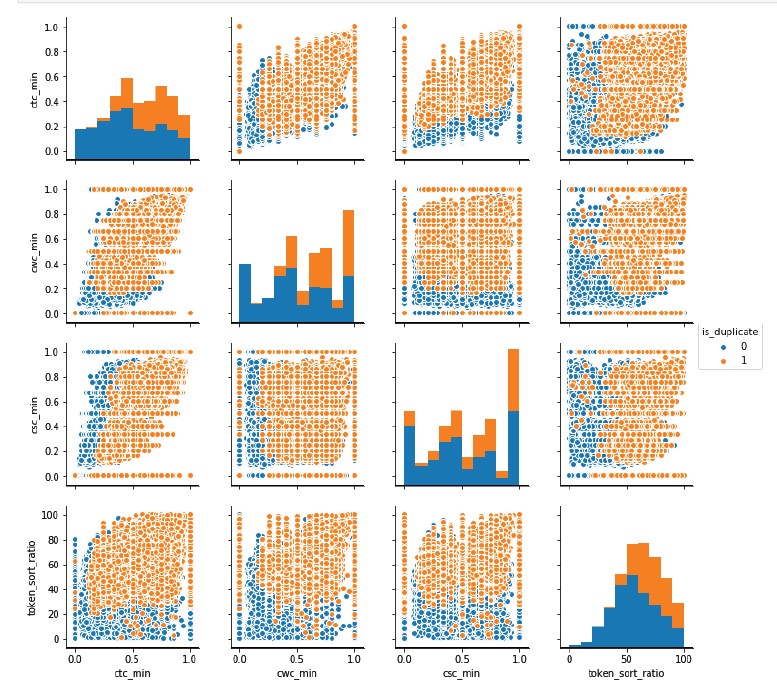
**Word Cloud of both output classes**





In word cloud the size of the word represents its frequency in the question pairs of that output class. Word cloud is not looking of much help as there are some common words there too.

**PAIR PLOTS OF ADVANCED FEATURES:**

****

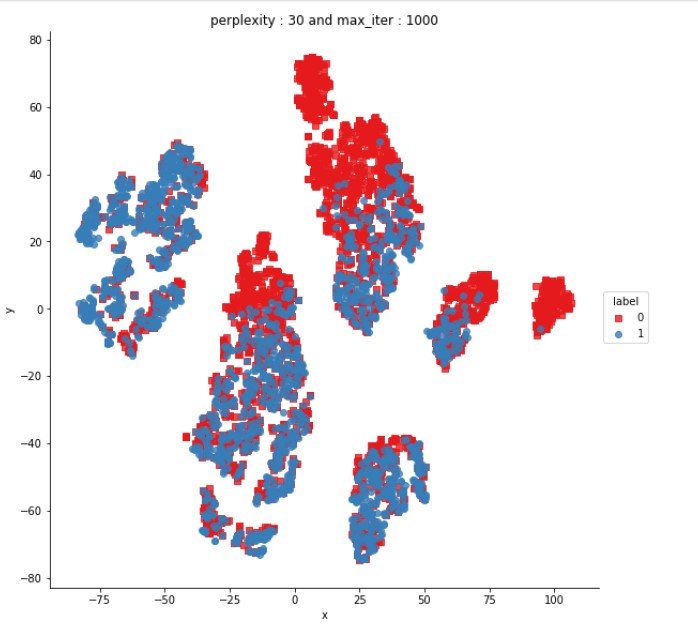
The pair plots give us the idea that many pair of features are helpful in classifying. This shows that the features which we took are working positively.

**TSNE**

Although since we have used more than a couple of features and all of them will be applied at the same time so it becomes necessary to see the overall behaviour of all the features. The visual representations are not possible beyond 3D so it is necessary to do dimensionality reduction. For dimensionality reduction we have used **TSNE.**

The perplexity is 30 and max iterations is 1000.

The following results are obtained by using TSNE.



**TEXT TO VECTOR CONVERSION**

Since it is necessary to convert any text data into numbers first to deploy any machine learning model on it, we have to convert the questions into numbers. Since a single number cannot carry the semantic meaning of the text so we will have to use vectors.

We already have some existing methods like **word2vec** to do this. Word2vec converts the text into vectors in such a way that the semantic meaning of text is preserved.

**Tf-Idf word2vec** is an advanced version in which each number of the vector is multiplied by its tf-idf value.

**In this project, we have calculated tf-idf in two different ways. The first will be called as tf-idfA and second will be called as tf-idfB.**

**Tf-idf A**

We make a list which contains question1 and question2. First the column of question1 is copied to that list then column of question2 is appended to the same list. We apply tf-idf on that.

**Tf-idf B**

We make a list. A single element of that list contains the text which is the sum of one question from question1 and its corresponding question from question2. We apply tf-idf on that.

In this project all the machine learning models are applied by two set of features. First set of features were obtained by tf-idfA dataset and the second set of features which were obtained by tf-idfB dataset.

After applying tf-idf we have converted text into vector using GLOVE. Glove is similar to tf-adf word2vec. It is easier to use hence GLOVE is preferred in this project.

After doing all the data preprocessing and cleaning, the following is the nature of the data.

* Number of features in nlp dataframe : 17
* Number of features in preprocessed dataframe : 12
* Number of features in question1 w2v dataframe : 384
* Number of features in question2 w2v dataframe : 384
* Number of features in final dataframe : 797

**MACHINE LEARNING MODELS**

To deploy any model, we first have to divide the dataset into train and test dataset. We have a significant number of data so we will divide the data into 70:30 ratio. We could have done it 80:20 also.

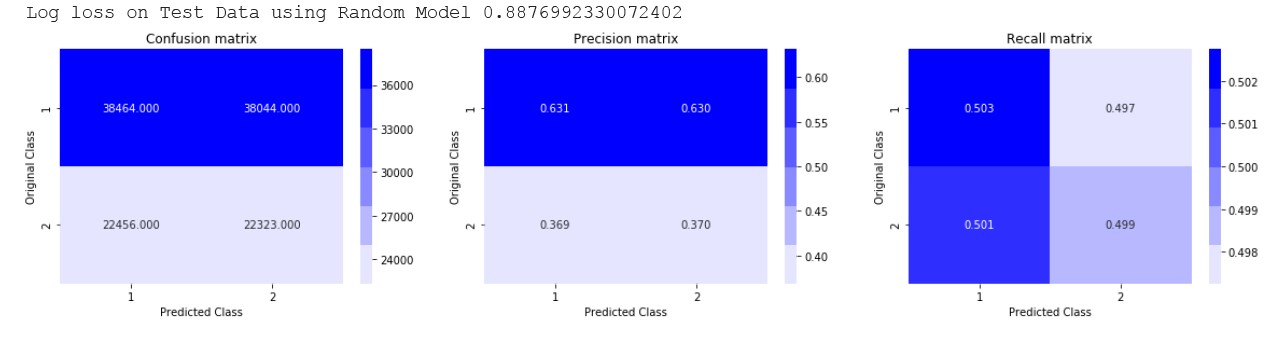
Every machine learning model needs a metric to be measured or analysed. For this project the metrics used are:

* Log Loss
* Confusion Matrix

**RANDOM MODEL**

The need of building a random model is that we need a reference point to measure the parameters of our built model. Random model is the dumbest model possible. It represents the worst case.

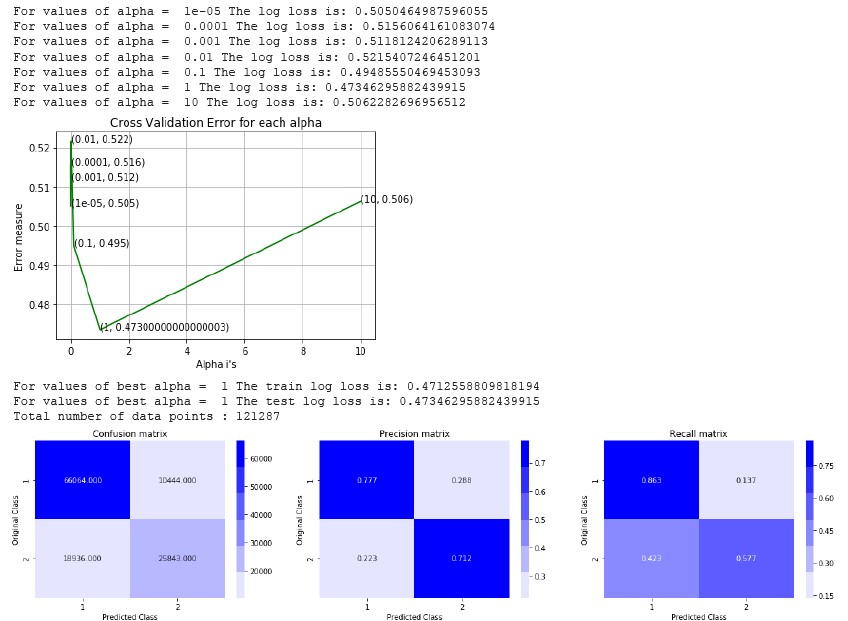
Following are the parameters of the random model:

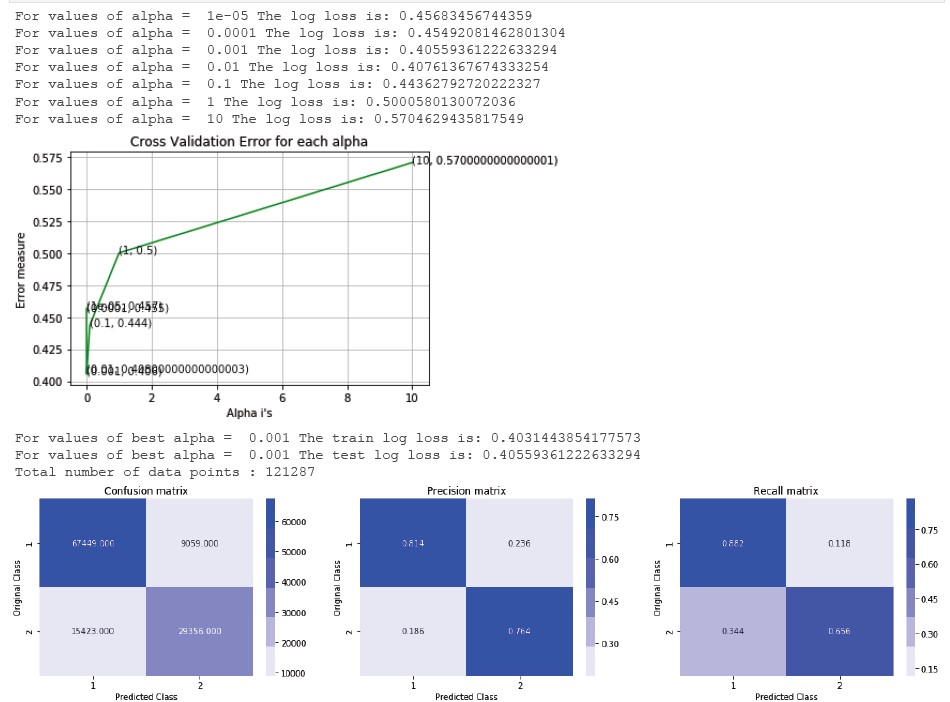


The **log loss on test data using Random Model is 0.8876992**. It shows that log loss of our built model must be less that 0.8876992. The lesser it is from log loss of random model the better the model is.

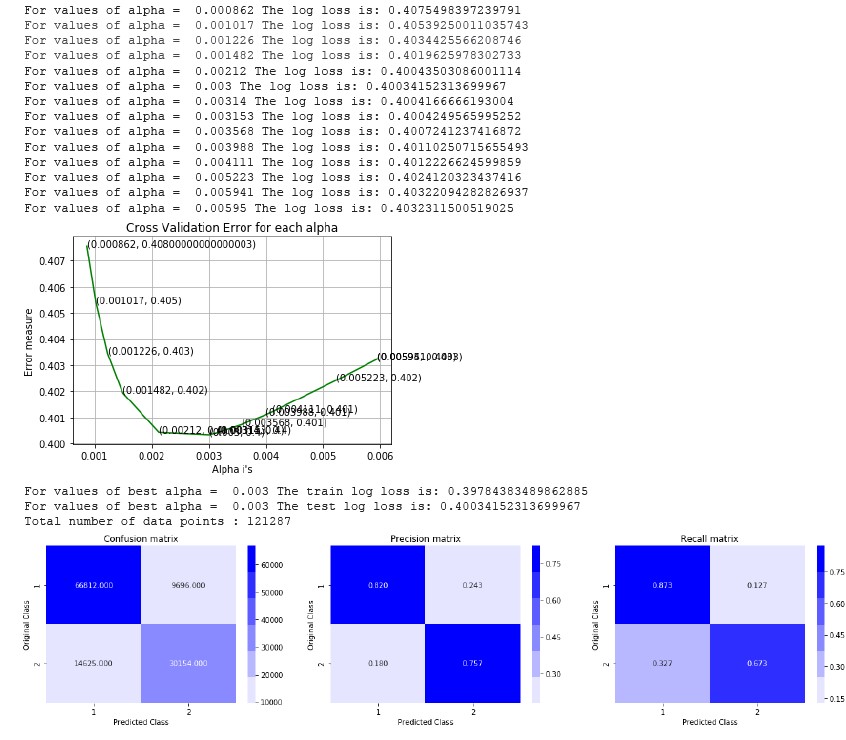
**LOGISTIC REGRESSION**

We deploy the logistic regression models for different values of **alpha**. We have used **l2 regulariser**. The aim is to find the alpha with least value of log loss. The model at that very alpha should perform well on train and test data both. We also plot the graph of cross validation error and try to see whether we get the least error at that particular value of alpha or not. We also plot confusion matrix.

Following are the results after first attempt.

Since **gradient descent** is sensitive to feature scaling so repeat this process after doing feature scaling. Following are the results after feature scaling. 

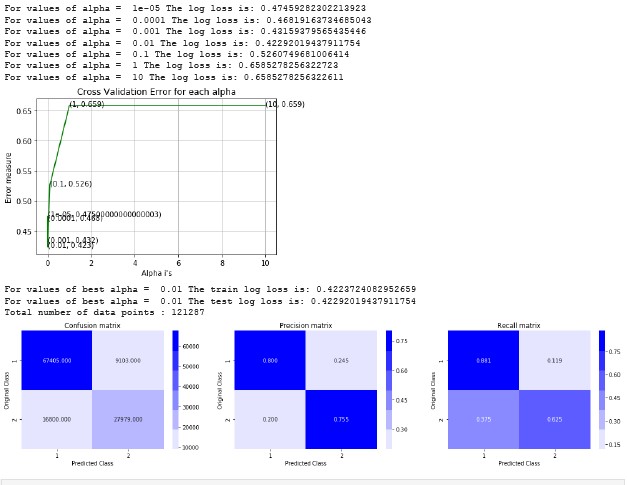
Now repeating the process using random search cross validation.

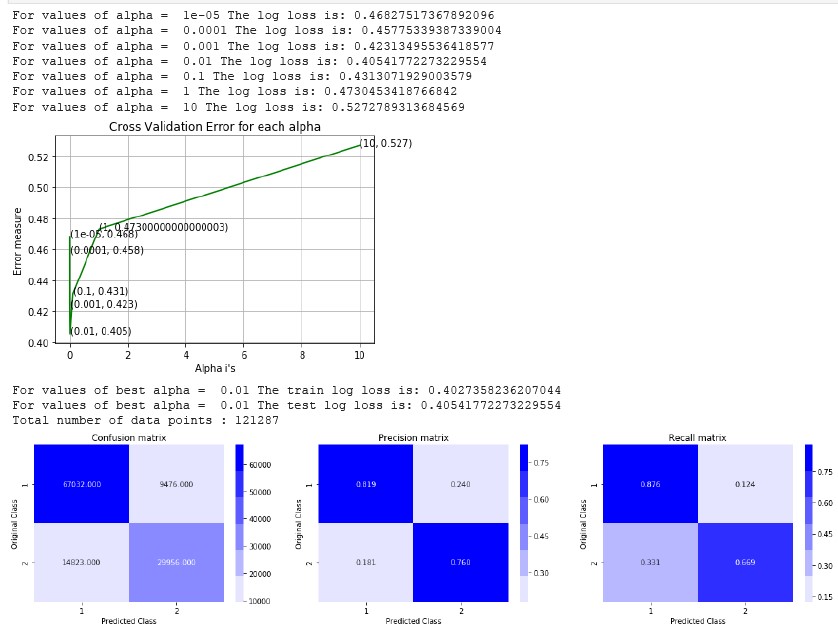


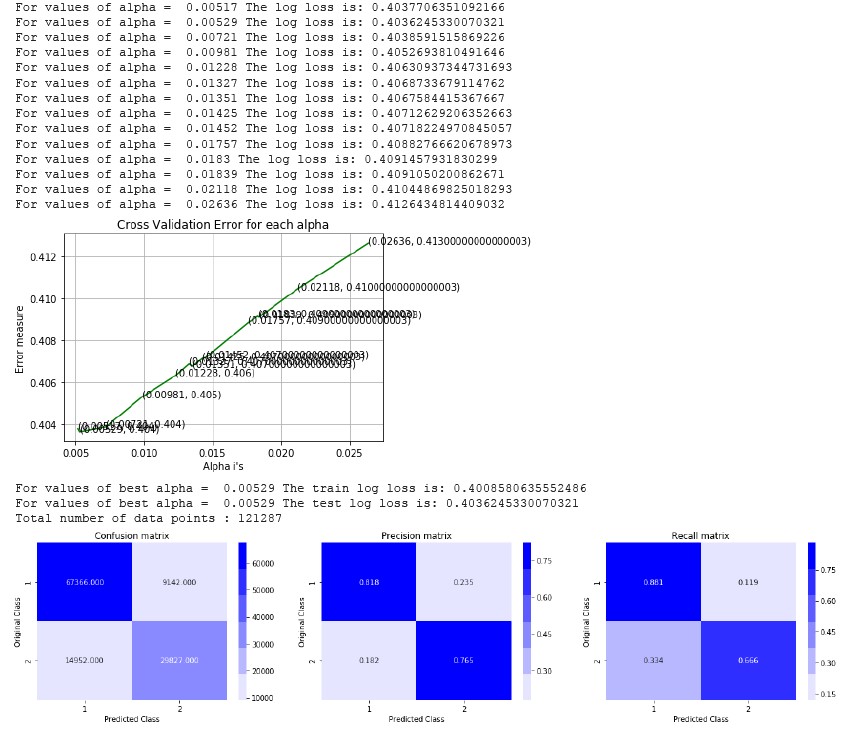
**LINEAR SUPPORT VECTOR MACHINES WITH HYPERPARAMETER TUNING**

Following are the results.

* Using l1 regulariser.

****

* Using l2 regulariser****
* Using random search cross validation

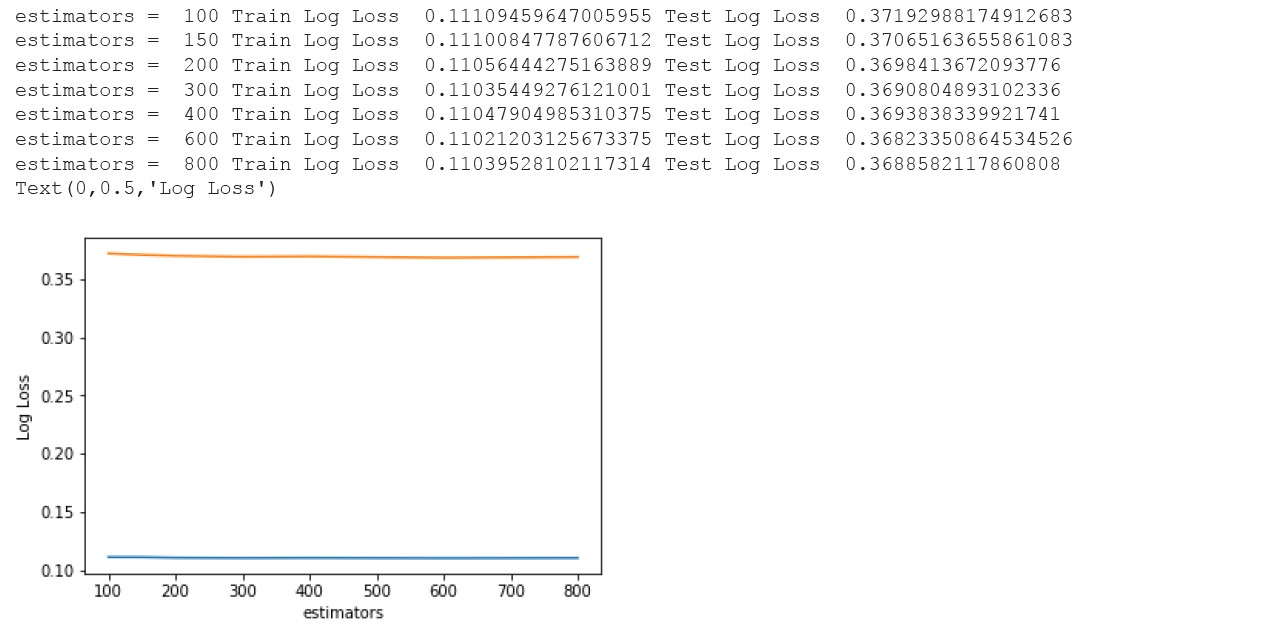
****

**RANDOM FOREST**

Following are the results:

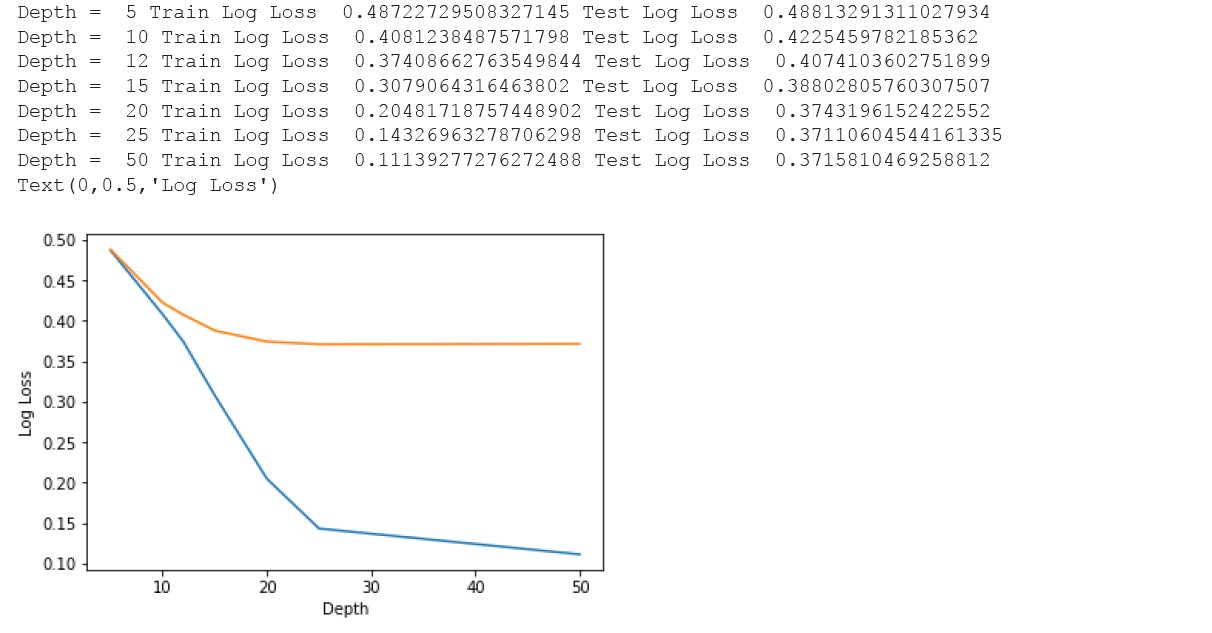
* changing estimators while keeping the max depth as default

In this approach we can see that train log loss is decreasing as estimators is increasing while test log loss is nearly at the same level. There is a significant difference between train and test log loss as the value of estimator is increased. This suggests that the model is overfitting.



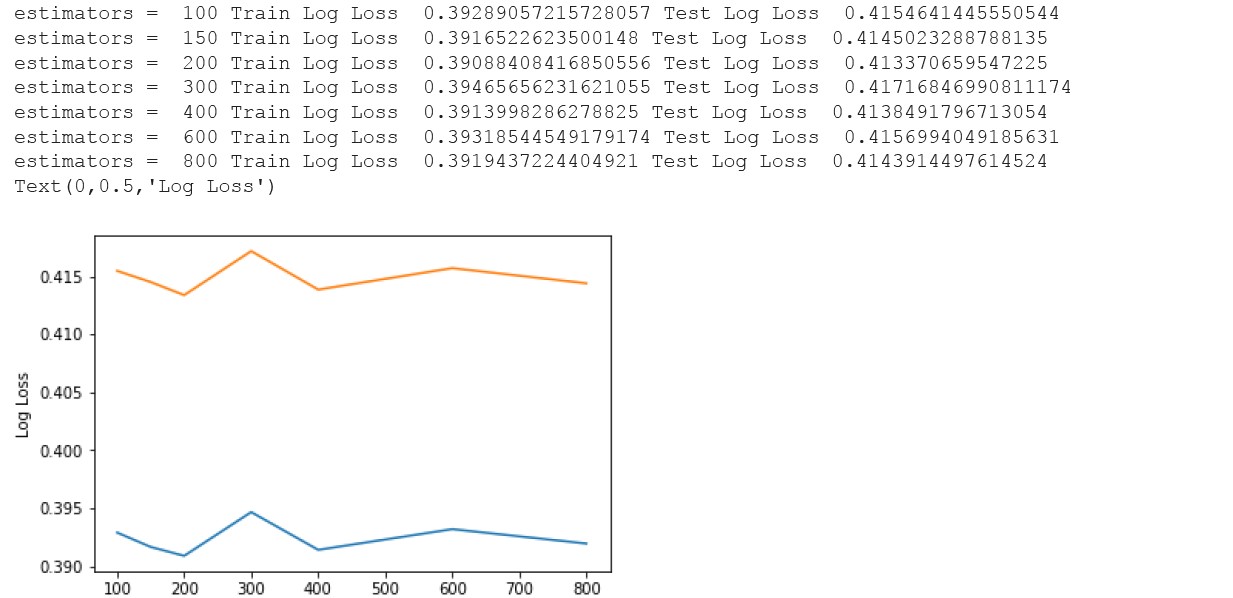
* changing max depth while keeping estimator as constant

In this approach too we can see that train log loss is decreasing as estimators is increasing while test log loss is nearly at the same level. There is a significant difference between train and test log loss as the value of estimator is increased. This suggests that the model is overfitting.



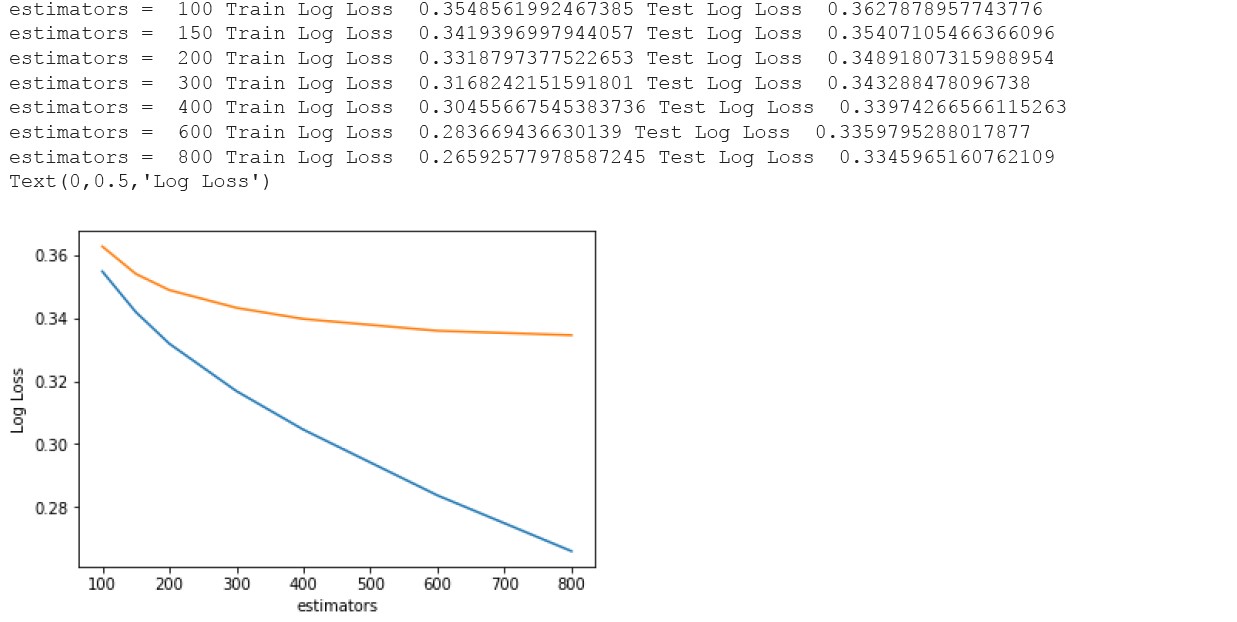
* changing estimators while keeping max depth as 11

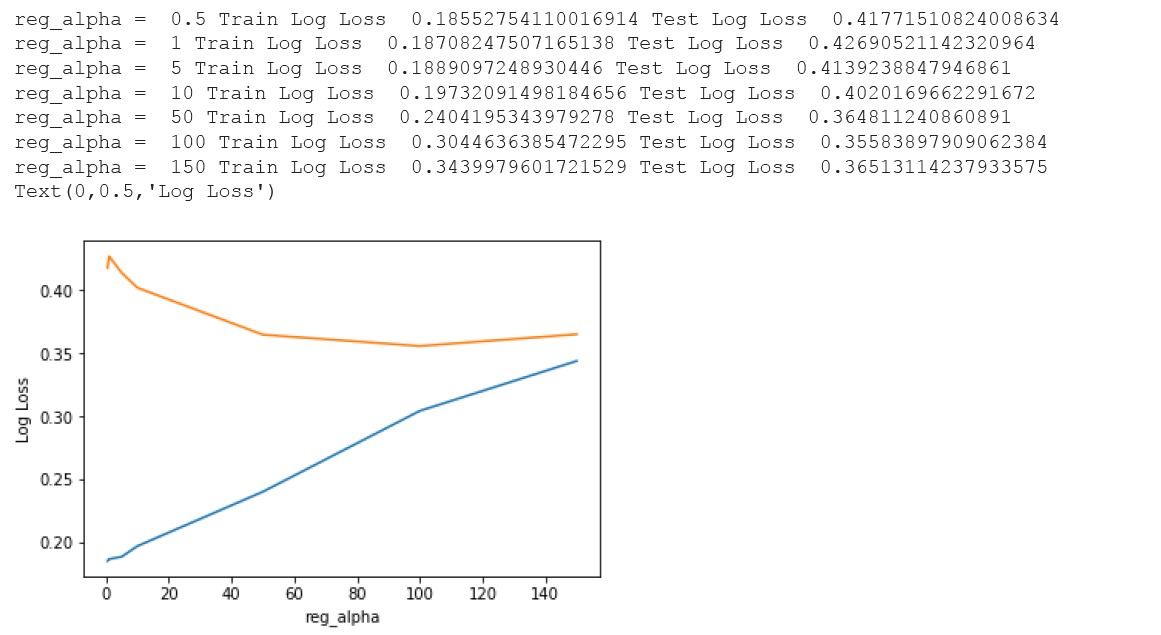
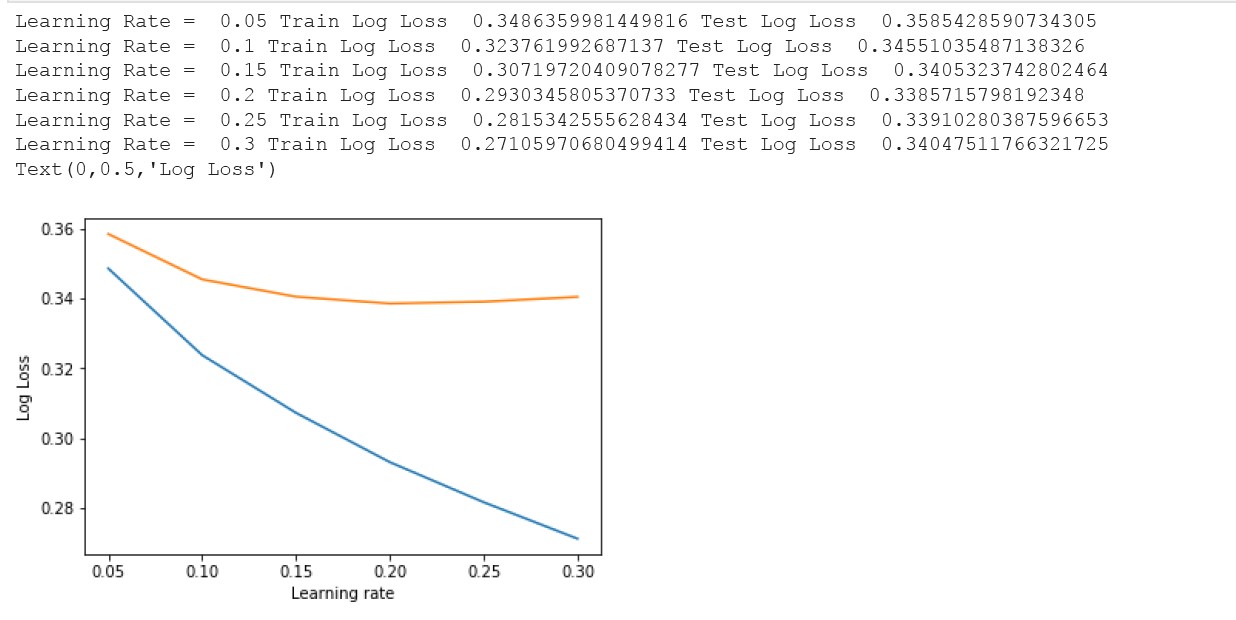
In this approach the model is behaving fairly well.



**XGBoost**

There are several parameters in XGBoost. To find out the optimum parameters we have to first run the model on random parameters and then start improving from there. Following are the results of a few trials.

l.



After several attempts the optimum values of the parameters are decided. Following is the final result.



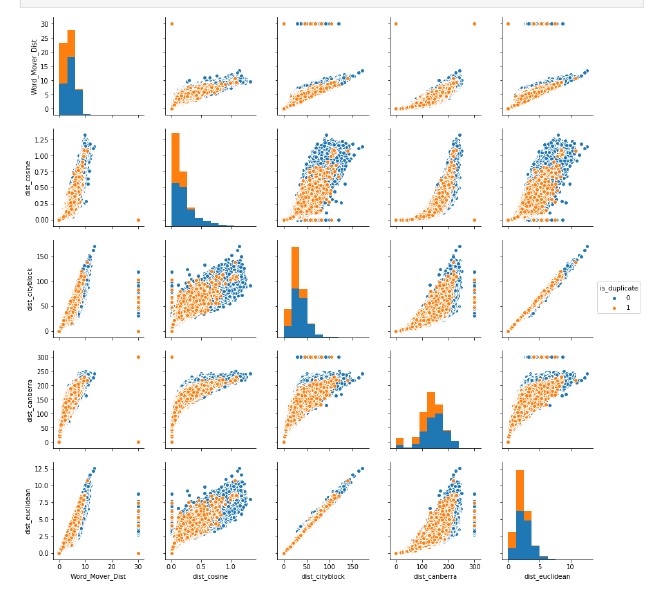
All the models deployed in the above section were build by using the data set which was made by applying tf-idf of first kind i.e. tf-idfA. The whole process is done again with tf-idfB and with some additional features such as average word2vec to see whether there is any sort of improvement or not.

Basically, the machine learning models are trained three times:

1. On tf-idfA data
2. On tf-idfB data
3. On data using additional features such as distance metrics and average word2vec.

As what is being done on the whole has been made clear so it won’t make any sense to explain each of the model on each kind of data set. So, we should move towards the final result and conclusion.

**Distance Features Analysis**

****

**RESULT**

*Log loss of random model –***0.8876992**

* On first kind of data (tf-idfA)

1. Logistic Regression Log Loss – 0.4003415
2. Linear SVM Log Loss – 0.4036245
3. Random Forest Log Loss – 0.4143914
4. XGBoost Log Loss – 0.362546

* On second kind of data (tf-idfB)

1. Logistic Regression Log Loss – 0.358445
2. Linear SVM Log Loss – 0.362049

* Using extra features

1. Logistic Regression Log Loss – 0.3882535
2. Linear SVM Log Loss – 0.394458
3. XGBoost Log Loss – 0.313341