CSN-382

Project Report

Thanmai Sahith Asavadi – 20114018
Nimmagadda Vasavi – 20114064
Murthati Mahibau – 20114058

Prof. Balasubramanian Raman
22 Apr.23

Table of Contents

1 About The Project

1.1 Introduction

Quora is an online 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. It is better to have 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.

1.2 Objective

To train a model to detect similar / duplicate questions in Quora.

1.3 Dataset

- Quora Question Pairs
- Dataset Structure: {id, qid1, qid2, question1, question2, is_duplicate}
- Size: approx. 400k
- So, from the dataset we can see that this is simply a binary classification problem, where we have to classify a a touple containing question pairs into a class of 0 or 1.
    - 0 – The questions are not similar.
    - 1 – The questions are similar.
- *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

1.4 What I Did?

- EDA
- Train on random model
- Train with logistic regression
- Train with SVM
- Train with XGBoost.

2 Exploratory Data Analysis(EDA)

2.1 Feature Extraction

- Analysed and extracted some basic features which might come in handy in the future.
- Some exmaples:
  - *freq_id1, freq_id2 :* Frequency of occurrence of questions
  - *q1_len, q2_len :* Length of questions
  - *q1_wordcount, q2_wordcount :* No. of words in each question
  - *common_words_count :* No. of words in common b/w q1 and q2
  - *tot_words :* Total no.of words in question 1 and 2 combined
  - *word_ratio : common_word_count / tot_words*
  - *add_freq : freq_id1 + freq_id2*
  - *sub_freq : abs(freq_id1 - freq_id2)*

2.2 Feature Analysis

- Plotted some plots of different attributes vs the *is_duplicate* attribute to get an idea of importance of some of the attributes.
- Compared violin plots to get an idea about the distance b/w the means of each classes in *is_duplicate.*
- If the difference between mean is significantly large then we can conclude that the feature is important, since it is able to distinguish b/w classes to a significant extent.

2.3 Preprocessing the text

- Did some preprocessing on the text data like:
  - Stemming the words in each sentence
    - *Stemming: Converting multiple forms of the same word into one stem.*
  - Removing stop words
  - Expanding Contractions
  - Lowering the case to make the case uniform
  - Truncating and trimming spaces to name a few.

2.4 NLP Based Features – Tokens

- Analyse the stop words in our text corpus.
- Extract some more features related to tokens like:
  - *cwc_max : common_words_count / max(q1_wordcount, q2_wordcount)*
  - *cwc_min : common_words_count / min(q1_wordcount, q2_wordcount)*
  - *csc_max : common_words_count / max(len(q1_stop_words), len(q2_stop_words))*
  - *csc_min : common_words_count / min(len(q1_stop_words), len(q2_stop_words))*

- *ctc_max : common_words_count / (max(q1_tokencount, q2_tokencount)+e)*
- *ctc_min : common_words_count / (min(q1_tokencount, q2_tokencount)+e)*
- *first_word_eq :* 1 if first word of q1 = first word of q2
- *last_word_eq : 1* if last word of q1 = first word of q2
- *absdiff_token : abs(q1_tokencount – q2_tokencount)*
- *mean_token : (q1_tokencount + q2_tokencount) / 2*

2.5 Fuzzy Logic Features

- Fuzzy string matching is the technique of finding strings that match a pattern approximately (rather than exactly).
- Compared various fuzzy parameters like:
  - *fuzz_ratio :* Percentage of matching b/w the whole strings q1 and q2.
  - *fuzz_partial_ratio :* Percentage of matching b/w the partial strings q1 and q2.
  - *fuzz_sort_ratio :* Percentage of matching b/w strings q1, q2 ignoring order of the words.
  - *fuzz_set_ratio :* Percentage of matching b/w strings q1, q2 ignoring duplicate occurances.

2.6 Fuzzy Feature Analysis

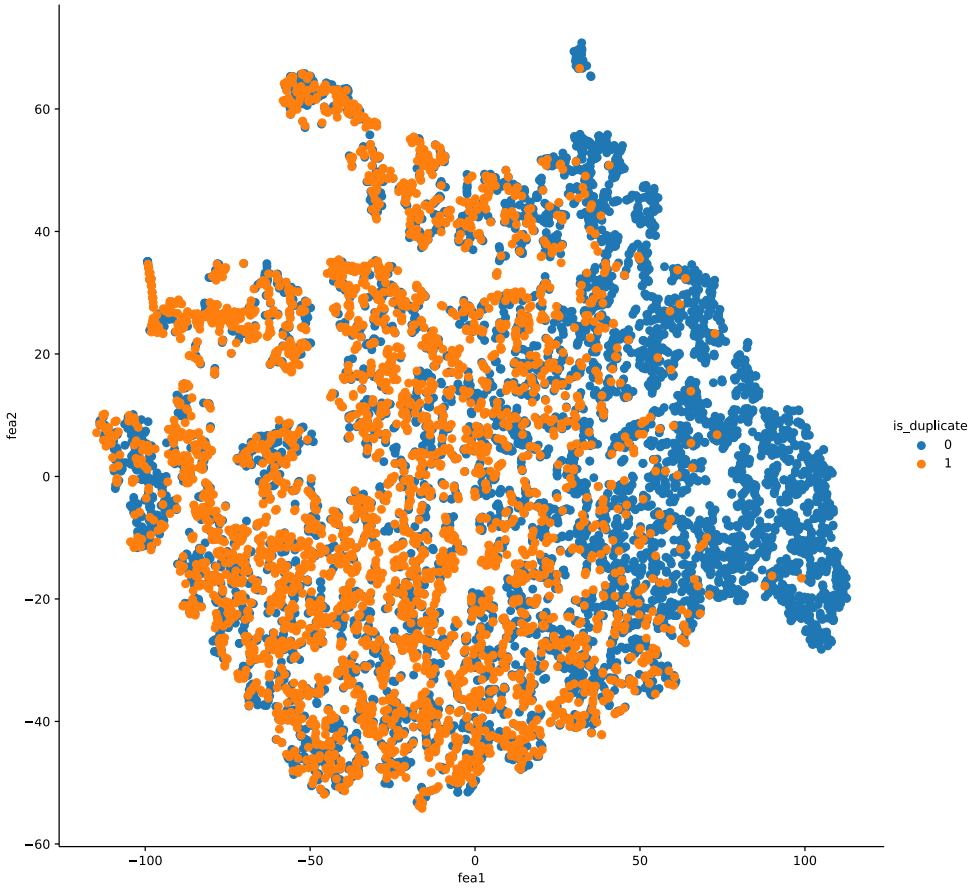
- Performed Univariate analysis of the above four fuzzy features.
- Plotted some violin plots and probability distributions for each of these features to get a better insight into the data.

2.7 Feature Analysis of NLP Features

- Pair plotted some of the features like common word count and common token count to get an insight into relations b/w various variables.
- Also plotted some wordclouds as parth of the anlaysis.
  - *Word Cloud: A word cloud (also known as a tag cloud) is a visual representation of words. Cloud creators are used to highlight popular words and phrases based on frequency and relevance. They provide you with quick and simple visual insights that can lead to more in-depth analyses. They highlight the most frequent words and thereby we can understand which words occur more frequently as part of the analysis.*

2.8 Visualisation and Dimensionality Reduction

- Used t-SNE for dimenstionality reduction.
  - t-SNE is a nonlinear dimensionality reduction technique that is well suited for embedding high dimension data into lower dimensional data (2D or 3D) for data visualization.
  - t-SNE stands for t-distributed Stochas- tic Neighbor Embedding, which tells the following:
    - Stochastic → not definite but the random probability
      Neighbor → concerned only about retaining the variance of neighbour points.
      Embedding → plotting data into lower dimensions



t-SNE Visualisation

2.9 Using Tfi – idf word vector

- Used Tfi – idf vectorizer for converting words into vector of numbers.
- Tfi-idf is short for term frequency-inverse document frequency is a nu- merical statistic that is intended to reflect how important a word is to a document in a collection or corpus.

$$TF(t,d) = \frac{number\ of\ times\ t\ appears\ in\ d}{total\ number\ of\ terms\ in\ d}$$

$$IDF(t) = log\frac{N}{1+df}$$

$$TF - IDF(t,d) = TF(t,d) * IDF(t)$$

2.10 Feature Selection

- We've extracted and added many features to the dataset during the EDA phase many of which might not be very useful for our model, so we can discard them safely to reduce the noise and prevent overfitting.
- Used some of sklearn's feature selection techniques to delete some of the attributes.

3 Training

3.1 Train Test Split

- We need a validation set for tuning hyperparameters to improve the performance of our model.
- So, we've split our dataset in a 7:3 ratio for train and test.
- And we've split it using 'stratify' to make the split random, so the essence and balance of the dataset is preserved in both the subsets.

3.2 Hyperparameter Tuning

- In machine learning, hyperparameter optimization or tuning is the problem of choosing a set of optimal hyperparameters for a learning algorithm.
- A hyperparameter is a parameter whose value is used to control the learning process.
- Hyperparameter tuning is an essential part of controlling the behavior of a machine learning model. If we don't correctly tune our hyperparameters, our estimated model parameters produce suboptimal results, as they don't minimize the loss function.
- Here there is an example of graph plot between the logloss and various values of hyperparameter used. We will choose the best hyperparameter which minimises the loss.

3.3 Evaluation Metrics

- In this section we'll define the metrics we'll be using to evaluate our model
  - Log Loss :
    - Log-loss is indicative of how close the prediction proba- bility is to the corresponding actual/true value (0 or 1 in case of binary classification).
    - The more the predicted probability diverges from the actual value, the higher is the log-loss value.
  - Confusion Matrix :
    - An NxN table that aggregates a classifica- tion model's correct and incorrect guesses.
    - One axis of a confusion matrix is the label that the model predicted, and the other axis is the ground truth. N represents the number of classes.
    - For example, here is a sample confusion matrix for a binary classi- fication model.

	Tumor (predicted)	Non-Tumor (predicted)
Tumor (ground truth)	18	1
Non-Tumor (ground truth)	6	452

3.4 Training a Random Model

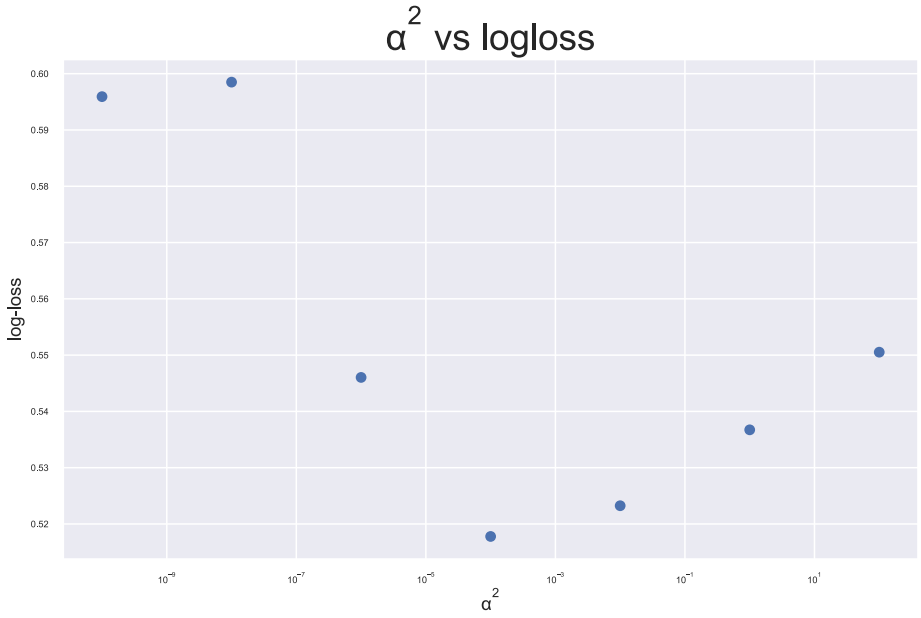
- Used a random guesser to get a better understanding of log loss and confusion matrix behaviour for our dataset.

Log loss on Test Data using Random Model 0.8853319606206358

3.5 Training on Different Models

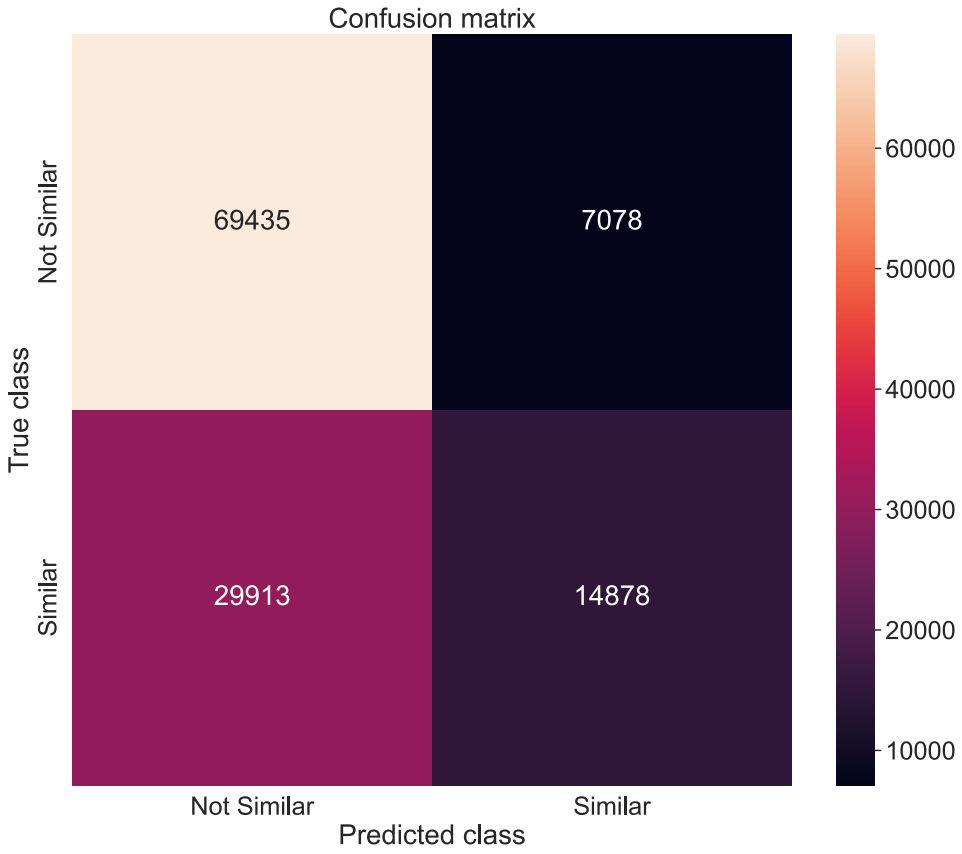
3.5.1 Logistic Regression

- Logistic Regression models the probability of an outcome given input.
- It is generally used for binary classifi- cation.
- In this model we use sigmoid function to define conditional probability.
- Now firstly we need to find optimal line which gives a sigmoid function which indeed fits the train data and also performs well over training data.
- So we need to determine optimal weights (w^t) which minimises log loss, for that we use stochastic gradient descent. In stochastic gradient we start from random value of weights and then move towards the optimal weights step by step.
- Each step size might be different. If we are closing towards optimal weights then step size decreases or if we are far away step size would be more.
- The variation in step size depends on current weights and a hyperparameter termed learning rate.
- So varying learning rate might gives optimal results.
- So, we've tried tuning alpha by trying out many values and choosing the one that gives the minimal log loss.



Hyper parameter tuning for logistic regression

Log loss on Test Data using Logistic Regression 0.5469825829409015

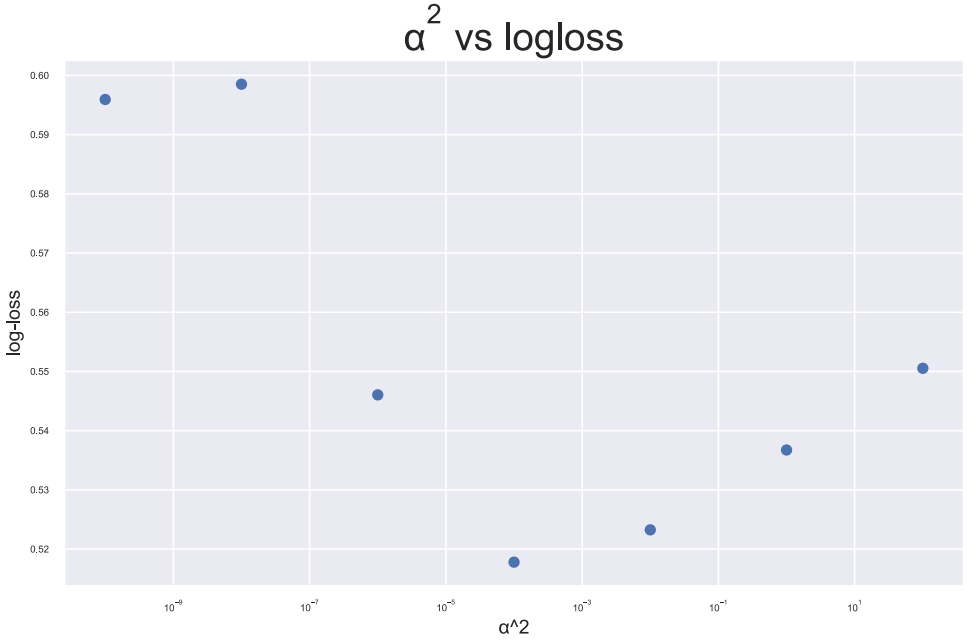


Confusion matrix for Logistic Regression

Test Accuracy for Logistic Regression 69.5%

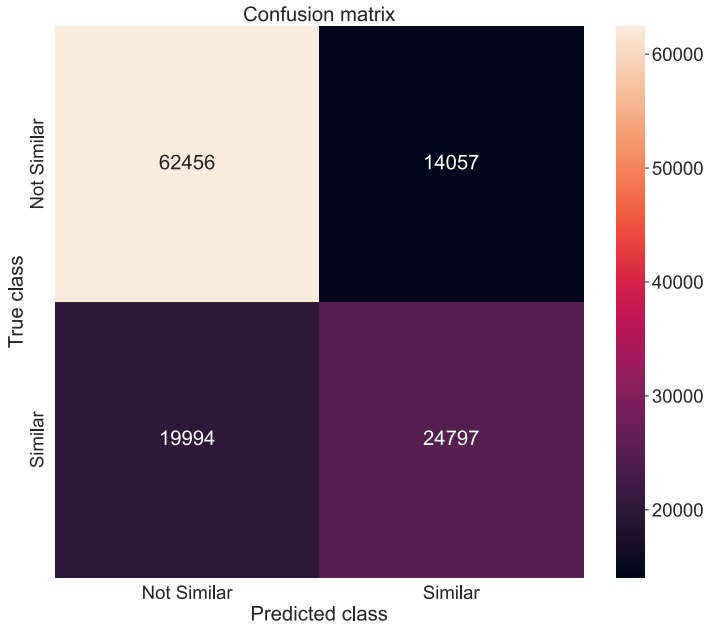
3.5.2 Support Vector Machine (SVM)

- Support Vector Machine(SVM) is a supervised machine learning algorithm used for both classification and regression.
- The objective of SVM algorithm is to find a hyper- plane in an N-dimensional space that distinctly classifies the data points.
- The dimension of the hyperplane depends upon the number of features. If the number of input features is two, then the hyper- plane is just a line. If the number of input features is three, then the hyperplane becomes a 2-D plane.
- We have trained a Support Vector Machine with the sigmoid kernel and did some hyperparameter tuning to optimise the performaance / accuracy.



Hyper parameter tuning for SVM

Log loss on Test Data using SVM 0.5177835174239483



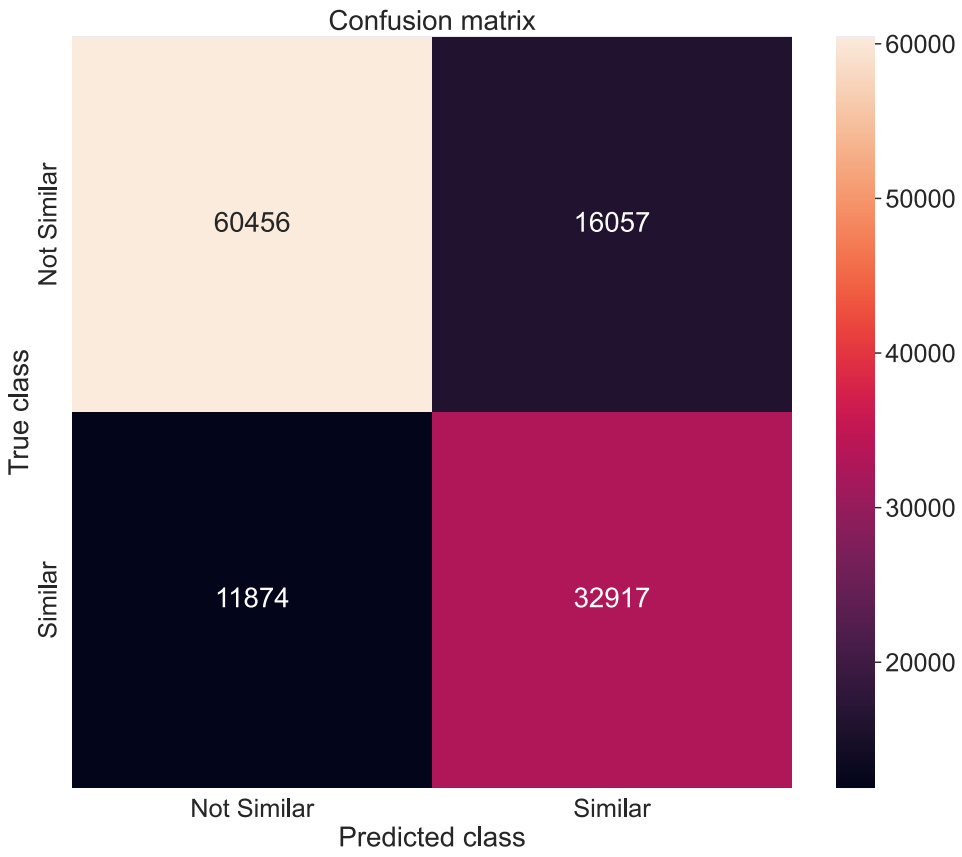
Confusion matrix for SVM

Test Accuracy for SVM 71.93%

3.5.3 XGBoost

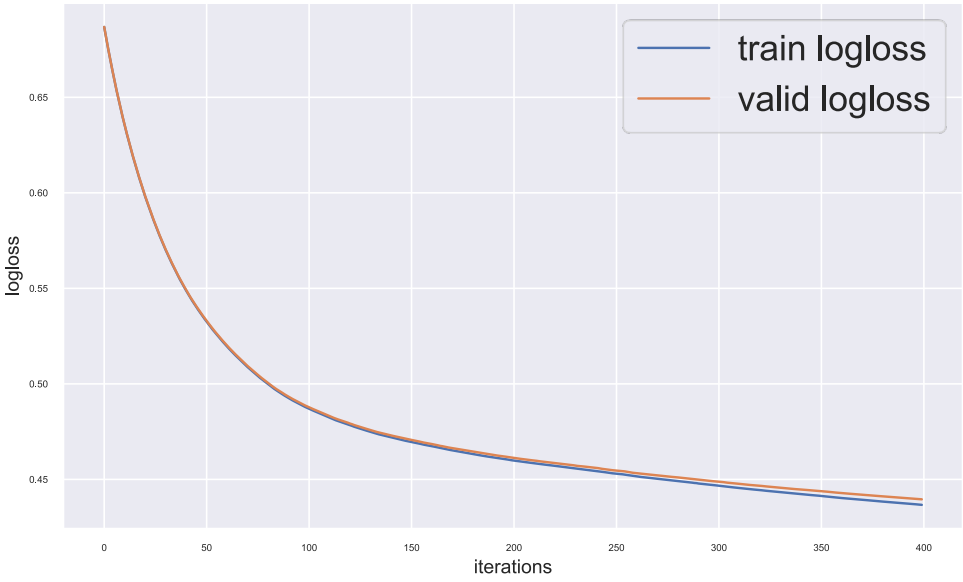
- XG boost is an extreme gradient boosting technique for descision trees.
- In this algorithm, decision trees are created in sequential form. Weights play an important role in XGBoost.
- Weights are assigned to all the independent variables which are then fed into the decision tree which predicts results.
- The weight of variables predicted wrong by the tree is increased and these variables are then fed to the second decision tree.
- These individual classifiers / predictors then ensemble to give a strong and more precise model. So, simply speaking it is an ensemble learning model.
- It can work on regression, classification, ranking, and user-defined prediction problems.

Log loss on Test Data using XGBoost 0.43958440769961377



Confusion matrix for XGBoost

Test Accuracy for XGBoost 76.97%



4 Conclusion

- To conclude, we've analysed, cleaned, processed the data at the beginning followed by trying various models to check for theeir results and performance for the given task.
- In the end, the observations are as follows
  - Log loss : Logistic Regression > SVM > XGBoost
  - Accuracy : XGBoost > SVM > Logistic Regression\
- Just to conclude, why XGBoost over a perceptron? Since it's just a binaary classification problem, a simple perceptron should be good enough.
- But here, the reason for choosing XGBoost over perceptron is thanks to its ability to capture and model non – linear and complex information which a simple perceptron fails to.
- But of course, if the data is fairly simple and a linear decision boundary's good enough as a classifier then perceptron's better than XGBoost since, it's significantly less intensive computationally, compared to XGBoost.

5 References

- https://www.kaggle.com/c/quora-question-pairs
- https://www.kaggle.com/datasets/quora/question-pairs-dataset
- https://en.wikipedia.org/wiki/Support_vector_machine
- https://www.analyticsvidhya.com/blog/2018/09/an-end-to-end-guide-to-understand-the-math-behind-xgboost/