Toxic Comment Classification - Project Report

Shaishav Shah
CISE department, University of Florida
shah.sh@ufl.edu - 1136-3317

***Abstract*—Freedom of speech is a principle that allows an individual to express his/her ideas or opinions freely without any kind of fear, censorship or restraint. However this basic principle can be misused by some people when they express something in an abusive way, which is inappropriate and may offend other people participating in that communication. In current times as people are much more connected than ever due to the multitude of social media platforms, this situation arises more often than not. I personally feel that such nuisance should be regularly moderated to create a more user friendly online communication experience for each one of us. Only then the users would feel comfortable and enjoy participating in online conversations.**

In this project, I have implemented a machine learning[2] model that can autonomously identify and classify the toxic comments posted online.

I. Introduction

In today's world, social media has become one of the most efficient ways for people to communicate and discuss. People can share their ideas, their feelings, their opinions, the events happening in their life as well as react to the comments posted by other people. Considering all these advantages, social media may seem like a paragon for communication, but it's a double-edged sword as it can easily be abused by toxic comments from some users. This kind of abuse or harassment may make other users leave the conversation and may also prohibit them from participating in future conversations. This makes it an important requirement for any platform to have a system in place in order to moderate such behaviors. While some organizations have traditional moderators in place to identify these types of comments and take appropriate actions, that is not the most efficient way for solving this problem. This can be more effectively solved by using a machine learning model that will do the job of identifying such comments as well as classifying them in different level of toxicity.

This issue falls under the classification of Natural Language Processing where we attempt to recognize the intention of the speaker or writer, and act likewise. The idea of this project is inspired from one of the Kaggle competition named toxic comment classification challenge, which is the research being conducted by the Conversation AI team, an initiative by Jigsaw and Google.

This model can further be extended to be distributed as an API, which can then be used by any application wanting to moderate comments on their platform. They can simply send their comment text in request and get the toxicity level for that text in response.

II. Dataset

The dataset used in this project can be found at Kaggle's Toxic Comment Classification Challenge [1]. It contains a large number of Wikipedia comments which have been identified and labeled by human moderators for toxic behavior.

id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
0575db58337a8fb2	I know that's not mature. And anyway, that was...	1	0	0	0	1	0
0575dea6a4fdc093	HELLO Garmiamel (admittedly I have trouble s...	0	0	0	0	0	0
0576883be8b7c2b0	Thank you, I will take that under advisement. ...	0	0	0	0	0	0
057894cf4738a5d8	You are a gay homo. I hope you choke on your p...	1	0	1	1	1	1
05794a042eca3645	You act like you are constructive and you make...	0	0	0	0	0	0

Fig. 1. Training dataset: A glimpse

id	comment_text
00001cee341fdb12	Yo bitch Ja Rule is more succesful then you'll...
0000247867823ef7	== From RfC == \n\n The title is fine as it is...
00013b17ad220c46	" \n\n == Sources == \n\n * Zawe Ashton on Lap...
00017563c3f7919a	:If you have a look back at the source, the in...
00017695ad8997eb	I don't anonymously edit articles at all.

Fig. 2. Testing dataset: A glimpse

A glimpse of training dataset we used can be seen in Figure 1. It contains the following fields:

- id: identifier for the user who had written the comment.
- comment_text: comment text to be classified in one of the toxicity levels.

Below Six toxicity classes are defined which has two values 0 and 1 denoting if a text belongs to that class or not respectively.

- toxic
- severe_toxic
- obscene
- threat
- insult
- identity_hate

id	toxic	severe_toxic	obscene	threat	insult	identity_hate
001d39c71fce6f78	0	0	0	0	0	0
001d739c97bc2ae4	1	0	0	0	0	0
001eba8bbe728e54	0	0	0	0	0	0
001eff4007dbb65b	1	0	1	0	1	0
0020baa22b95d1b0	0	0	0	0	0	0

Fig. 3. Testing labels dataset: A glimpse

A glimpse of data we used for testing can be see in Figure 2 and Figure 3. The testing data is divided into two parts, the Test dataset which contains the actual data, and the Labels dataset which contains the results for the test data. We used this data for calculating the accuracy of our model. The data between these two sets is mapped using the id field.

A. Data Visualization & Analysis

I have created few visualizations of the data that helped me in preprocessing the dataset and eventually improving the training time of our model. First, I created the frequency distribution graph of the training data as shown in Figure 4. From this graph, the following implications can be made about the data.

- toxic count - 15294, almost 1 out of every 10 comments
- obscene count - 8449, almost 1 out of every 20 comments
- insult count - 7877, almost 1 out of every 20 comments
- The frequency of severe_toxic(1595), threat(478) and identity_hate(1405) is rare.
- Overall there are 16225 number of comments which belongs to one of the toxic class, and 9865 number of comments which belongs to two or more classes.

Second, I have done analysis for the length of the comments to get an idea of average comment length as well as to know how varying the comment lengths are and it can be seen in Figure 5. From this graph, it can be observed that the comments have varying lengths ranging from 250 to 2000 and more than half of the comments have length up to 500. It it also observed that as we move towards longer length comments the count keeps on falling.

Also, I have made a graph depicting the length distribution by each individual class present in our dataset as shown in Figure 6. This graph also gives us the similar observation as above that as we move towards longer length comments the count keeps on falling across all the comment classes and most of the comments labeled as toxic irrespective of the toxicity class have length up to 500.

Using this graphs and observations, I have made a decision to remove all the comments having length more than 500, since long length comments for training would increase the number of words manifold, it is vital to set a threshold in order to achieve optimum results for our model. Hence before preprocessing the dataset, I have removed all the comments having length more than 500.

B. Data Preprocessing

- Convert all comments to lowercase: I have converted all the comments to lowercase for easier processing.
- Removing special characters: I have used punctuations field from string library for this. I have removed ’ from it because there are some comments containing words such as can’t, won’t etc. I have also appended numbers from 0 to 9 to it and applied the *maketrans* function to create a map of each character in the punctuation string with a space.

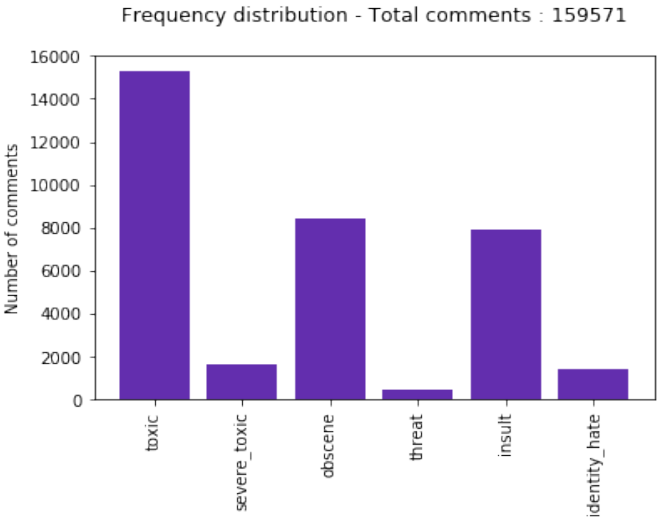

Fig. 4. Testing dataset: Frequency distribution

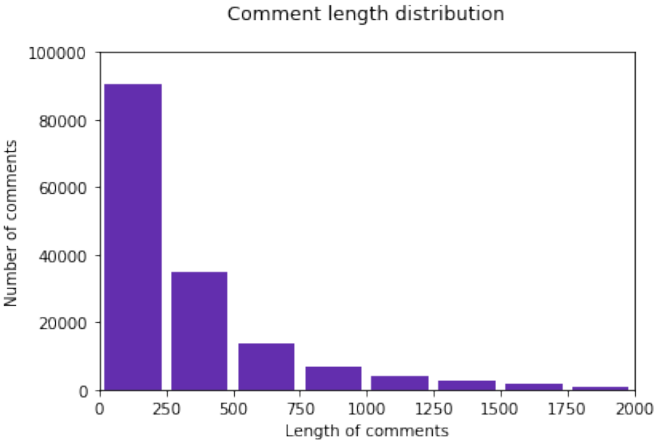

Fig. 5. Testing dataset: Length distribution

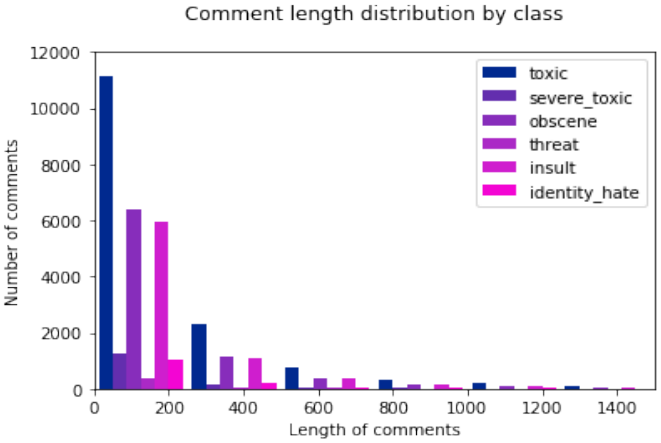

Fig. 6. Testing dataset: Length distribution by class

- Removing Stop words: The words which are used frequently in written and verbal English communication and don't have any positive or negative impact on the statements are called Stop words. For example, *a*, *this*, *the* are Stop words. I have used *nltk* library to get English language stop words, and appended letters from 'b' to 'z' to it since single letter words don't convey any meaningful information.
- Stemming: The process of transforming inflected or derived words to their root form is called Stemming. For example, it reduces words like fishing, fished, and fisher to the stem fish. This helps in achieving higher accuracy in training process.
- Lemmatising: The process of grouping the inflected forms of a word together is called Lemmatising. It is used so that these group of words can be analysed as a single item.
  I have used *nltk's wordnet* libray for this performing Stemming and Lemmatising.
- Count Vectorizer: It is used for transforming a list of words into a matrix of words where the column headers represent the words and their values represent the frequency of occurance for that word. I have used *sklearn* library's *CountVectorizer* module for this process and passed our custom created stop words to it's constructor.

III. Implementation

As the comment here may belong to one or more toxicity classes, this is a multi-label classification type of problem. This kind of problems can be solved using Problem Transformation methods or Adaptation algorithms.

A. Problem Transformation methods

Most of the conventional learning algorithms are created for single-label classification problems. Therefore a lot of methods focuses on transforming the multi-label problem into multiple single-label classification problems, to take advantage of the existing single-label algorithms.

In this method, we take a multi-label classification problem and try to apply some transformation to it in order to convert it in to a single-level classification, and then we execute a binary or multi-class classification on it.

1) Binary Relevance method: This is the simplest type of transformation method where a separate classifier is created for each label and the interdependence of different labels on each other is not taken into account. Each label here is solved individually like a single-label classification problem. Each classifier's job is to simply predict whether the comment belongs to that label or not and the union of all the classifier's prediction is taken as multi-label output. Figure 7 illustrates an example of a multi-label classification problem having three labels {L1, L2, L3} using Binary Relevance.

In other words, if there are n labels, this method creates n data sets from the original data set, one for each label and then trains single-label classifier on each of them. This is simple and easy to implement but does not work well when there is interdependence among the labels.

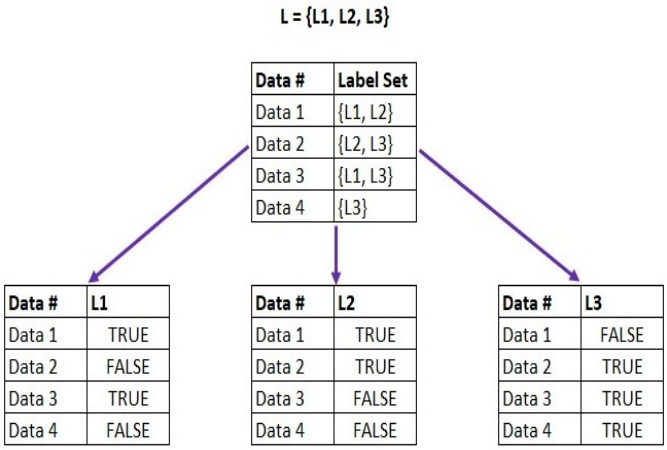

Fig. 7. Binary Relevance

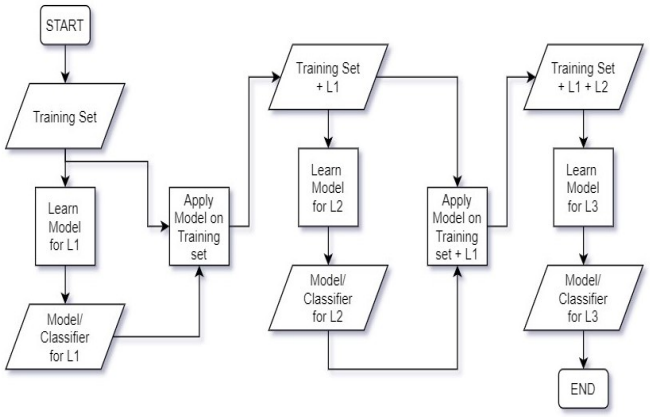

Fig. 8. Classifier Chain

I have implemented this using different classifiers from *sklearn* library such as, *MultinomialNB*(Multinomial Naive Bayes), *GaussianNB*(Gaussian Naive Bayes), *SVC*(Support Vector Classification).

2) Classifier Chain method: In order to overcome the shortcoming of Binary Relevance method, I implemented Classifier Chain method which considers the interdependence among the labels. This method works by creating a chain of classifiers C_0, C_1,..., C_n where each classifier C_i uses the predictions made by all the classifiers up to C_j where $j < i$. This way the method takes the label correlations into account. It is good idea to use this method because there can be some data that may show dependence between some classes such as toxic and severe_toxic.

The total number of classifiers here is equal to the number of labels. Figure 8 illustrates an example of a multi-label classification problem having three labels {L1, L2, L3} chained in the same order.

Data #	L1	L2	L3	L4
Data 1	1	0	0	1
Data 2	0	1	1	0
Data 3	1	0	1	0
Data 4	0	0	0	1
Data 5	1	1	1	0

Data #	Label Set
Data 1	1001
Data 2	0110
Data 3	1010
Data 4	0001
Data 5	1110

Fig. 9. Label Powerset

3) Label Powerset method: This method considers all possible correlations between the labels into account. Each possible combination of labels hence serves as an input to the single-label classifier, transforming our multi-label classification problem to a multi class classification. This method serves good for our dataset since we have many comments which have 0's for all labels and many have two or more labels together.

The maximum number of classifiers required for this method in worst case is $2^{|L|}$, where $|L|$ gives the total number of labels. It has very high computational complexity as it considers all possible combinations. This method is suitable when the count of labels is small, but not when there are large number of labels, because the number of classifiers can grow exponentially with the number of labels which would make it computationally infeasible. Moreover, from all the possible combinations, some might have very few positive examples.

The illustration of this method on four labels {L1, L2, L3, L4} can be seen in Figure 9.

B. Adaptation Algorithms

The Adaptation Algorithms for solving multi-label classification problems focuses on adapting the single-label classification algorithms to multi-label by making changes in cost or decision functions. In this project, I have implemented Multi Label Lazy Learning model known as ML-KNN and Back-propagation for Multilabel Learning Neural Network known as BP-MLL.

1) ML-KNN: [4][6] K-Nearest Neighbors[**?**] is one of the simplest and most used classification algorithm in Pattern Recognition. KNN is a lazy learning algorithm which is non-parametric in nature, meaning it does not make any assumptions about the distribution of underlying dataset. It works by storing every single instance available, and classifies new unseen instances depending on the similarity measure e.g. distance functions.

The ML-KNN model is based on the traditional K-nearest neighbor (KNN) algotihm. In other words it is an adapted multi label version of KNN. For any unseen instance, the k nearest neighbors of it in the training set are recognized as the first step. Following it, based on the statistical knowledge obtained from the label sets of these neighboring instances, for example the number of neighboring instances with every conceivable class, maximum a posteriori (MAP) rule is used to decide the label set for that unseen instance. This is

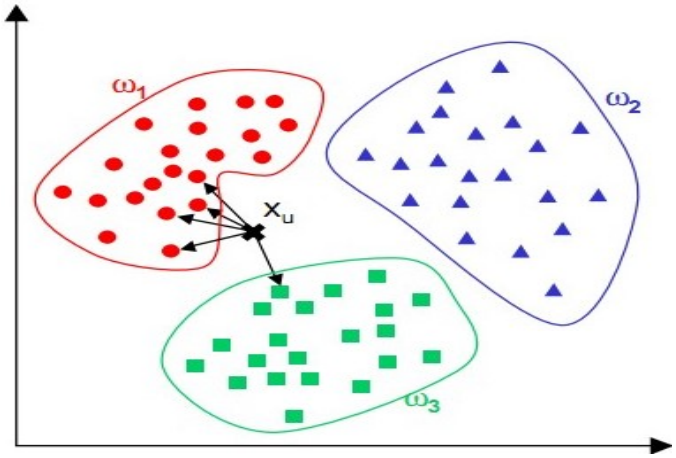

Fig. 10. ML-KNN

demonstrated in Figure 10 where X_u is an unseen dataset, and ω_1 and ω_3 are its nearest neighbors.

The model has proven to perform better in comparison to some well established multi-label learning algorithms for some real-world datasets such as natural scene classification, yeast gene functional analysis and autonomous categorization of web pages. Since our problem is similar to page categorization problem, I have chosen this algorithm in expectation of getting better results. However, the time complexity associated with this algorithm is too large and therefore it is preferred to train this model on smaller datasets.

I have implemented this using *skmultilearn.adapt* module from *scikit-learn*[3] library. This module contains many other adaptation algorithms for multi-label classification including ML-KNN.

2) BP-MLL: [5]

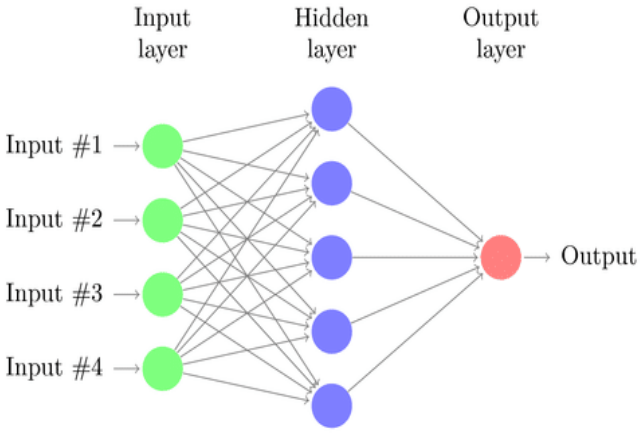

Fig. 11. A simple Neural Network

Neural Network refers to a circuit of neurons that are interconnected together identical to the biological nervous systems. It consist of three layers as shown in Figure 11: 1. Input layer - takes initial input from multiple sources, 2.

Hidden layer - intermediate layer between the input and output layers where all the computation takes place, and 3. Output layer - gives the output for the given input(s). The middle layer is called hidden simply because they are unobserved and there can be more than one hidden layers depending on the use case.

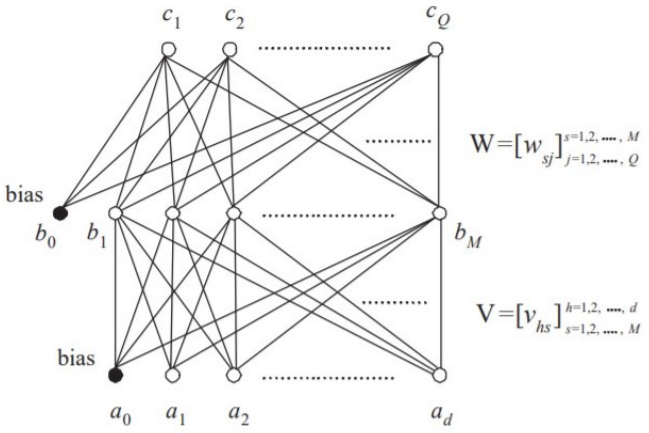

Fig. 12. BP-MLL neural network architecture

The BP-MLL Neural Network is an architecture as shown in Figure 12, the goal of which is to optimize the pair-wise ranking error. This algorithm adapts the traditional multi-layer feed-forward neural networks to implement multi-label classification. The hidden layer here is fully connected with the input and output layers, and do not have any connections to the outside world. I have implemented six output layer nodes, one for each of the label in our dataset. The training time for this algorithm is reasonable given that we use appropriate number of nodes for the hidden layer.

I have implemented BP-MLL using *Keras* library's *Sequential* model with 16 nodes in the hidden layer.

IV. Performance Evaluation

An important thing to note is the fact that the dataset we are using for this project is skewed, which means the ratio of the number of comments which are labeled as toxic for one or more of the six labels to the number of comments which are not toxic is very low. Actually only about 10% comments are toxic as per our data analysis. Therefore, taking accuracy as an evaluation metric would provide invalid measures for our model. For example, even if we design a basic classifier model which classifies every comment to be non toxic, the accuracy score would be 90%. For this reason, I have used *Hamming Loss* and *Log Loss*, which works fine even for skewed dataset.

A. Hamming Loss

Hamming Loss is characterized as the ratio of wrong output labels to the total number of labels. Hamming Loss for a multi-label problem can be implemented by assigning equal weight to each label, and then combining the predictions of each label to produce the final loss value. The formula for the Hamming Loss is given as 1.

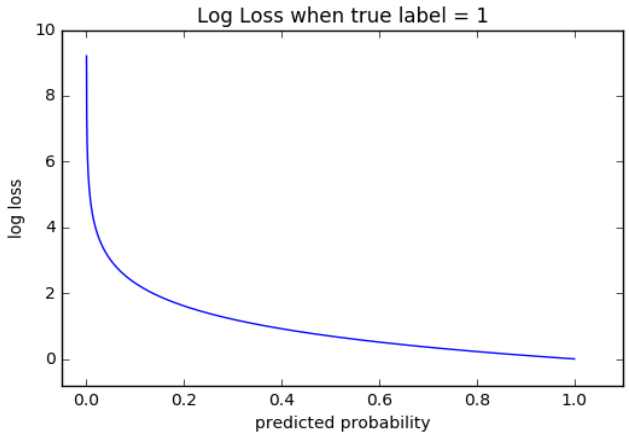

Fig. 13. Log Loss

$$\frac{1}{NL}\sum_{i=1}^{L}\sum_{j=1}^{N} Y_{j,i} \oplus X_{j,i} \tag{1}$$

N = Number of comments
L = Number of Labels
The $\oplus$ in 1 denotes the exclusive-or between the $X_{j,i}$ i.e true value of i^th label for j^th comment and $Y_{j,i}$ i.e the predicted value for the same.

B. Log Loss

Log loss analyzes the accuracy of a model by penalizing the false results produced. The exponentially decaying curve shown in Figure 13 depicts the same. It is suited only for the problems having two or more labels. For calculating the Log loss, each class must be assigned a probability rather than simply giving out the most favorable class. The formula for the Log loss is given as 2.

$$-\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{L} Y_{i,j} \log p_{i,j} \tag{2}$$

N = Number of comments
L = Number of Labels
$Y_{i,j}$ is binary indicator of whether label j is correct or not for comment instance i.
$p_{i,j}$ is model probability of classifying comment instance i to j.
The accuracy of the models we developed based on these selected metrics are shown by the graphs in Figure 14 and 15, while the accuracy comparison can seen in Figure 16.

V. Conclusion

Based on hamming-losses, *LP-MultiNB* performs better than any other loss model. Based on log loss, *BP-MLL* model performs better. The parameters for *BP-MLL* was found using grid-search mechanism, with 16 hidden layers, learning rate of 0.001, 10 epochs and batch size of 64 leads to better performance. To summarize, all models generated by problem

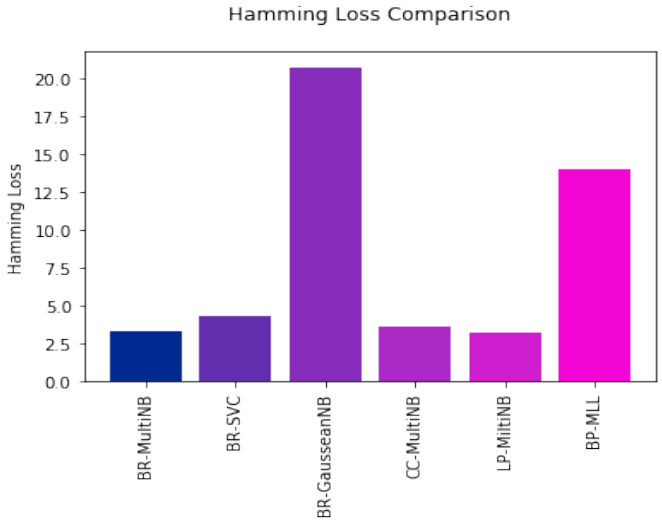

Fig. 14. Log Loss Comparison

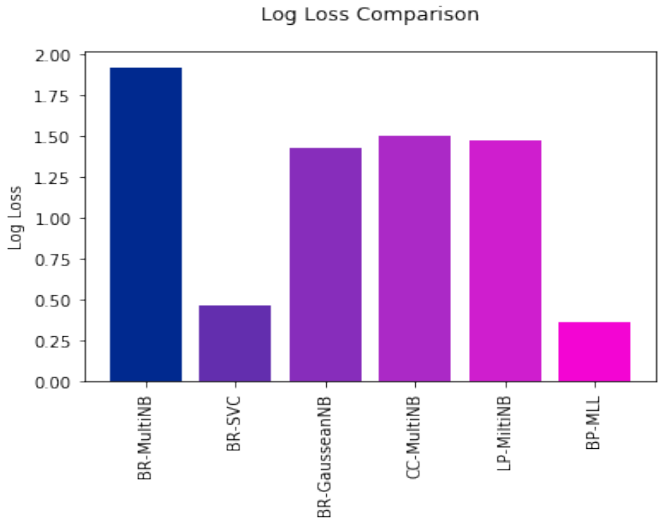

Fig. 15. Log Loss Comparison

transformation methods lead to low hamming loss values, while the models obtained using adaptive algorithm lead low log value losses.

From this project, I have learned so many things including but not limited to data analysis, data preprocessing, multi-label classification problems, solutions for multi-label classification problems such as problem transformation methods and adaptive algotithms, and different evaluation metrics such as Hamming Loss and Log Loss.

REFERENCES

[1] Kaggle. *Toxic Comment Classification Challenge*, 2020 (accessed March 11, 2020).

[2] Diego Oliva and Erik Cuevas. *An Introduction to Machine Learning*, pages 1–11. 11 2017.

[3] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[4] Min-Ling Zhang and Zhi-Hua Zhou. A k-nearest neighbor based algorithm for multi-label classification. volume 2, pages 718 – 721 Vol. 2, 08 2005.

[5] Min-Ling Zhang and Zhi-Hua Zhou. Multilabel neural networks with applications to functional genomics and text categorization. *Knowledge and Data Engineering, IEEE Transactions on*, 18:1338– 1351, 11 2006.

[6] Min-Ling Zhang and Zhi-Hua Zhou. Ml-knn: A lazy learning approach to multi-label learning. *Pattern Recognition*, 40:2038–2048, 07 2007.

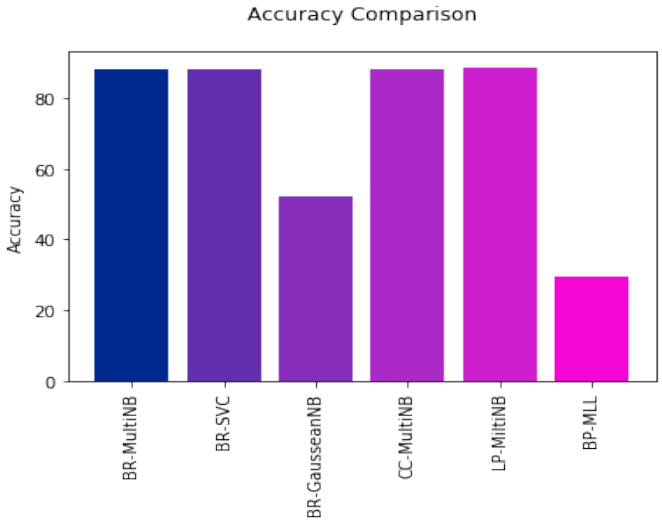

Fig. 16. Accuracy Comparison