IEE 598 – Data Science for System Informatics

Project Report

On

Toxic Comment Classification

Instructor

Dr. Hao Yan

Harish Siva Subramanian (1213355570)

Anirudh Reddy Goluguri (1213278051)

Vignesh Sathyamurthy (1213520150)

I. Introduction

1. Introduction and Motivation to the problem

With a rapid increase in the use of social media at present, there is a large exchange of information within the society. Though this exchange of knowledge is beneficial, there is also the possibility of toxic comments which might influence the emotions of the people, especially the younger generation. So, there is need to filter out these comments and prevent them from appearing in the platform which every person in the world has access to. So, we have developed an approach which can classify each comment posted into two categories – 'Toxic' and 'Non-toxic'.

2. Objective

The main aim of our project is to classify the comment whether they are toxic or nontoxic by tackling with the big data and high dimensionality challenges associated with text classification. We address the big data challenge by using techniques such as partial fit to reduce the time complexity of the problem. To classify the model, we have used Stochastic Gradient Descent (SGD), Naïve Bayes Classification techniques with Gaussian and Multinomial, Random Forest Classification and Multilayer Perceptron techniques. We then used Grid Search and Randomized Search to find the best hyper parameters associated with each classifier. Among the different penalization techniques like L1, L2 norm and elastic net. Gridsearch CV was used to predict the best possible penalization and parameter available in a set to produce high accuracy. These were then used for initial training of the model and the predicting efficiency was found for each classification models.

II. Data Exploration and Preprocessing

1. Initial Exploration of Data

The data consists of the comments with different text formats which includes punctuation marks, spaces and other symbols along with the value of toxic or non-toxic are provided for every comment. The dataset used in this project was obtained from Kaggle site. It does not contain any null value and all the comments are organized properly ready to proceed with data pre-processing. Besides, topic modelling also known as LDA (Latent Dirichlet Allocation) is also performed to explore the split of our data topic-wise. We divided our datasets into 4 topics by using the LDA. It was evident from the result that the data was focused on one topic and the other topics had very less probability of word occurrence. This gave us an idea that a non-linear classifier had to be used to predict even the less probable words. As to get better a view and understanding we have plotted a heat map and we inferred that the distribution of data is heterogeneous. We have also made a scatter plot using the topic modelling topics to show how topics are split. The heat map and cluster plot of the topic is attached below in the appendix section:

2. Data Preprocessing

In preprocessing we have used natural language toolkit (NLTK) method to remove no significant words and unwanted characters to improve the model. First, the datasets were imported, and a model was created in which non-useful words like “the”, “is”, “on” etc., were removed. Besides, we removed the punctuations and converted all the words into lower case. Then the method of stemming was applied to the words. Here all the words are reduced to their lowest form. For Example, the word ‘loved’ is reduced to ‘love’. After this, the sparse matrix was created, and this sparse matrix is like the classification model matrix which is created by a series of independent variables. Then the Bag of words model was created using the Count Vectorizer.

a. **Splitting of Training and Testing Data**

   The data was then split into Training and Testing data. Here a split ratio of 0.2(20%) was used to ensure that we have maximum values in the training set to ensure high model accuracy. Here the number of observations used was 20,000 and the number of features came up to 31,000.

b. **Methods to reduce High observations and High Dimensions**

   Since both the number of observations and features were high, we decided to use **Partial Fit and ‘max_features’** to counter them respectively. The ‘max_features’ argument which is parameter in the count vectorizer is used to limit the features and that take the words that has maximum influence in the dataset has been used to reduce the problems associated with high dimensions. Here we have limited the features to 15000. The partial fit is applied while fitting each classifier model to the dataset. The main idea here is, it will reiterate and fit the classifier for the predefined chunk size for the predefined number of times. This method has eliminated the problems associated with high memory usage.

Finally, we used 20000 observations and 15000 features for training the model.

III. Models Used & Interpretations

1. High Dimensional Challenge(Penalization)

Initially as discussed before, the ‘max_features’ method reduces the sparsity of the matrix by considering the most frequently used words and eliminating the rest. Further penalization techniques were used in the parameter level to reduce the problems of high dimensionality.

In this project we have used SGD Classification as one of our classifiers. The regularization is a penalty added to the loss function that shrinks model parameters towards the zero-vector using best penalty among the squared Euclidean norm L2, the absolute norm L1 and Elastic Net (combination of both L1 and L2). If the parameter update crosses the 0.0 value because of the regularization, the update is truncated to 0.0 to allow for learning sparse models and achieve online feature selection [1]. Since it is very difficult to use each penalization technique individually to fit this high observation dataset, we decided to use GridSearch CV method to find the best penalization technique to be used. The best penalization technique obtained was the L2 penalization. We decided to fit the classifier with L2 penalization to our dataset. The prediction accuracy obtained was around 93%.

2. **Non-Linear Classification Methods**

In this project, we used five different non-linear classifiers to predict the results. The classifiers are Naïve Bayes (Gaussian), Multinomial Naïve Bayes, SGD (Stochastic Gradient Descent), Random Forest Classifier and Multilayer Perceptron.

a. **Naïve Bayes (Gaussian)**

Naïve Bayes classification is based on the principle of Bayes Theorem. They work on independent assumptions which may not be correct sometimes.

Posterior = Prior occurrences * Likelihood/ Evidence

The likelihood here is taken as Gaussian and their parameters in general are estimated using the Maximum Likelihood function [5]. The only parameter associated with Naïve Bayes is 'priors' and here we have not specified the prior values indicating that here the prior values are taken depending on the dataset. The technique of partial fit is applied in the classifier level to deal with high observation datasets as discussed before. The prediction accuracy obtained here is 71.3%.

b. **Multinomial Naïve Bayes**

In the multinomial Naïve Bayes, the multinomial distribution is used instead of the Gaussian or Poisson. In general, in naïve Bayes the distribution is general not defined but where as in the multinomial distribution we defined the distribution of the function as multinomial. Here the different alpha is used and using the grid search CV we have obtained the alpha: '1', fit_prior: True as the best parameters and we have used partial fit to fit the data. The accuracy obtained is 94.69%.

c. **Stochastic Gradient Descent(SGD)**

Stochastic Gradient Descent Classifier is another classification technique where the stochastic optimization over the number of iterations are executed to minimize the loss functions. Here, the gradient of the loss is estimated each sample at a time and the model is updated along the way with a decreasing learning rate. This implementation works with data represented as dense or sparse arrays of floating point values for the features. The model it fits can be controlled with the loss parameter [1].The following combination of parameters from each set were used for the Grid Search:

'Loss': ["log", "hinge"], 'penalty': ["elasticnet", "l1", "l2"], 'alpha': [0.00001,0.0001, 0.001, 0.01,0.05, 0.1].

The 'log' loss implies Logistic Classifier and the Hinge loss implies Support Vector Machine (SVM) classifier. In our case, the data is heterogeneous and so the decision boundary of Logistic Regression being a linear curve won't be able to classify perfectly. On the other hand, the decision boundary of SVM is non-linear capable of classifying accurately. The results also support giving a value of 'hinge' loss. The penalty values are discussed above under the section "High Dimensional Challenge (Penalization)". Elastic net being a combination of both L1 and L2 has much higher complexity and so it is not a good penalization factor to depend on.

L2 regularization in general can eliminate some features better than the L1 regularization. The grid search results also showed supporting evidence giving “penalty: L2”. The alpha value indicated the regularization factor which also reflects the learning rate. The results of the confusion matrix give a value of 93.875% prediction accuracy.

d. Random Forest Classifier

Random Forest Classifier creates a set of decision trees from randomly selected subset of training set. It then aggregates the values from the different decision trees to obtain the result. This kind of approach avoids the noise associated with a single decision tree [2]. Here the ‘n_estimators’ is given as 10 indicating that the number of trees is 10. The following combination of different parameters from each set is used for the Grid search:

"max_depth": [3, None], "max_features": [1, 3, 10], "min_samples_split": [2, 3, 10], "min_samples_leaf": [1, 3, 10], "bootstrap": [True, False], "criterion": ["gini", "entropy"]

Here the criterion attribute decides the homogeneity of the dataset. In this, ‘gini’ indicates that the dataset is more homogeneous and ‘entropy’ indicates that the data is heterogeneous and performs classification based on the information gain. The grid search result also gave the criterion as ‘entropy’ indicating that information gain is the best way to deal with this dataset and proving that our dataset is heterogeneous without following pattern.

The bootstrap is a statistical resampling technique that involves random sampling of the dataset with replacement and are used to quantify the uncertainty in the model. Since we have many observations, we do not require bootstrapping here. Also, the gridsearch CV results gave “Bootstrap: False”. The ‘max_depth’ parameter indicates the depth at which the ensemble converges. Here ‘None’ indicates faster convergence and 3 indicates a little slower convergence. Also, the gridsearch CV results showed ‘None’ value indicating that faster convergence is better. The ‘min_samples_leaf’ is mainly used for smoothing the votes for regression. In this problem, since we deal mainly with classification, we get a minimum value of 1 in the Grid search. The ‘min_sample_split’ indicates the minimum number of samples required to split and predict using the Random Forest Classifier. Here a value of 10 indicates that it requires at least 10 samples to split the dataset. The ‘max_features’ indicates the number of features required for the best split. Here the grid search result obtained is 10 indicating that it needs 10 features to perform a split and predict efficiently. The accuracy obtained is 92.12%.

e. Multi-Layer Perceptron(MLP)

Multi-Layer Perceptron is a feedforward artificial Neural Network that uses the technique called Back-Propagation to train the samples. Here, there will be at least 3 nodes where except the input node all layers are neurons which uses a non-linear activation function [3] [4]. Inside the MLP classifier, we used the parameters “solver='adam', hidden_layer_sizes=30, alpha=1e-04”. The solver ‘adam’ refers to the Stochastic Gradient Descent Optimizer. This method is found to very efficient and they are used for problems with high number of features and memory. Our problem having both more number of samples and features can be solved computation efficiently using this method [6]. The number of hidden layers here is chosen as 30 and the alpha value of 1e -04 was chosen. The accuracy obtained by this model is 93.67%.

IV. Results and Interpretations

a. Topic Modelling

The datasets here are split into 4 topics using the Latent Dirichlet Allocation (LDA called as Topic Modelling. From the above figure, it can be concluded that the data is not homogeneous, and we need a non-linear classifier to predict even the less probable words with greater accuracy. The results obtained by the different classifiers are discussed below and the cluster plot is attached below:

b. Naïve Bayes Classifier (Gaussian)

In Gaussian Naïve Bayes classifier, the area under the curve for the ROC curve obtained was 0.84077. The values for the Precision, Recall and F1 score was 0.93, 0.71 and 0.77 respectively. The prediction accuracy of model 71.3 %.

c. Multinomial Naïve Bayes Classifier

In Multinomial Naïve Bayes classifier, the grid search results gave an alpha value as 1 and the fit_prior as true as the best parameters. We have got the area under the roc curve as 0.909. The values for the Precision, Recall and F1 score was 0.94, 0.94 and 0.94 respectively. The accuracy of model was 94.49 %.

d. Stochastic Gradient Descent(SGD)

In SGD, as discussed before, the gridsearch CV results gave us the best parameters and the reasons for the results are discussed above. For these parameters, we got an accuracy value of 93.875%. Here, the recall and F1 score obtained was 0.94 and 0.93 respectively. Since the grid search gave our loss function as ‘hinge’, we performed the SVM classification for our dataset. As the SVM is not a probabilistic classifier the Area under curve is not well defined hence we did not plot area under the curve for ROC and instead we have used the F1 score.

e. Random Forest

We got the best parameters as min_samples_split: 10, min_samples_leaf: 1, max_features: 10, max_depth: None, criterion: entropy, bootstrap: False. By utilizing these parameters to fit our dataset, we got the area under the ROC curve as 0.85, and the precision, recall and F1 score as 0.93, 0.92 and 0.90 respectively. The accuracy of model was 92.125 %.

f. Multi-Layer Perceptron

Using the Adams solver, we have got the area under the roc curve as 0.944. The values for the Precision, Recall and F1 score was 0.94, 0.94 and 0.94 respectively. The accuracy of model was 93.675 %.

V. Comparison of Models

The various results obtained are summarized using the table below,

Classifiers	AUC for ROC	Precision	Recall	F1 Score	Accuracy
Gaussian Naïve Bayes	0.84	0.93	0.71	0.77	71.3%
Multinomial Naïve Bayes	0.909	0.94	0.95	0.94	94.49%
SGD	-	0.94	0.94	0.93	93.875%
Random Forest	0.85	0.93	0.92	0.90	92.125%
Multi-Layer Perceptron	0.920	0.94	0.94	0.92	93.67%

From this table, it can be inferred that the accuracy for Naïve Bayes is only 71.3% which is comparatively lesser than other classifiers used and in multinomial naïve bayes we got an accuracy of 94.49% which show the difference between both the gaussian and multinomial is in the ability to deal with multiple categories used and multinomial distribution is still more efficient for the text data.

On the other hand, the performance of the other classifiers is also sound enough where SGD gives an accuracy of 93.875% and Random Forest gives an accuracy of 92.125%. The high accuracy of the SGD can be attributed to the 'hinge' loss function used. This makes use of an SVM classification method, where the decision boundary depends on the closest supporting vector. The high accuracy value of Random Forest is due to the nature of the classifier which aggregates many decision trees, which makes more accurate convergence resulting in accuracy value of 92.125%. Similarly, the accuracy of Multi-Layer Perceptron (MLP) is 93.67%. The hidden layer in MLP be a "distillation layer" that distills some of the important patterns from the inputs and passes it onto the next layer to see. It makes the network faster and efficient by identifying only the important information from the inputs leaving out the redundant information [7]. This makes them more accurate.

Precision tell how many selected items are relevant. This means out of the total predicted samples, how many of them were predicted accurately. On the other hand, recall tells how many relevant items are selected. This means in a sample space of relevant items, how many of them are predicted as true positives. This gives a measure of sensitivity. F1 score is an average of precision and recall. High F1 score is a characteristic of a good classifier. Here, it follows the same trend as the accuracy of the model where Multinomial Naïve Bayes has the highest value followed by Stochastic Gradient Descent, Multi-Layer Perceptron, Random Forest and Gaussian Naïve Bayes. AUC will also show how accurate the models. As we have observed that the area AUC values and the accuracy are depicting similar results expect for the multilayer perceptron because of greater true positives.

Learning Curves of different Models:

The learning curves for different models are provided in the Appendix section. The Naïve Bayes classifier showed a decreasing trend of both the training score and the cross-validation score. This can be attributed to the fact that unlike other problems, with the increase in number of training samples, the associated features also increase causing Overfitting. This causes lower bias and higher variance. Thus, Naïve Bayes has a better score in lower training samples.

On the other hand, the cross-validation score increases or remains stable for the other classifiers with the increase in the training samples. This shows better prediction performance, and this can also be validated with the accuracy of these classifiers. These classifiers were also able to tackle the increase in features with the training samples and in MLP we found that increasing furthermore samples would result in still more better accuracies.

VI. Conclusion

Thus, on performing various classification algorithms on python, we can conclude that Multinomial Naïve Bayes is best followed by SGD and Multi-Layer Perceptron are found to be the most efficient classifiers followed by Random Forest and Gaussian Naïve Bayes is not that efficient. Thus, when dealing with heterogeneous data like this, a classification model like Multinomial Naïve Bayes can be used to predict efficiently. On added benefit of the Multinomial Naïve Bayes is that the computation time is also comparatively lesser than other classifiers.

VII. Future Work

Further, we would like to extend this project by working on more number of observation and increase the features that are being utilized by implementing the spark or Hadoop. This aids in optimizing the computational time required by the program. Besides, we would like to try Recurrent Neural Network method to find the toxic comments present with still much better parameters to obtain better accuracy.

VIII. References

[1] http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html

[2] https://medium.com/machine-learning-101/chapter-5-random-forest-classifier-56dc7425c3e1

[3] Rosenblatt, Frank. x. Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms. Spartan Books, Washington DC, 1961.

[4] Rumelhart, David E., Geoffrey E. Hinton, and R. J. Williams. "Learning Internal Representations by Error Propagation". David E. Rumelhart, James L. McClelland, and the PDP research group. (editors), Parallel distributed processing: Explorations in the microstructure of cognition, Volume 1: Foundation. MIT Press, 1986.

[5] Siva Charan Reddy Gangireddy, Supervised Learning for Multi-Domain Text Classification (2016).

[6] Kingma. P. D, Ba. J. L.Adam- A Method for Stochastic Optimization. ICLR (2015).

[7] https://towardsdatascience.com/introduction-to-neural-networks-advantages-and-applications-96851bd1a207

IX. Appendix

Heat Map and Cluster Plot of Topic Modelling

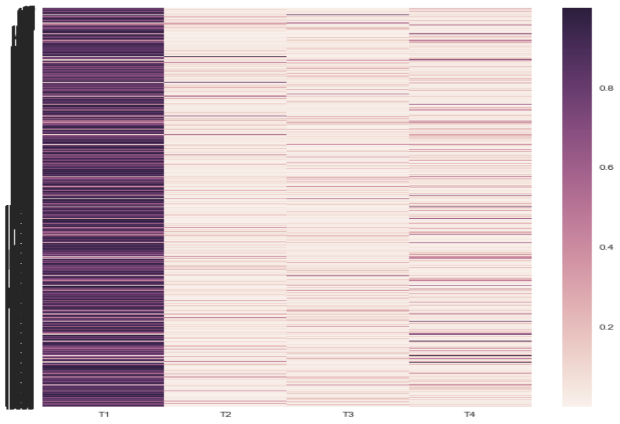


Fig -1 Heat Map

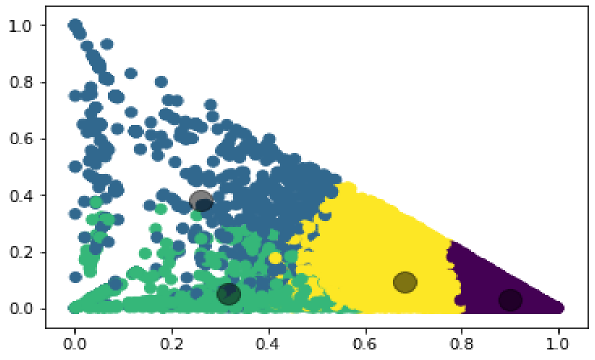


Fig -2 Cluster Plot

Gaussian Naïve Bayes Classifier

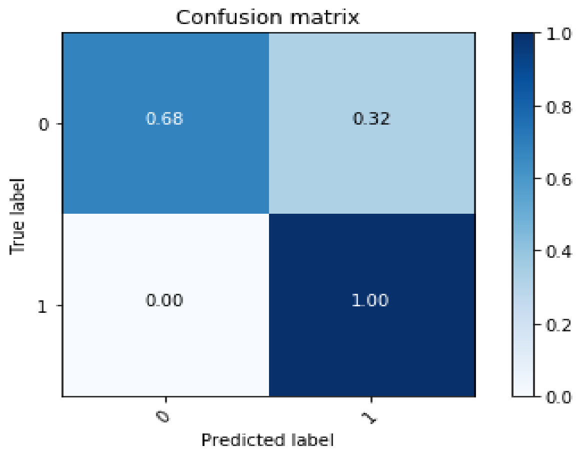


Fig -3 Confusion Matrix for Gaussian Naïve Bayes

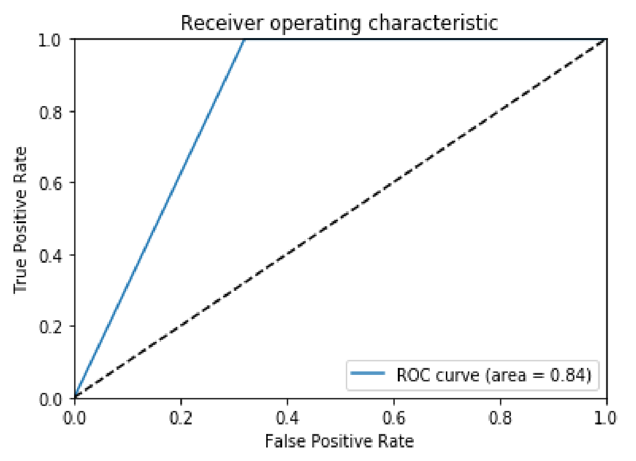


Fig -4 ROC Curve for Gaussian Naïve Bayes

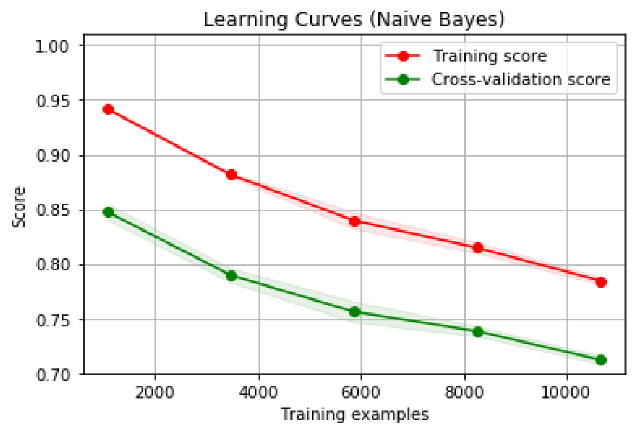


Fig -5 Learning Curve for Gaussian Naïve Bayes

Multinomial Naïve Bayes Classifier

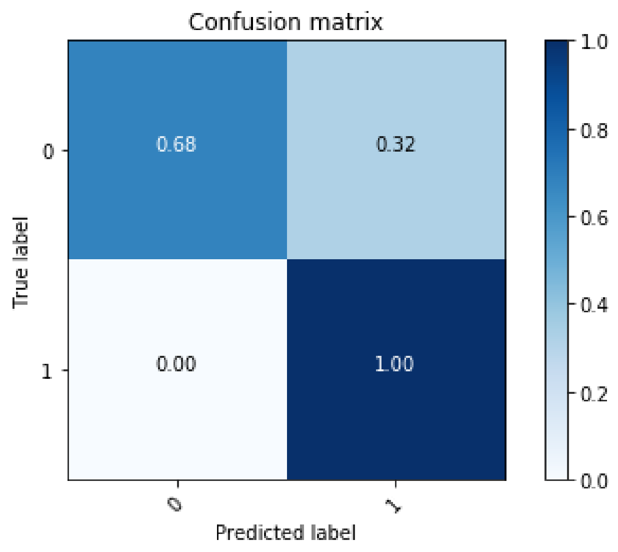


Fig -6 Confusion Matrix for Multinomial Naïve Bayes

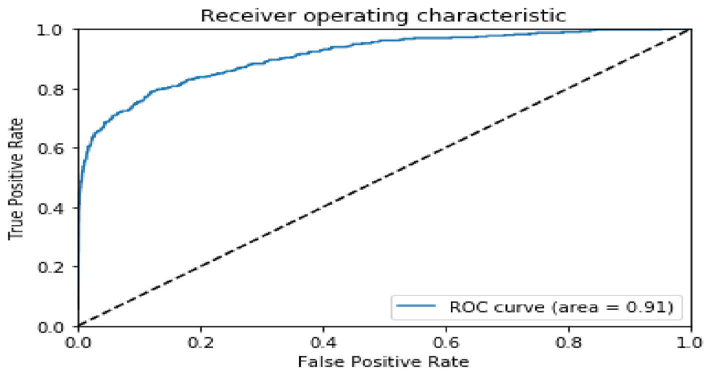


Fig -7 ROC Curve for Multinomial Naïve Bayes

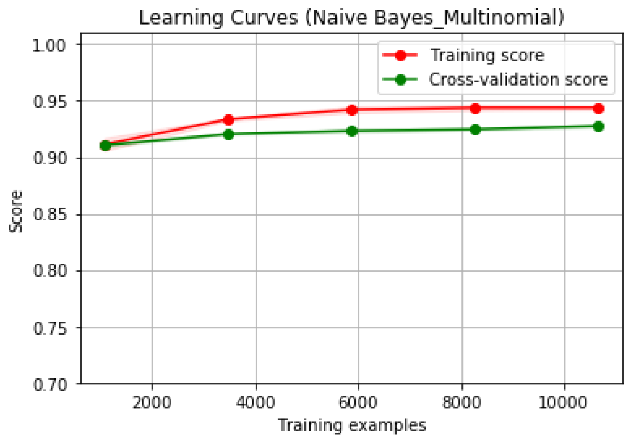


Fig -8 ROC Curve for Multinomial Naïve Bayes

Stochastic Gradient Descent (SGD) Classifier

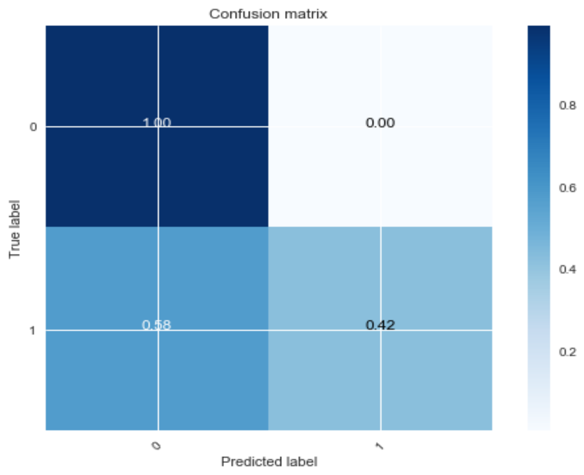


Fig -9 Confusion Matrix for SGD

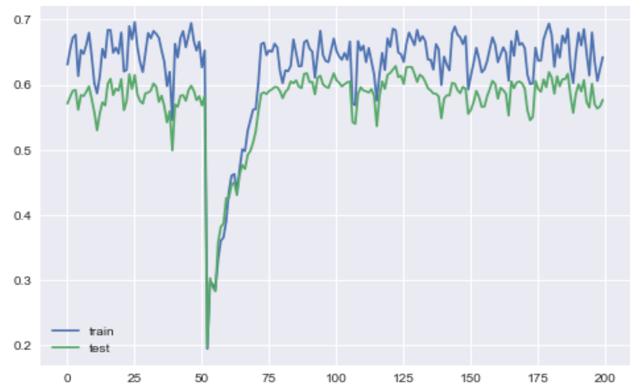


Fig -10 F1 score Graph for SGD

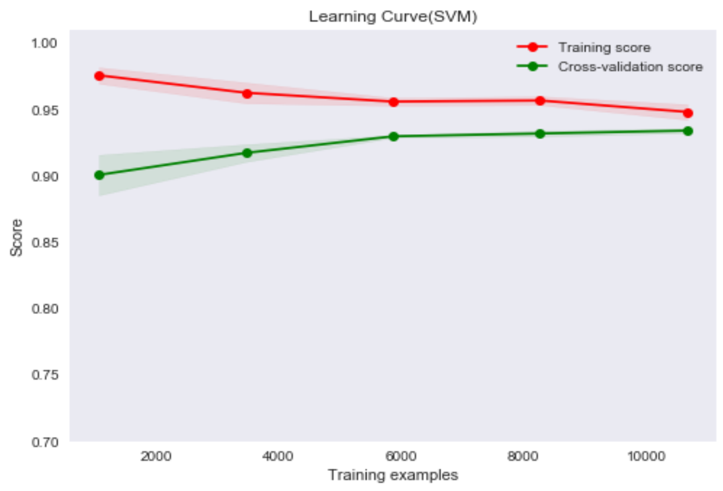


Fig -11 Learning Curve for SGD

Random Forest Classifier

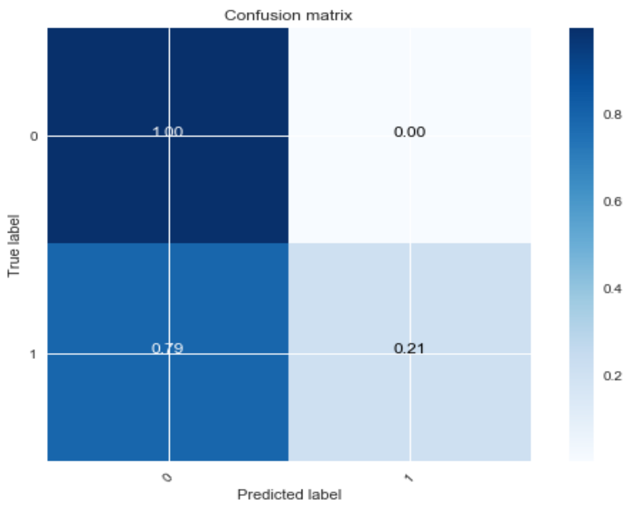


Fig -12 Confusion Matrix for Random Forest

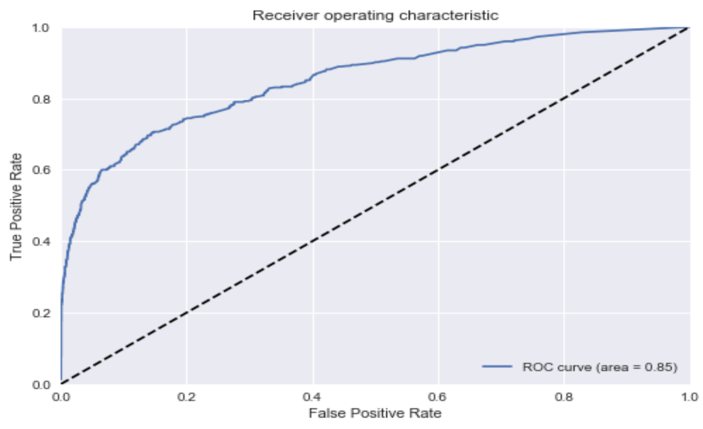


Fig -13 ROC Curve for Random Forest

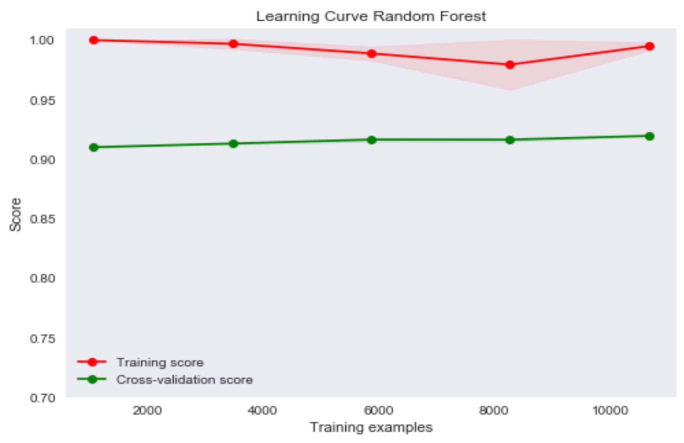


Fig -14 Learning Curve for Random Forest

Multi-Layer Perceptron (MLP)

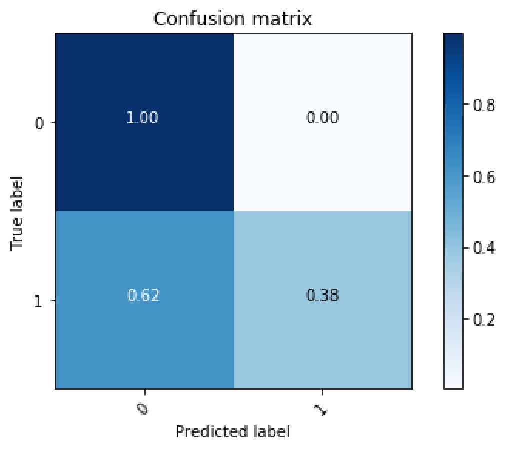


Fig -15 Confusion Matrix for Multilayer Perceptron

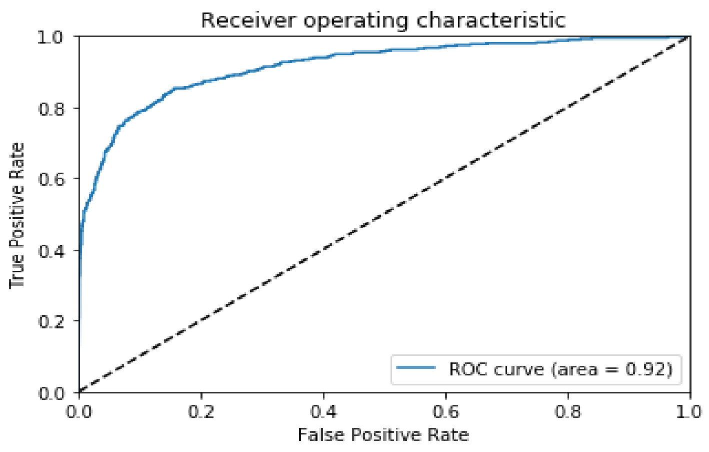


Fig -16 ROC Curve for Multilayer Perceptron

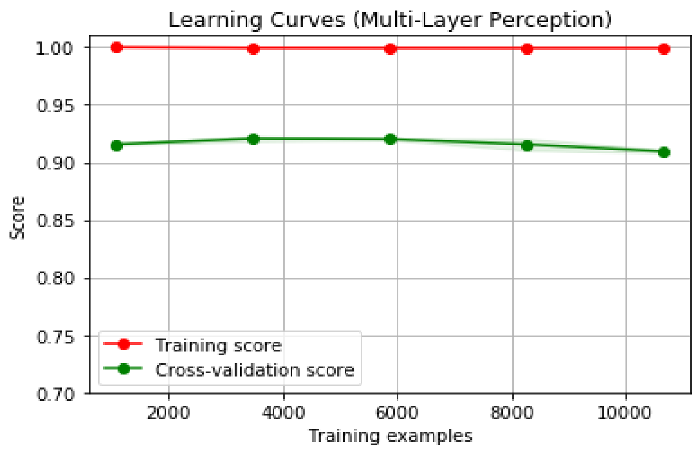
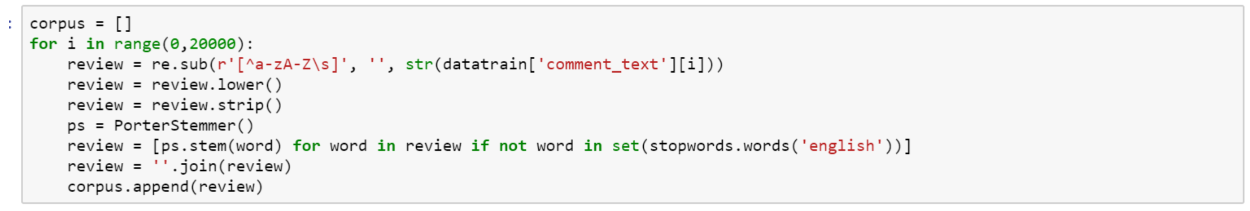
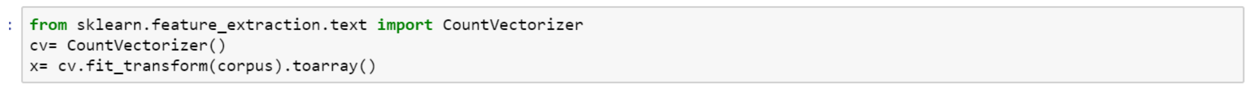


Fig -17 Learning Curve for Multilayer Perceptron

Some Important Codes(Full Codes are in the iPython file) :

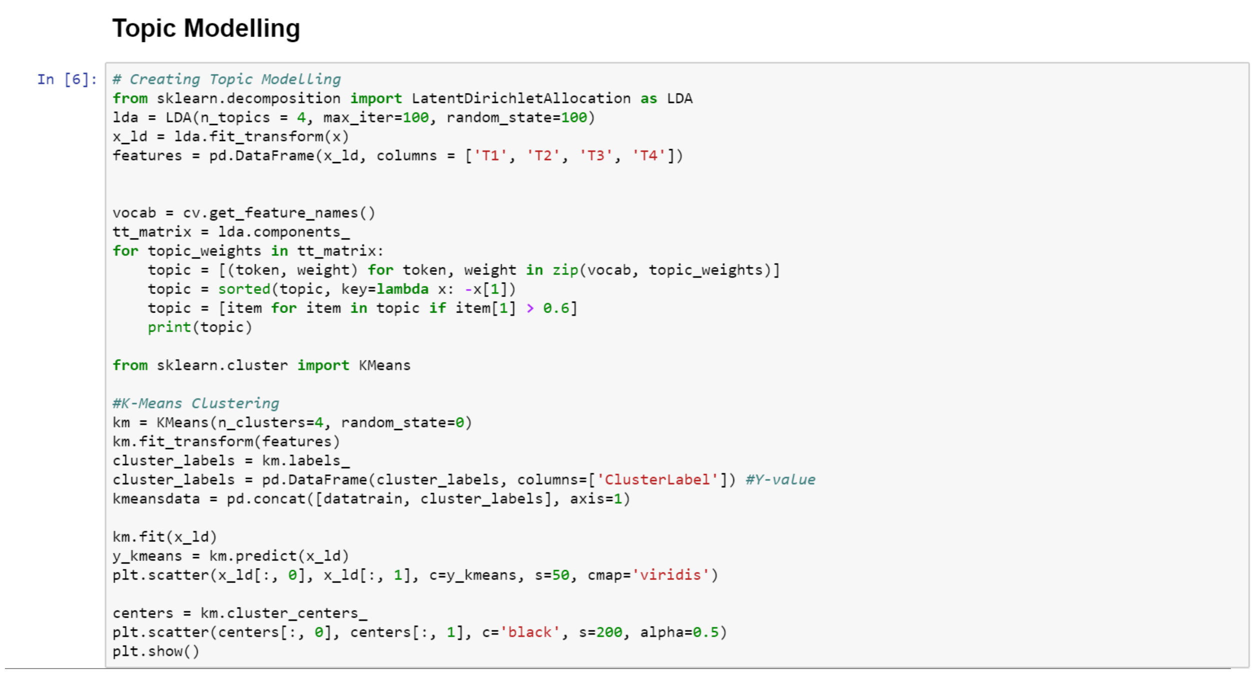


```
corpus = []
for i in range(0,20000):
    review = re.sub(r'[^a-zA-Z\s]', '', str(datatrain['comment_text'][i]))
    review = review.lower()
    review = review.strip()
    ps = PorterStemmer()
    review = [ps.stem(word) for word in review if not word in set(stopwords.words('english'))]
    review = ''.join(review)
    corpus.append(review)
```



```
from sklearn.feature_extraction.text import CountVectorizer
cv= CountVectorizer()
x= cv.fit_transform(corpus).toarray()
```

Fig -18 preprocessing codes



Topic Modelling

```
In [6]: # Creating Topic Modelling
from sklearn.decomposition import LatentDirichletAllocation as LDA
lda = LDA(n_topics = 4, max_iter=100, random_state=100)
x_ld = lda.fit_transform(x)
features = pd.DataFrame(x_ld, columns = ['T1', 'T2', 'T3', 'T4'])


vocab = cv.get_feature_names()
tt_matrix = lda.components_
for topic_weights in tt_matrix:
    topic = [(token, weight) for token, weight in zip(vocab, topic_weights)]
    topic = sorted(topic, key=lambda x: -x[1])
    topic = [item for item in topic if item[1] > 0.6]
    print(topic)

from sklearn.cluster import KMeans

#K-Means Clustering
km = KMeans(n_clusters=4, random_state=0)
km.fit_transform(features)
cluster_labels = km.labels_
cluster_labels = pd.DataFrame(cluster_labels, columns=['ClusterLabel']) #Y-value
kmeansdata = pd.concat([datatrain, cluster_labels], axis=1)

km.fit(x_ld)
y_kmeans = km.predict(x_ld)
plt.scatter(x_ld[:, 0], x_ld[:, 1], c=y_kmeans, s=50, cmap='viridis')

centers = km.cluster_centers_
plt.scatter(centers[:, 0], centers[:, 1], c='black', s=200, alpha=0.5)
plt.show()
```

Fig -19 Topic Modelling for clustering codes