Classification of E-mails

Betul Mescioglu
Department of Engineering,
Computing, and Mathematical Sciences
Lewis University
Chicago, USA
betulari@gmail.com

***Abstract*— Classification is a supervised machine learning technique that categorizes the data into correct classes based on the labeled data. In this project, we will attempt to classify e-mails sent to newsgroups into correct newsgroup classes with the help of several machine learning models.**

Keywords—Classification, machine learning.

I. Introduction

In this project, we have a collection of approximately 20,000 emails sent to 20 newsgroups. Our job is to classify them into correct newsgroups using machine learning algorithms. The following algorithms were applied:

- Multinomial Naïve Bayes
- Logistic Regression
- Support Vector Classification
- Stochastic Gradient Descent
- K-Neighbors
- Neural Networks

II. Description of the Dataset

The dataset consisted of training and testing folders. Both folders contained 20 newsgroup folders each. Each newsgroup folder comprises of e-mails sent to that newsgroup. In total, there were 18,846 emails. We formed a new dataset combining training and testing sets, performed data exploration and preprocessing on this set, applied bag of words model, then finally split it into training and testing datasets. [1]

Names of the newsgroups with their corresponding target values are:

0 - alt.atheism,
1 - comp.graphics,
2 - comp.os.ms-windows.misc,
3 - comp.sys.ibm.pc.hardware,
4 - comp.sys.mac.hardware,
5 - comp.windows.x,
6 - misc.forsale,
7 - rec.autos,
8 - rec.motorcycles,
9 - rec.sport.baseball,
10 - rec.sport.hockey,
11 - sci.crypt,
12 - sci.electronics,
13 - sci.med,
14 - sci.space,
15 - soc.religion.christian,
16 - talk.politics.guns,
17 - talk.politics.mideast,
18 - talk.politics.misc,
19 - talk.religion.misc

III. Data Cleaning

E-mails contained many punctuations and html tags that did not contribute to decision making. By using regex techniques, all unnecessary punctuations and html tags were removed and the text was converted to lower case. Newsgroup names were target values; they were replaced by numbers from 0 to 19. Finally, the data were randomized.

IV. Data Exploration

A. *Missing Data:*

Upon exploration, no missing e-mails were found. However, in newsgroup 6 which is originally named '*misc.forsale*', a duplicate e-mail was found, which was then eliminated from the dataset.

B. *Word Count of E-mails:*

When we counted the words in e-mails, we observed that the most frequent word count was 1000 or fewer. Very few of the emails contained more than 4000 words. On average, there were 1634 words in e-mails, and the e-mails containing the minimum and maximum words had 107 and 67560 words in them respectively.

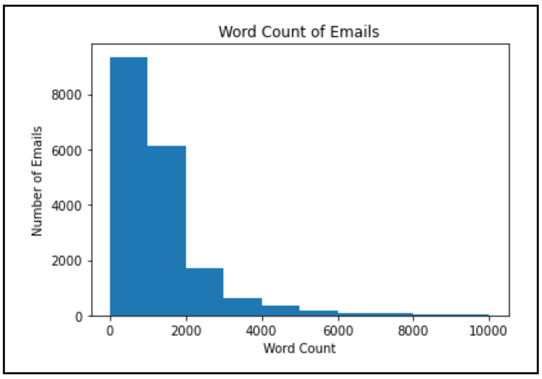


Figure 1: Word counts of e-mails.

C. *Length of E-mails:*

On average, the most lengthy e-mails for sent to Group 17, which corresponds to *'talk.politics.mideast'*.

'talk.politics.misc' and 'talk.religion.misc' are the second and third runner-ups respectively. This is not surprising, as the politics and religion are the two topics people are generally very passionate to talk about.

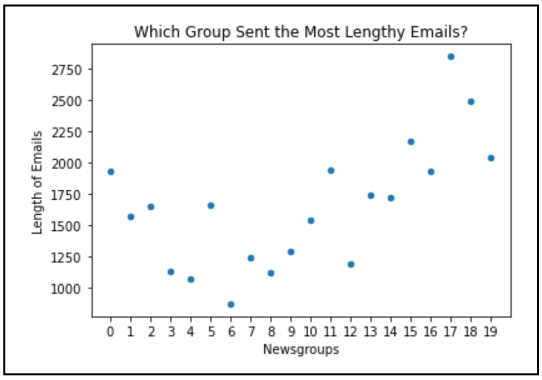


Figure 2. Plot of length of e-mails.

V. Data preparation

We need to convert each e-mail into a vector containing numbers to make them suitable for machine learning algorithms. We used Bag of Words approach, where each unique word in a text is represented by a number. We tokenized words in text, removed stop words (commonly used English words that do not contribute in decision making), and normalized words into their base forms (lemmas). In simpler terms, in vectorization process, first, we count how many times a word occurs in each document, then weight the counts, so that frequent tokens get lower weight. Finally, normalize the vectors to unit length. Each vector has as many dimensions as there are unique words in the document. (CountVectorizer() was used in this process.) In the end, we obtained a sparse matrix with dimensions of 18844 by 130656, former number corresponding to the number of e-mails, while latter corresponding to the number of unique words.

Finally, to perform machine learning, the data need to be split into 2 sets called training and testing sets. The algorithms will learn from the training set and make predictions on the testing set. We will then use several metrics to gauge how well the algorithms worked. As the data were already randomized previously, a manual splitting performed. First 12,000 e-mails were reserved for training, next 6,834 of them were held for testing. For illustrative purposes, we kept the last 10 e-mails. These 'never seen' e-mails were then fed to the algorithms to predict their classes.

VI. Metrics used to measure the performance of the models

Once, the models were created, several metrics were used to evaluate how well they classified the e-mails. Training and testing accuracies along with a classification report of the testing set's predictions were printed out. Classification report contains precision, recall, f1 score and support values for each class's predictions. A confusion matrix was also displayed. This very informative tool clearly shows how many e-mails were correctly classified and, if any misclassifications happened, in which class they were placed in. We also took into consideration of training loss where it is applicable.

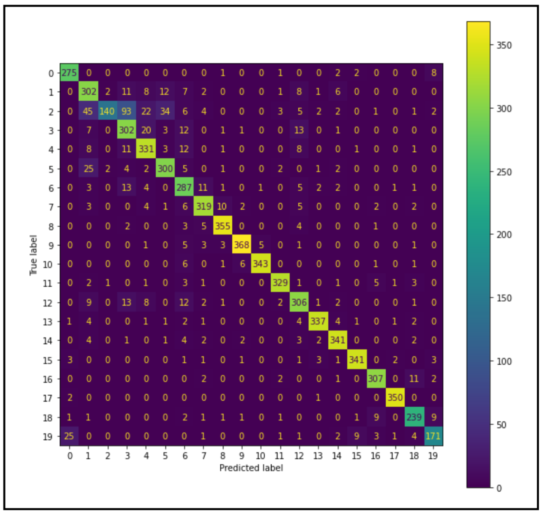
Accuracy measures the percentage of correct predictions. It does not determine the nature of error i.e., false positive or false negative. Precision and recall are better in determining the type of error. F1 score is the harmonic mean of precision and recall. Since f1 score combines both precision and recall into a single metric, we decided to print out average f1 scores of the algorithms as well. [2]

VII. Algorithms

A. *Multinomial Naïve Bayes (MNB):*

TABLE I. MNB Results

Training Accuracy	*Testing Accuracy*	*Training Log-Loss*	*F1 score*
98.63%	88.43%	0.21	0.88



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	275	0	0	0	0	0	0	0	1	0	0	1	0	0	2	2	0	0	0	8
1	0	302	2	11	8	12	7	2	0	0	0	1	8	1	6	0	0	0	0	0
2	0	45	140	93	22	34	6	4	0	0	0	3	5	2	2	0	1	0	1	2
3	0	7	0	302	20	3	12	0	1	1	0	0	13	0	1	0	0	0	0	0
4	0	8	0	11	331	3	12	0	1	0	0	0	8	0	0	1	0	0	1	0
5	0	25	2	4	2	300	5	0	1	0	0	2	0	1	2	0	0	0	0	0
6	0	3	0	13	4	0	287	11	1	0	1	0	5	2	2	0	0	1	1	0
7	0	3	0	0	4	1	6	319	10	2	0	0	5	0	0	0	2	0	2	0
8	0	0	0	2	0	0	3	5	355	0	0	0	4	0	0	0	1	0	0	0
9	0	0	0	0	1	0	5	3	3	368	5	0	1	0	0	0	0	0	1	0
10	0	0	0	0	0	0	6	0	1	6	343	0	0	0	0	0	1	0	1	0
11	0	2	1	0	1	0	3	1	0	0	0	329	1	0	1	0	5	1	3	0
12	0	9	0	13	8	0	12	2	1	0	0	2	306	1	2	0	0	0	1	0
13	1	4	0	0	1	1	2	1	0	0	0	0	4	337	4	1	0	1	2	0
14	0	4	0	1	0	1	4	2	0	2	0	0	3	2	341	0	0	0	2	0
15	3	0	0	0	0	0	1	1	0	1	0	0	1	3	1	341	0	2	0	3
16	0	0	0	0	0	0	0	2	0	0	0	2	0	0	1	0	307	0	11	2
17	2	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	350	0	0
18	1	1	0	0	0	0	2	1	1	1	0	1	0	0	0	1	9	0	239	9
19	25	0	0	0	0	0	0	1	0	0	0	1	1	0	2	9	3	1	4	171

Figure 3. Confusion Matrix of MNB.

B. *Logistic Regression:*

TABLE II. Logistic Regression Results

Training Accuracy	*Testing Accuracy*	*Training Log-Loss*	*F1 score*
100%	88.84%	0.02	0.89

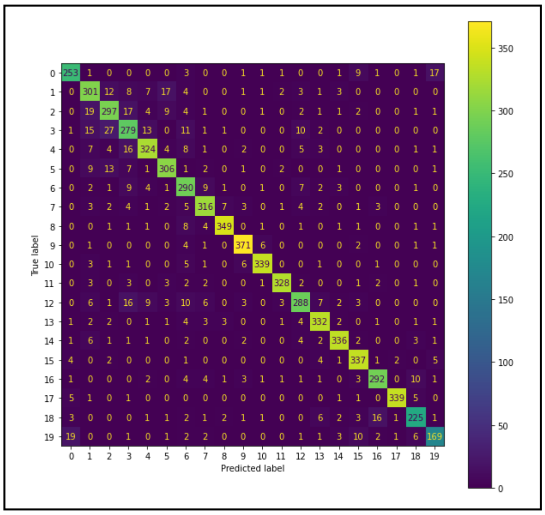


Figure 4. Confusion Matrix of Logistic Regression.

C. Support Vector Classification:

TABLE III. SUPPORT VECTOR CLASSIFICATION RESULTS

Training Accuracy	***Testing Accuracy***	***Training Log-Loss***	***F1 score***
99.76%	86.13%	0.08	0.86

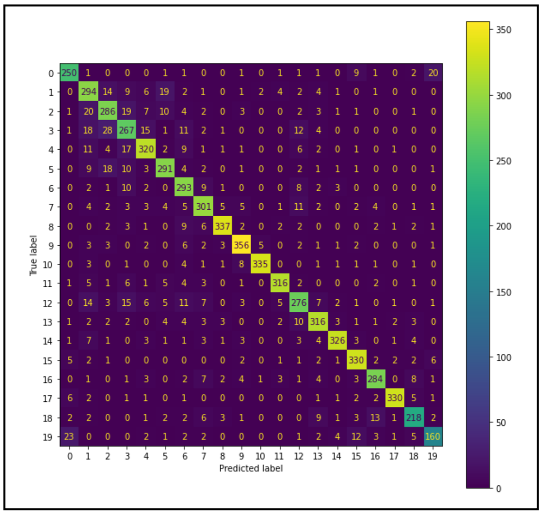


Figure 5. Confusion Matrix of Support Vector Classification

D. K Neighbors Classification:

TABLE IV. K NEIGHBORS RESULTS

Training Accuracy	***Testing Accuracy***	***Training Log-Loss***	***F1 score***
100%	61.56%	-	0.63

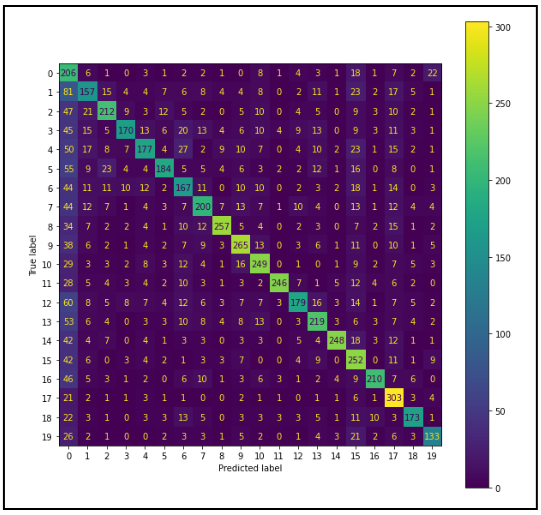


Figure 6. Confusion Matrix of K Neighbors.

E. Neural Networks:

TABLE V. NEURAL NETWORKS RESULTS

Training Accuracy	***Testing Accuracy***	***Training Log-Loss***	***F1 score***
99.99%	88.56%	0.187	0.88

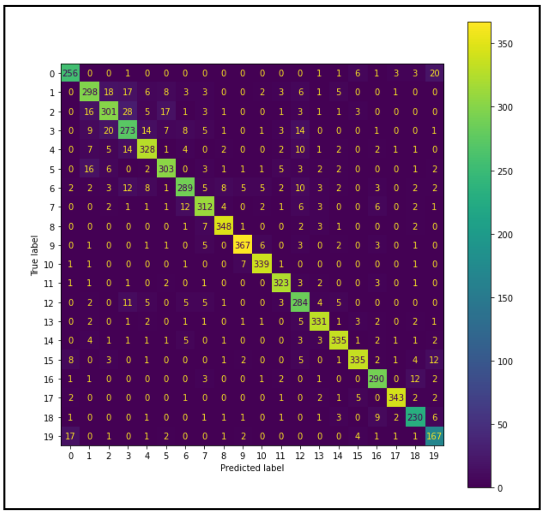


Figure 7. Confusion Matrix of Neural Networks.

VIII. CONCLUSION

TABLE VI. RESULTS

Model	***Training Accuracy***	***Testing Accuracy***	***Training Log-Loss***	***F1 score***
Multinomial Naïve Bayes	98.63%	88.43%	0.21	0.88
Logistic Regression	100%	88.84%	0.02	0.89
Support Vector Classification	99.76%	86.13%	0.08	0.86
K Neighbors	100%	61.56%	-	0.63
Neural Networks	99.99%	88.56%	0.187	0.88

Obviously, Multinomial Naïve Bayes, Logistic Regression, and Neural Networks were the best performing algorithms with more than 88% testing accuracy. They are followed by Support Vector Classification. The worst performing was K-Neighbors. It's probably due to the curse of dimensionality. K-Neighbors performs best with fewer number of features. If the number of features is high, it requires more data. When there is more data, it is prone to overfitting, which is exactly what we see here with 100% training accuracy and only 62% testing accuracy. In our case we have 130,656 features, which is extremely large in size. Overall, Logistic Regression proves to be the best choice as it also provides higher f1 score on testing set and lower loss on training set.

REFERENCES

[1] Qwone.com, '20 Newsgroups', 2008. [Online]. Available: http://qwone.com/~jason/20Newsgroups/. [Accessed: 20- Oct- 2022].

[2] J. Korstanje, 'The F1 Score', 2021.[Online] Available: https://towardsdatascience.com/the-f1-score-bec2bbc38aa6 [Accessed: 20- Oct- 2022].