IDS 575 Machine Learning Statistics
Group 16 Final Report

Online News Popularity Classification

Team members
Anviksha Gupta (UIN: 659709167)
Golsa Riahipour (UIN: 654270437)
Raymond Sutanto (UIN: 674387992)

Abstract

In this paper, we show that a news article's popularity can be deconstructed through a grading system, which is further explored to elicit impactful insights for relevant stakeholders. Moreover, we reveal a reliable multiclass classification machine learning model, which can be used to predict the popularity. Naive Bayes was used as the baseline, which is compared with models such as logistic regression, KNN, SVM, Adaboost, and k-means based on certain metrics. This paper will show how logistic regression is identified as the best model among all options, which is a valuable tool for accurate classification.

Introduction

With the rapid growth of online news sources, accurately predicting the popularity of a news article has become challenging. Machine learning algorithms for online news popularity classification have been developed to address this need. Understanding the factors that influence the popularity of news articles is crucial for news organizations, content providers, and marketers. This information can aid in content creation, audience targeting, and advertising campaigns. However, predicting news popularity is complex due to the vast amount of information available, the dynamic nature of news subjects, and the changing online environment. Factors such as the content of the article, timing of publishing, news source, tone, and social media discussion can all impact a news article's popularity. Moreover, news articles can vary widely in popularity, ranging from virally unpopular to highly popular.

To effectively classify the popularity of online news, advanced machine learning algorithms are needed to process and analyze diverse data sources and accurately predict the popularity of news stories. In this study, we explore the dataset to collect any interesting insights about the dataset. We then compare the performance of several machine learning algorithms for multiclass classification, starting with Naive Bayes as the baseline model, and then exploring more simple and advanced techniques such as Logistic Regression, k-Nearest Neighbors (KNN), Support Vector Machines (SVM), Adaboost, and k-means. Finally we evaluate several metrics alongside the provision of confusion matrices.

Related work

In order to better understand this problem, we reviewed other relevant research projects in this field.

1. **A Proactive Intelligent Decision Support System for Predicting the Popularity of Online News :** In this project the authors aim to forecast the popularity of news on social media platforms. They used several different machine learning models to predict the popularity of online news articles such as random forest, adaboost, support vector machine and KNN[1].
2. **Predicting the volume of comments on online news stories:** in this research paper, their goal is to identify the best approach for predicting the volume of comments on online news using machine learning models. They used different features such as length of the article, number of unique words and time of the publication[2].
3. **Predicting popular messages in Twitter:** this paper provides insights into factors that contribute to the popularity of messages on twitter. The authors examined different features that contribute to the popularity of a tweet such as the content, number of mentions, time of publication, number of followers and mainly number of tweets. They used a classification approach to predict the popularity of messages. For binary class they used PR and F1 score, and for multi-class classification they reported accuracy of each class[3].

Grading system

Our research endeavors are supported by a comprehensive dataset comprising 39,797 instances and 61 attributes, including factors such as the number of words, title, number of images, and number of videos. The focal point of our analysis is the target variable, which is the number of shares each online article receives. This key variable will be instrumental in determining the popularity of each news article in our study.

To streamline our analysis and classification, we will develop a grading system with five distinct bins that reflect varying ranges of the number of shares. This system will enable us to categorize news articles based on their level of popularity and delve into patterns and trends associated with each category. The grades will be labeled as follows:

a. Grade A for extremely popular news articles
b. Grade B for highly popular news articles
c. Grade C for fairly popular news articles
d. Grade D for slightly popular news articles
e. Grade E for unpopular news articles

Our approach to classifying news articles based on popularity aims to provide a nuanced understanding of how different articles are received by online audiences. By examining the factors that contribute to news article popularity, we hope to identify key features that can drive engagement and enable content creators and publishers to produce more compelling content. Our grading system will serve as a framework to capture the dynamic nature of online news consumption and shed light on the factors that influence the sharing behavior of readers, ultimately contributing to a deeper understanding of news article popularity in the digital age.

Exploratory Data Analysis

We started the exploratory data analysis by investigating the distribution of articles published. In this case, we observed the count of articles published across different days of the week. The analysis revealed that there is a notable disparity in the number of articles published between weekdays and weekends.

According to Figure 1, the count of articles published on weekdays is significantly higher than those published on weekends. This finding could be attributed to several factors, including reader engagement and publisher strategies. It is common for people to have more leisure time during weekends and holidays, which could lead to a decrease in news consumption. Therefore, publishers may opt to release more articles during weekdays to increase the likelihood of reader engagement.

Furthermore, releasing articles on weekdays may also align with publisher strategies aimed at maximizing reader activity. For instance, publishers may schedule articles to be released during peak traffic hours, such as early in the morning or during lunch breaks, when readers are more likely to engage with the content.

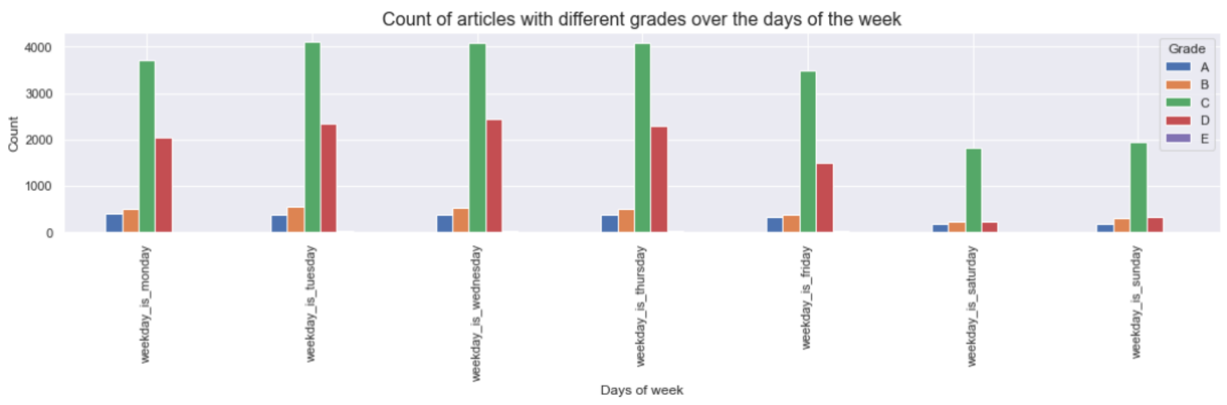

Figure 1

Our next finding, as illustrated in our analysis stated in Figure 2, suggest that there are discernible patterns in the popularity of content across different popularity grades and categories. Specifically, when examining the top two popularity grades, 'A' and 'B', we found that both the entertainment and technology industries had the highest number of articles. This indicates that these two topics are particularly successful in capturing a significant share of the most popular content across various platforms and media outlets. On the other hand, when we looked at the third and fourth popularity grades, 'C' and 'D', we found that the technology and world categories had the highest number of articles.

This particular analysis highlights the importance of considering both the category and popularity grade of content when examining patterns in content popularity. To build upon this analysis, news companies could conduct further research to understand why certain topics (entertainment and technology) are more successful in capturing audience attention than others. This could involve surveys or focus groups to gain insights into audience preferences and behavior.

Another strategy to respond to this analysis could be for media outlets and content creators to use this information to inform their content strategy. For example, if they want to target a wider audience, they could focus on creating content in the entertainment and technology categories, as these categories are more likely to be successful in capturing a significant share of the most popular content across various platforms. Furthermore, these firms should focus on improving the quality of content related to technology and world news, which tend to dominate the lower popularity grades.

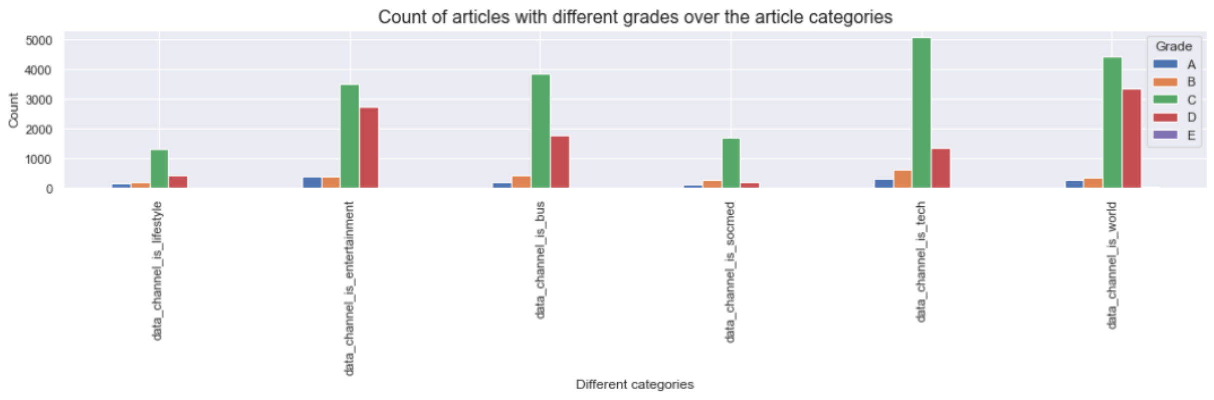

Figure 2

Our final exploratory data analysis was conducted specifically for grade ‘A’ articles since we wanted to know how a high quality article is determined. As shown in figure 3, it was revealed that the number of words used in these articles varied widely, with some as low as 100 words and others as high as 4,000 words. Despite this variation, the majority of the articles analyzed fell within the range of 100 to 2000 words. This particular finding implies that there is no definitive connection between the length of an article and its popularity. In other words, an article's length is not a decisive factor in determining how often it will be shared or how well it will perform in terms of engagement. This is because both shorter and longer articles can achieve a high number of shares, and the length of the article itself may not necessarily be the primary reason for its popularity.

In the world of online publishing, there can often be a common misconception that longer articles equate to higher readership and engagement. However, our analysis shows that the length of an article may not be the sole determining factor of its popularity. Instead, other crucial elements such as the quality of the writing, the relevance of the topic, and the appeal of the content to the intended audience can significantly impact the success of an article.

For instance, a well-written article that resonates with its readers, and provides valuable insights or solutions to a problem, is more likely to gain traction and generate interest than a lengthy article that lacks substance. Similarly, an article that addresses a trending or relevant topic that resonates with its target audience is more likely to be shared and discussed on social media platforms. Furthermore, the appeal of the content to the intended audience is essential in driving engagement and shares. A content piece that is engaging, informative, and entertaining is more likely to capture and hold the reader's attention, leading to higher engagement levels and a broader audience reach.

Therefore, instead of solely focusing on the length of their articles, authors and publishers should prioritize creating high-quality content that is both relevant and engaging to their target audience. By doing so, they can increase the likelihood of their articles being shared, discussed, and read by a broader audience, leading to greater success and recognition in the world of online publishing.

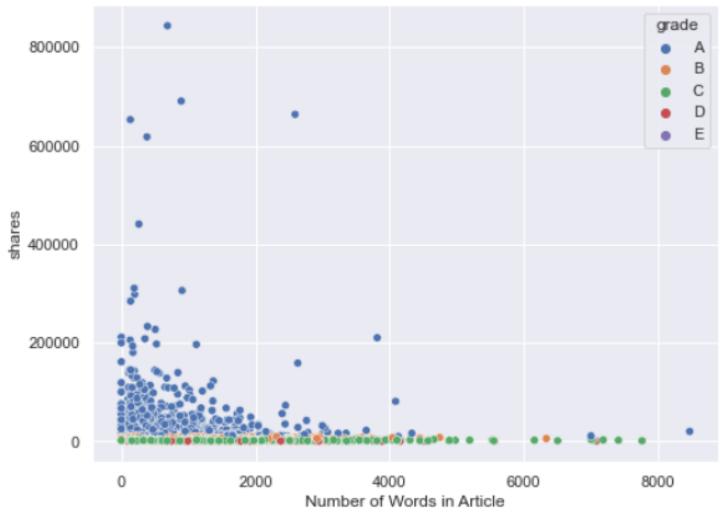

Figure 3

Model Analysis

In this section we expand our machine learning models that are used for multiclass classification:

I. Naive Bayes (Baseline Model):

The naive bayes model is a simple and fast classification model which is based on the bayes theorem. Despite its simplicity this model surprisingly performs well in many cases. Naive Bayes model works by calculating the conditional probability. The probability of class A given feature vector B can be calculated as:

$$P(A|B) = P(B|A) \times \frac{P(A)}{P(B)}$$

Where $P(A|B)$ is the likelihood.

<u>Pseudo code for Naive Bayes:</u>

```
Input:
    - Training dataset T
    - Feature values F = (f1, f2, f3, ..., fn) // values of the predictor variables in the testing
      dataset
Output:
    - Predicted class for the testing dataset
Steps:
    1) Read the training dataset T
    2) For each class c in T:
        a) Calculate the prior probability P(c)
        b) Calculate the mean and standard deviation of the predictor variables for class c
    3) For each class c in T:
        a) Calculate the likelihood L(c) for class c using the following steps:
            i) Initialize L(c) = P(c)
            ii) For each feature value fi in F:
                (1) Calculate the probability of fi using the Gaussian density equation
                (2) Multiply L(c) by the calculated probability of fi
    4) Determine the class with the greatest likelihood
    5) Return the class with the greatest likelihood as the output
```

II. Logistic regression:

This model is a linear model often used for classification problems. Logistic regression belongs to the family of supervised machine learning models which uses mathematics to find the relationships between data factors.

Pseudo code for logistic regression:

```
Input:
    - x: the input data matrix, with n rows (samples) and m columns (features)
    - y: the target vector, with n elements (binary classes: 0 or 1)
    - alpha: the learning rate
    - num_iterations: the number of iterations for gradient descent
Output:
    - theta: the learned parameters, a (m+1) dimensional vector
Steps:
    1) Initialize the weight vector w = [w1, w2, …, wm] to small random values
    2) Set the learning rate alpha
    3) Repeat until convergence (or a maximum number of iterations):
        a) Compute the hypothesis function h(x) = 1 / (1 + exp(-w.T * x))
        b) Compute the cost function J(w) = (-1/m) * sum(y*log(h(x)) + (1-y)*log(1-h(x)))
        c) Compute the gradient vector of the cost function with respect to each weight:
           dwj = (1/m) * sum((h(x) - y) * xj)
        d) Update the weights: wj = wj - α * dwj
    4) Return the weights vector w.
```

III. K-nearest neighbor (KNN):

KNN is an algorithm that estimates the class label based on the majority vote of the k-nearest neighbors in the feature space.The algorithm computes the distance (e.g., Euclidean distance) between the instance and all training instances, selects the k instances with the smallest distances. Then it determines the majority class among these neighbors.

Pseudo code for KNN:

```
Input:
    - Training dataset D = {(x1, y1), (x2, y2), ..., (xn, yn)}, where xi is the feature vector of the
      i-th training example, and yi is its corresponding class label.
    - Test example x_test
    - The number of nearest neighbors K
Output:
    - The predicted class label y_test for x_test
Steps:
    1) For each training example (xi, yi) in D:
        a) Compute the distance between x_test and xi, e.g. Euclidean distance, Manhattan
           distance, etc.
        b) Store the distance and the class label yi in a list
    2) Sort the list by distance in ascending order.
    3) Select the first K elements from the sorted list.
    4) Compute the frequency of each class label among the K nearest neighbors.
    5) Assign the class label with the highest frequency as the predicted class label y_test.
    6) Return y_test.
```

IV. Support vector machines (SVM):

SVM is a machine learning model which is capable of handling both linear and non-linear classification and regression. SVM aims to find the optimal hyperplane that maximizes the margin between classes.

Pseudo code of SVM

```
Input:
    - Dataset D with instances X and labels Y
    - Positive bags P with multiple instances
Output:
    - (w, b) as the SVM solution
Steps:
    1) Initialize:
        a) Set Y_i = Y[i] for all i in I (instance indexes)
    2) Repeat (until imputed labels have no more changes):
        a) Compute SVM solution (w, b) for the dataset with imputed labels
        b) Compute outputs f_i = (w, x_i) + b for all x_i in positive bags
        c) Update labels:
            i) Set Y_i = sgn(f_i) for every i in I, where Y_i = 1 (positive bags)
            ii) If (sum of L_i (L_i + Y_i) / 2) == 0:
                (1) Compute i* = arg max_i {L_i * f_i}
                (2) Set Y_i* = 1
```

V. Adaboost:

The Adaptive Boosting (Adaboost) algorithm is a powerful ensemble learning method that combines multiple weak classifiers to construct a strong classifier.

Pseudo code for Adaboost:

```
Input:
    - A training dataset D = {(x1,y1), (x2,y2), ..., (xn,yn)}, where xi represents the i-th instance
      and yi represents its corresponding label
    - The number of weak classifiers to use, T
    - A set of weak classifiers h1(x), h2(x), ..., hT(x)
    - A vector of weights for each instance in the dataset, w1, w2, ..., wn
```

Output:

- The final strong classifier, H(x), as the weighted combination of the T weak classifiers: $H(x) = sign(sum(\alpha t * ht(x)))$
- Note: The sign() function returns 1 if the argument is positive, -1 if the argument is negative, and 0 if the argument is zero.

Steps:

```
1) Initialize the weights for each instance in the dataset: $wi = 1/n$, where n is the number of instances in the dataset
2) For t = 1 to T:
    a) Train a weak classifier ht(x) using the weights wi for each instance
    b) Calculate the error rate of ht(x): $\varepsilon t = sum(wi if ht(xi) != yi else 0) / sum(wi)$
    c) Calculate the weight for ht(x): $\alpha t = ln((1-\varepsilon t)/\varepsilon t)$
    d) Update the weights for each instance in the dataset:
        i) $wi = wi * exp(\alpha t if ht(xi) != yi else -\alpha t)$
    e) Normalize the weights so that they sum to 1: $wi = wi / sum(wi)$
```

VI. K_means:

K-means is a popular unsupervised machine learning algorithm for clustering data points into 'K' distinct groups based on their features. The algorithm iteratively assigns each data point to the nearest centroid and updates the centroids until convergence.

<u>Pseudo code for Kmeans:</u>

Input:

- Training dataset X = {x1, x2, ..., xn}, where xi is a d-dimensional feature vector.
- The number of clusters K.
- The maximum number of iterations T.

Output:

- A set of K cluster centroids mu = {mu1, mu2, ..., muK}.

Steps:

```
1) Randomly initialize the K cluster centroids mu = {mu1, mu2, ..., muK}.
2) Repeat until convergence (or a maximum number of iterations T):
    a) Assign each training example xi to the nearest cluster centroid mu_j, using the Euclidean distance or other distance metric.
    b) Update each cluster centroid mu_j to be the mean of the feature vectors assigned to it:
        i) $mu\_j = (1/|S\_j|) * sum(xi)$, for all i such that xi is assigned to mu_j.
3) Return the set of cluster centroids mu.
```

Note: |S_j| denotes the number of training examples assigned to the j-th cluster centroid mu_j.

Experimental Results

The dataset was first split into training and testing sets using a 70-30 split ratio. Next, the features were standardized using the StandardScaler from scikit-learn. Thus, several classification algorithms, as already mentioned, were trained on the preprocessed dataset. The performance of each model was evaluated using various performance metrics, including accuracy, precision, recall, and F1-score. Furthermore, confusion matrices were generated to visualize the models' performance across different classes.

We employed a Naive Bayes classifier to predict article popularity based on various features. The model demonstrated a relatively strong performance, achieving an overall accuracy of approximately 77.48%. The results offer valuable insights into the model's capability to classify articles into their respective popularity grades (A, B, C, D, and E). The model's performance varied across different popularity grades, with some grades exhibiting a trade-off between precision and recall. Despite these variations, the overall results provide a solid baseline for our analysis. These findings can be utilized as a benchmark for comparing the performance of other machine learning models when applied to the problem of predicting article popularity. Figure 4 shows the confusion matrix of this baseline model.

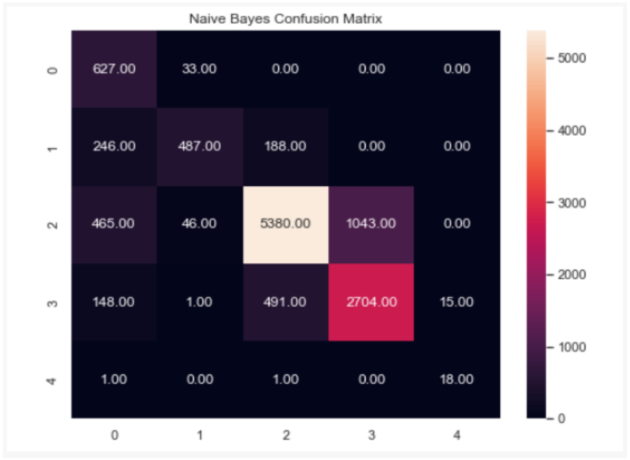

Figure 4

Our analysis also involved the application of a Logistic Regression classifier to predict article popularity based on the given features. The model achieved an impressive overall accuracy of approximately 95.65%, indicating a superior performance in classifying articles into their respective popularity grades (A, B, C, D, and E) compared to the Naive Bayes model.

Furthermore, the logistic regression classification report provides further insights into the model's performance for each popularity grade. For grade A articles, the model demonstrated high precision (99%) and recall (95%), indicating an accurate and reliable classification. Similarly, for grades B and C articles, both precision (96% and 97%, respectively) and recall (86% and 97%, respectively) were reasonably high, further showcasing the model's effectiveness. The macro average F1-score of 0.76 and the weighted average F1-score of 0.96 reveal that the model's performance was generally consistent across different popularity grades. A further illustration of this model can be seen in the respective confusion matrix, as shown in Figure 5.

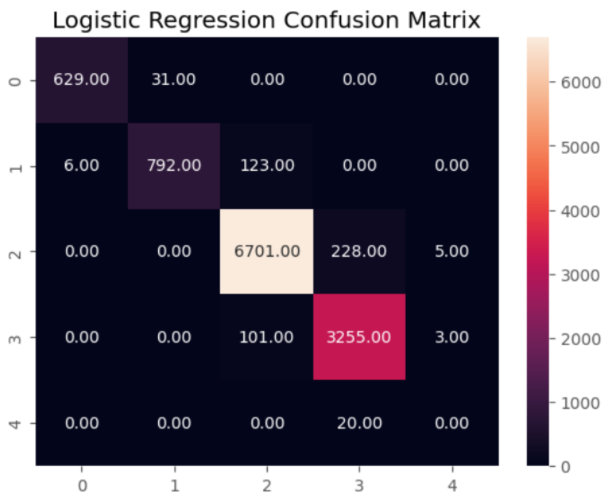

Figure 5

The confusion matrix given in Figure 6 for KNN has values that denote the number of instances that belong to a particular predicted class. For example, there were 131 instances of class 0 that were predicted to be class 1, and there were 5325 instances of class 2 that were predicted to be class 2. The diagonal elements represent the number of instances that were classified correctly. In this matrix, class 2 has the highest number of correctly classified instances, with 5325 instances being correctly classified as class 2, while class 4 has the lowest number of correctly classified instances, with only 10 instances being correctly classified as class 4.

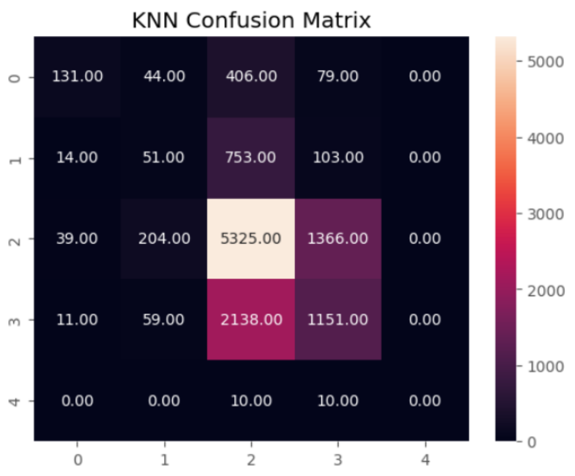

Figure 6

Looking at the matrix in Figure 7, it appears that the SVM has achieved high accuracy, with mostly high numbers on the diagonal (true positives), and low numbers off the diagonal (false positives and false negatives). However, it seems that the SVM struggles with class 3, where it incorrectly predicted 1858 instances as class 2, and 724 instances as class 4. This confusion may indicate that class 2 and class 4 share some similarities with class 3, and the SVM may need further optimization to differentiate between these classes more accurately.

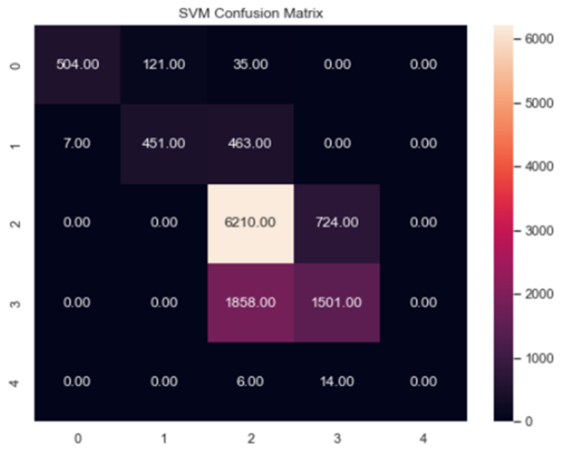

Figure 7

Figure 8 illustrates the adaboost confusion matrix. This model achieved an accuracy of approximately 94.28%, indicating its ability to correctly predict the grade of an online news article about 94.28% of the time. The model's performance was exceptional for classes C and D, with near-perfect precision, recall, and F1-scores. However, it struggled with classes A and E, failing to correctly predict any instances, and showed moderate performance for class B. The confusion matrix further confirmed the classifier's strengths and weaknesses across different classes.

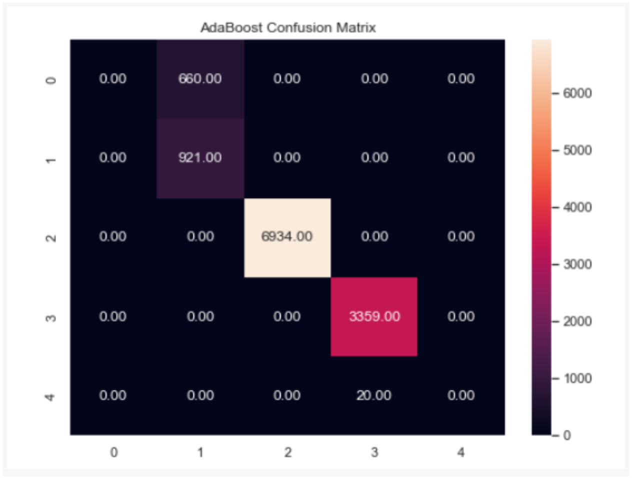

Figure 8

In our project, we used the K-means clustering algorithm to categorize articles into distinct groups based on their features, using k=4 to form four separate clusters. To evaluate the quality of the clustering, we calculated the Silhouette Score, which is a measure of how well-separated the clusters are. For our K-means clustering, we obtained a Silhouette Score of approximately 0.682. This score suggests that the clustering solution generated by the K-means algorithm is reasonably well-defined, with a good level of separation between the clusters.

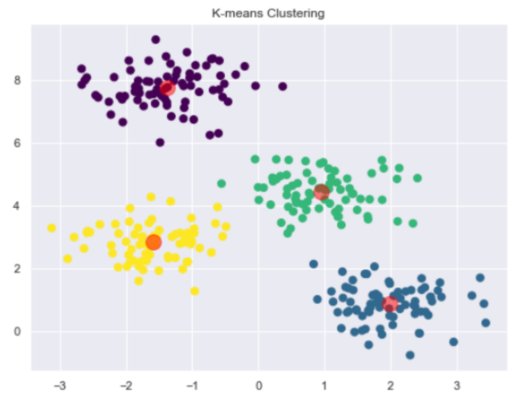

Figure 9

Algorithms	Accuracy	Precision	Recall	F1-score
Naive_Bayes	0.77	0.69	0.79	0.7
Logistic Regression	0.95	0.77	0.75	0.76
KNN	0.56	0.37	0.27	0.29
SVM	0.73	0.63	0.52	0.56
Adaboost	0.94	0.52	0.60	0.55

Table 1: summary of results

The table represents the performance of five different machine learning algorithms, namely Naive Bayes, Logistic Regression, KNN, SVM, and Adaboost, on an unknown task. The performance is measured using four different evaluation metrics, namely accuracy, precision, recall, and F1-score.

Based on the table, it can be observed that Logistic Regression and Adaboost algorithms perform well with an accuracy of 0.95 and 0.94 respectively. However, the KNN algorithm performs poorly with an accuracy of only 0.56. In terms of precision, Logistic Regression has the highest precision of 0.77, whereas, Adaboost algorithm has the lowest precision of 0.52.

Naive-Bayes has the highest recall of 0.79, whereas KNN has the lowest recall of only 0.27. The F1 score metric considers both precision and recall, and Logistic Regression has the highest F1 score of 0.76, whereas KNN has the lowest F1 score of 0.29.

From the analysis of the table, it seems that the logistic regression algorithm performs the best overall with the highest accuracy, precision, and F1-score, indicating that it has the best balance between correctly identifying positive and negative cases. Also, it can be concluded that the failure mode of KNN as the lowest performing model is likely due to its inability to capture the complex relationships between the input features and the target variable. This is because the KNN algorithm relies on finding the nearest neighbors of each instance in the data, which may not be suitable for datasets with high dimensionality or complex relationships.

Conclusion

To conclude, our project predicts news article popularity using machine learning algorithms. Initially we analyzed factors like word count, images, and videos to create a grading system with five bins. Moreover, we performed exploratory data analysis, which showed that weekdays have more publications than weekends, entertainment and technology are the highest graded categories, and quality, relevance, and appeal matters more than length. Finally, we compared several machine learning models, including Naive Bayes as the baseline model, Logistic Regression, k-Nearest Neighbors, Support Vector Machines, Adaboost, and K-means, which revealed that Logistic Regression had the best performance model, with the highest accuracy, precision and F1-score among all models.

There are various ways that can be considered to expand and improve upon our project. Firstly, the study could benefit from using more unsupervised learning machine learning models to further show the variety of results. We can also ensemble more complex machine learning algorithms or models to enhance the metrics. Lastly, we can explore the predictive power of the models over time and collect insights into how they adapt to evolving news cycles and media trends. Since our dataset is isolated to a specific timeframe and news companies can benefit from the momentum of trends, tracking the models' performance over time can be proved interesting and valuable.

References

1. Fernandes, K., Vinagre, P., & Cortez, P. (2015). A proactive intelligent decision support system for predicting the popularity of online news. *Progress in Artificial Intelligence*, 535–546. https://doi.org/10.1007/978-3-319-23485-4_53
2. Tsagkias, M., Weerkamp, W., & de Rijke, M. (2009). Predicting the volume of comments on Online News Stories. *Proceedings of the 18th ACM Conference on Information and Knowledge Management*. https://doi.org/10.1145/1645953.1646225
3. Hong, L., Dan, O., & Davison, B. D. (2011). Predicting popular messages in Twitter. *Proceedings of the 20th International Conference Companion on World Wide Web*. https://doi.org/10.1145/1963192.1963222