**Online News Popularity**

**GROUP 8**

**Akshay Sheokand** **Sakshi Sharma**

**Shivani Narahari** **Shalaka Thakare**

**Abstract**

In our project, news article statistics of a company mashable.com are used to evaluate the popularity and noteworthy information of the target variable using various Machine Learning techniques learned in class. First, we prepared our data and cleaned any impurities. Then we altered our target variable “shares” to be predicted in two different ways. For the first type of prediction, we created 3 buckets of our Y variable to make it a classification problem ( number of shares falls in which bucket) and for the second type we created a binary class where the output based on the number of shares is either 0 or 1. We also visualized our results, compared our accuracies and decided on the best models for our prediction.

**1.** **Introduction:**

As students in tech, we often access [multiple forms](https://towardsdatascience.com/10-best-free-websites-to-learn-more-about-data-science-and-machine-learning-f2c6d7387b8d) of informational social media online. How media is structured and viewed is an interesting attribute that can be measured and used to predict information related to a website to improve the operational efficiency of a company. The data that makes this media can help answer analytical queries based on social media prediction such as: what possible data, style, variable or combination of variables make an article and website popular or unpopular? As a team, we are very interested in social media analytics and what contributes to an article’s number of shares and viewership: whether it may be the sentiment of the article, the word count, genre or day the article is posted. We used the Mashable.com articles data set from the [UCI repository](https://archive.ics.uci.edu/ml/datasets/online+news+popularity) to find popularity related statistics of the company’s article data and performed the following steps:

I. Data Processing

II. Data Cleaning

III. Visualizing our data

IV. Feature Engineering

V. Defining our Y variables – Binary and Multiclass

VI. Running our machine learning models: Logistic Regression,

DT Classifier, XGBoost, Naive Bayes, KNN, SVM and

Random Forest.

VII. Compare Accuracies, find best model, visualize accuracies

**2. Related work:**

Writers K. Fernandes et al. paper “[A Proactive Intelligent ….Online News](https://www.researchgate.net/publication/283510525_A_Proactive_Intelligent_Decision_Support_System_for_Predicting_the_Popularity_of_Online_News)” is our main reference article used for this project. The writers speak about the importance of predicting online news popularity and how it navigates in the expansive internet spaces of the world wide web. Along with our own work using multiclass Y and other Machine Learning models, this article was very informative on how to split our dataset into testing and training data, use binary classification and models like SVM, KNN and Naïve Bayes and methods to find the best hyperparameter.

**3. Machine Learning Models used:**

I. Logistic Regression

II. DT Classifier

III. XGBoost

IV. Naïve Bayes

V. KNN

VI. SVM

VII. Random Forest

**Model Design:**

1. **Logistic Regression :**

Logistic regression is used to model the probability of a discrete outcome when an input variable is given. Here, the dependent variable is binary. It is used to derive the relationship between the dependent variable and one or more independent variables.

The equation for logistic regression is given by:





**Pseudo Code:**

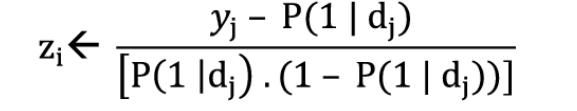
Input - Training data

Steps - 1) Read the training dataset

2) For i from 1 to k

3) For each training instance di:

4) Set the target value for regression to:



5) Initialize the weight of instance dj to P(1| dj) . (1-P(1|dj))

6) Finalize a f(j) to the data with class value (zj) and weights (wj)

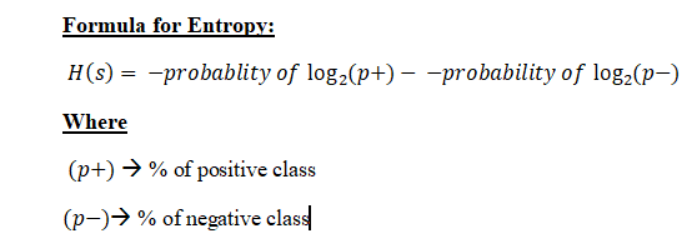
7) Assign (class label : 1) if P(1|dj) > 0.5, otherwise (class label : 2)

Output - A class of testing dataset

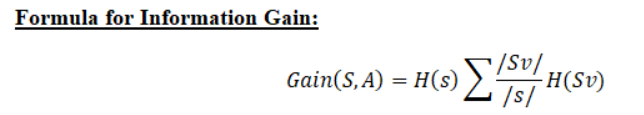
**II. DT Classifier:**

Decision Tree algorithms belong to the family of supervised learning algorithms. It is used for both classification as well as regression problems. It uses a tree representation to predict the class where the leaf nodes correspond to a class label and the internal nodes represent the attributes.

Entropy is used to measure the impurity, disorder, or uncertainty of a group of examples. It controls how a decision tree decides to split the data. It ranges from 0 to 1 - Lesser the value, more the confidence of predicting it as that class.



Information Gain is used to decide which feature to split on at each tree building step. The split which has the highest IG will be taken as the initial split and this process will continue until the information gain is 0.



**Pseudocode:**

1. Original dataset S is taken as the root node
2. On each iteration of the algorithm, it iterates through the very unused attribute of the set S and calculates Entropy(H) and Information gain(IG) of this attribute.
3. It then selects the attribute which has the smallest Entropy or Largest Information gain.
4. The set S is then split by the selected attribute to produce a subset of the data.
5. The algorithm continues to recur on each subset, considering only attributes never selected before.

**III. XGBoost:**

XGBoost is an ensemble learning algorithm. It is better than a single algorithm as it combines the predictive power of multiple learners. The resultant is a single model which gives the combined output from many models.

**Pseudocode:**

* An initial model F0 is defined to predict the target variable y. This model will be associated with a residual (y – F0)
* A new model h1 is fit to the residuals from the previous step
* Now, F0 and h1 are combined to give F1, the boosted version of F0. The mean squared error from F1 will be lower than that from F0:



* To improve the performance of F1, we could model after the residuals of F1 and create a new model F2:



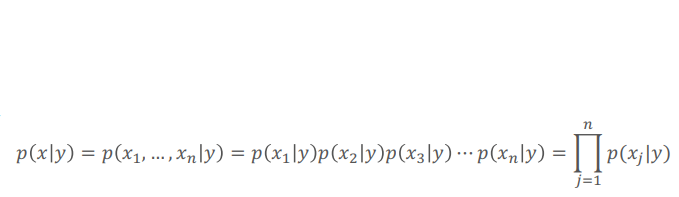
* This can be done for ‘m’ iterations, until residuals have been minimized as much as possible:



IV.  **Naïve Bayes:**

Naive Bayes is a classification strategy in view of Bayes' theorem with the "naive" presumption of conditional independence between each set of elements given the value of the class variable. Basically, the Naive Bayes classifier assumes that the presence of specific features of a class isn't connected with the presence of different features.

Given a class variable y and dependent feature vector x1 through xn, the Bayes' hypothesis expresses the beneath referenced relationship.



**Pseudocode:**

Input : Training dataset T

F = (f1, f2, f3,...,fn) // value of the predictor variable in testing dataset.

Output : A class of testing dataset

Steps:

1. Read the training dataset T
2. Calculate the mean and standard deviation of the predictor variables in each class.
3. Repeat

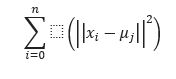
Calculate the probability of fi using the gauss density equation in each class until the probability of each predictor variable (f1, f2, f3, …, fn) has been calculated.

1. Calculate the likelihood of each class.
2. Get the greatest likelihood.

**V.**   **KNN:**

K-Nearest Neighbors (KNN) calculation is a straightforward, supervised AI algorithm. It works for taking care of both classification and regression issues. KNN tracks down the distance between a query and all examples in the information, and it picks the predetermined number models (K) that are nearest to the inquiry, and afterward either votes in favor of the mark with the most noteworthy recurrence (for classification) or the mean of the labels (for regression).

The k-means algorithm separates a bunch of N samples X into K disjoint groups C, each portrayed by the mean μj of the examples in the group. The means are for the most part alluded to as the cluster "centroids". The chosen centroids limit the inside cluster sum of-squares or dormancy model by:



**Pseudocode:**

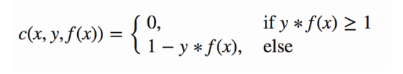
1. Calculate “d(x, xi)” i =1, 2, ….., n; where d denotes the [Euclidean distance](https://dataaspirant.com/2015/04/11/five-most-popular-similarity-measures-implementation-in-python/) between the points.
2. Arrange the calculated n Euclidean distances in non-decreasing order.
3. Let k be a +ve integer, take the first k distances from this sorted list.
4. Find those k-points corresponding to these k-distances.
5. Let ki denotes the number of points belonging to the ith class among k points i.e. k ≥ 0
6. If ki >kj ∀ i ≠ j then put x in class i.

**VI. SVM :**

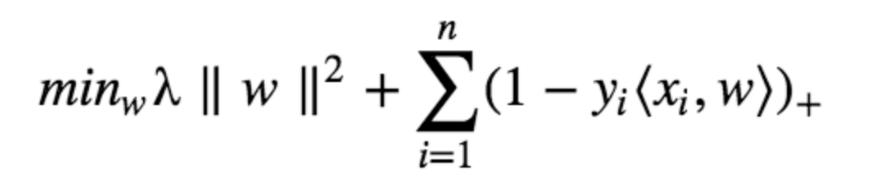
Support Vector Machines are used in classification problems when the classes are linearly or not linearly separable. The feature space is enlarged using kernels in order to accommodate a non-linear boundary between classes. When the degree of a kernel is greater than one, the decision boundary becomes more flexible.

The objective of SVM is to find a hyperplane with the maximum margin in an N-dimensional feature space that clearly classifies the data points. The greater the margin, more the confidence in classifying new data points in the future. Support vectors are those data points that are closest to the hyperplane and influence the position of the hyperplane. When the support vectors are removed, the margin also changes.

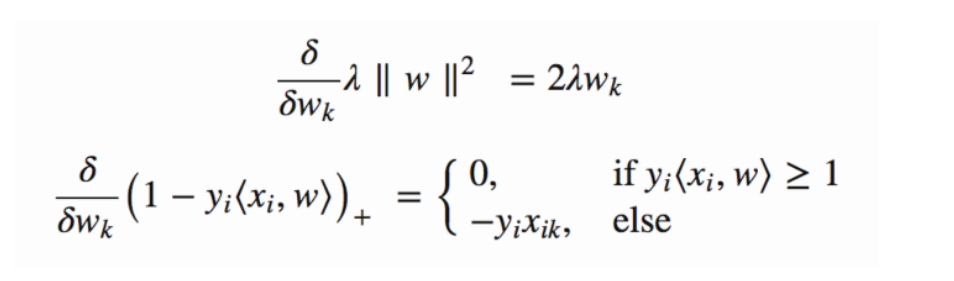
In order to maximize the margin, the loss function for SVM (hinge loss) has to be minimized. Hinge loss function is given by :



If the predicted class and the actual class are of the same sign, then the loss function is 0. If they aren’t the same, the loss value has to be calculated. A regularization parameter is added to balance out the margin maximization and the loss. The cost function with the regularization parameter is :

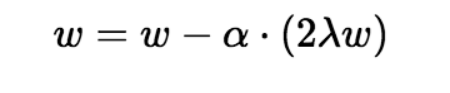


In order to find the gradients, we have to find the partial derivatives with respect to the weights. These gradients can be used to update the weights.



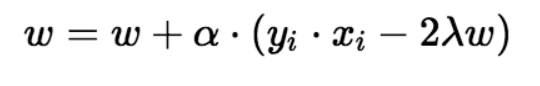
**No misclassification:**

Only the gradient from the regularization parameter has to be updated:



**When there is misclassification:**

In order to find the new gradients, we use the loss with the regularization parameter:



**VII. Random Forest -**

Random Forest is an ensemble of decision trees. An individual tree in the random forest model gives out one class at the output, and the class with the majority votes becomes the model’s prediction.

**Pseudocode:**

1. Randomly select **“k”** features from total **“m”** features. Where k<< m
2. Among the **“k”** features, calculate the node **“d”** using the best split point.
3. Split the node into **daughter nodes** using the **best split**.
4. Repeat **1 to 3** steps until the “l” number of nodes has been reached.
5. Build forest by repeating steps **1 to 4** for “n” number times to create **“n” number of trees**.

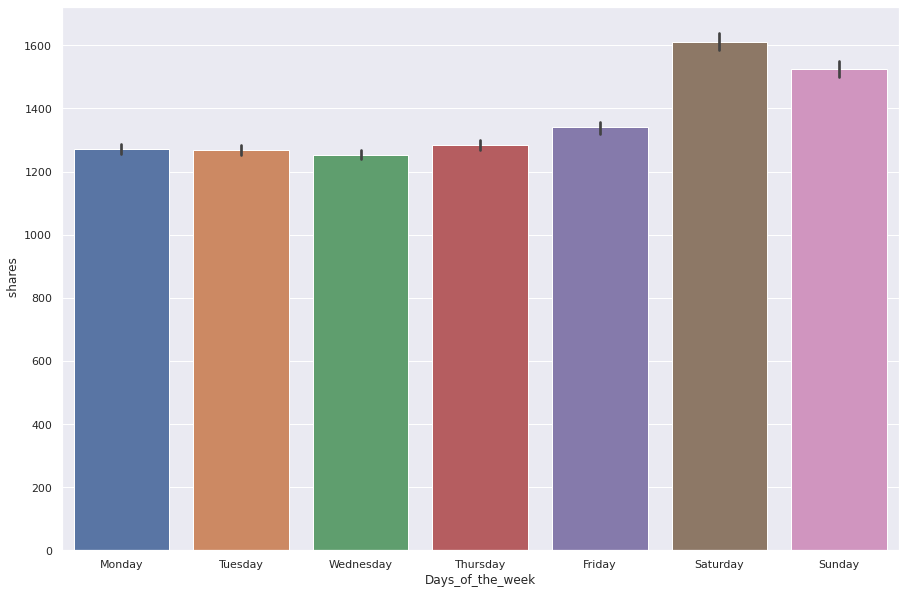
Then,

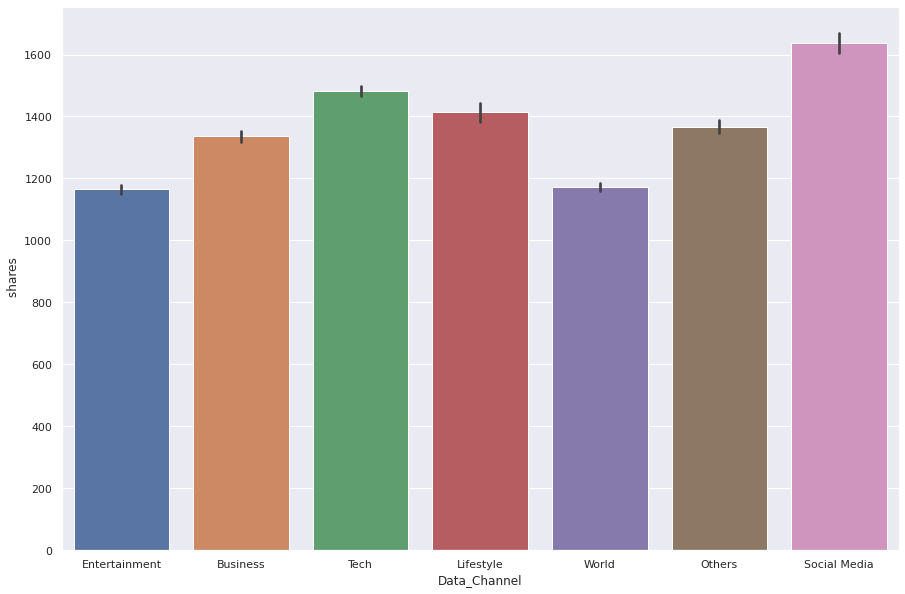
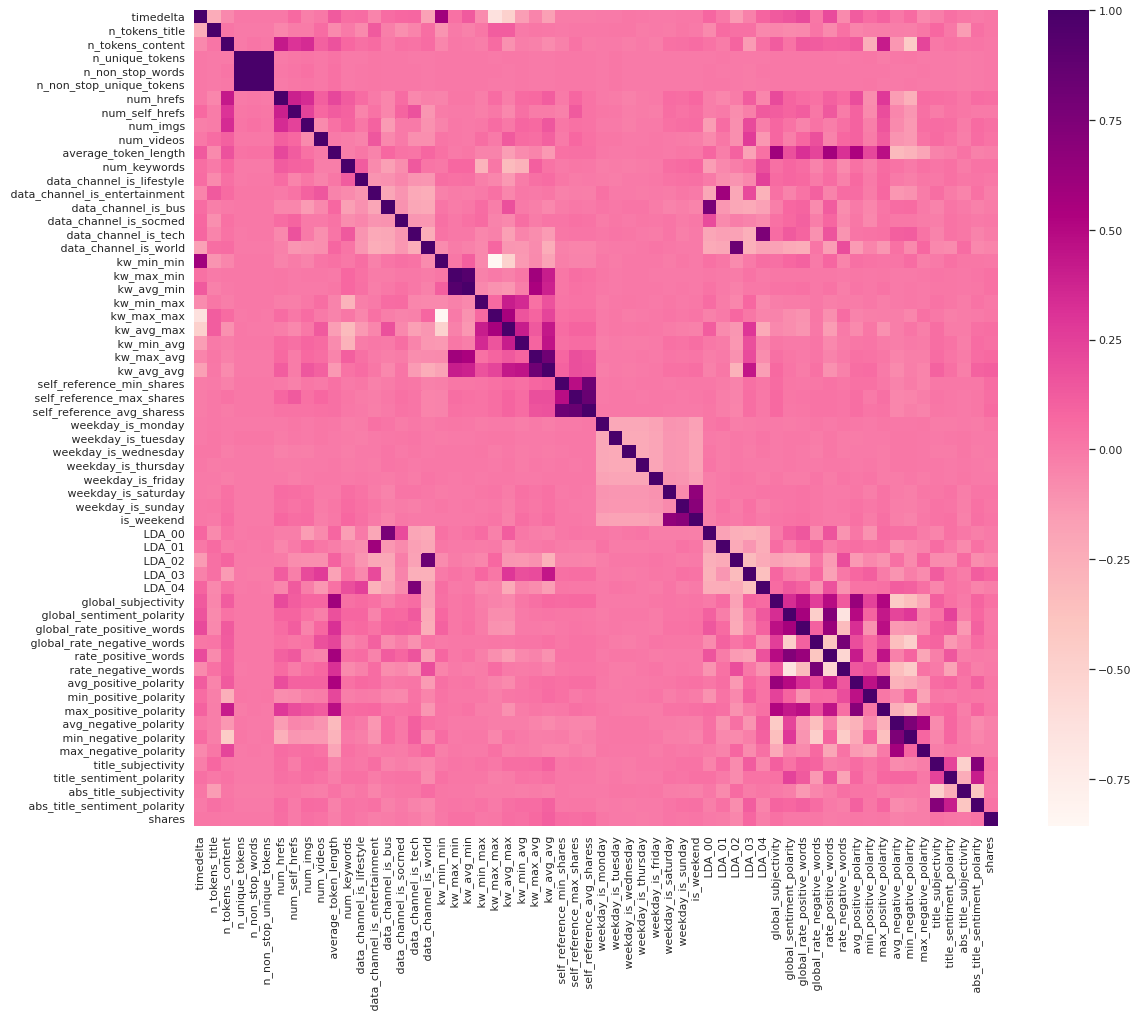
1. Takes the **test features** and use the rules of each randomly created decision tree to predict the outcome and stores the predicted outcome (target)
2. Calculate the **votes** for each predicted target.
3. Consider the **highly voted** predicted target as the **final prediction** from the random forest algorithm.

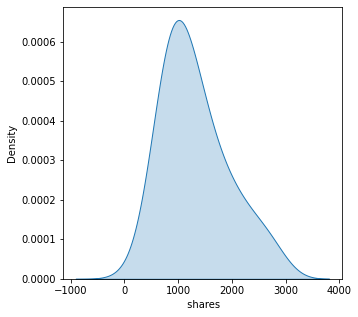
**4. Experimental Results:**

About the dataset:

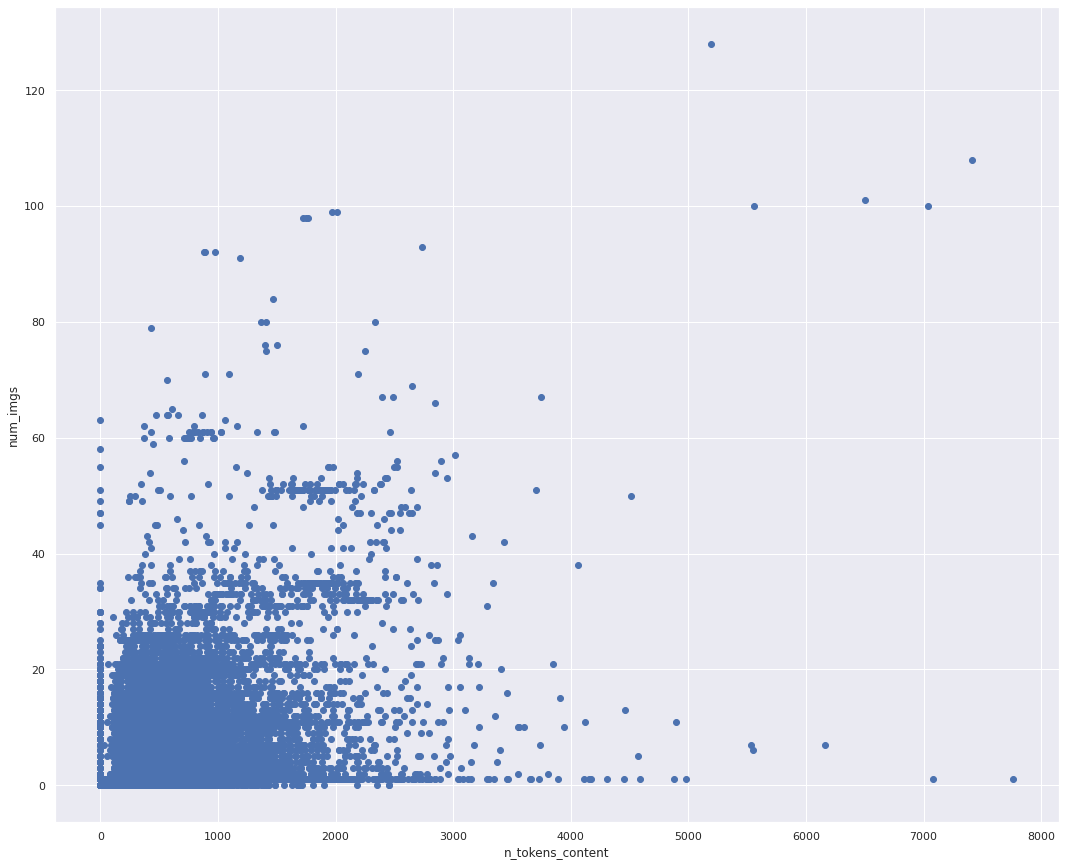
* Dataset: <https://archive.ics.uci.edu/ml/datasets/Online+News+Popularity>
* 39644 instances, 61 attributes
* Target variable : “shares”
* Median Shares: 1200
* Cleaned data:   
  We dropped unnecessary columns such as “URL” and “timedelta”. We further removed 813 rows with missing values or 0 values in “n\_tokens\_content (number of words in the article). We evaluated outliers and dropped 9274 outliers.
* Popular publishing day : Saturday



* Popular genre of articles : Social Media
* Heatmap to visualize correlation between variables: strong correlation.   
    
  
* Distribution of article shares: Most articles were shared between 1000- 2000 times.



* Images vs. Words shared



**Problems to solve:**

* This project is a classification problem where two different types of Y variables will be predicted.   
  Type 1: Binary Y variable for “shares” where,   
   - 0 : shares less than 1200   
   - 1 : shares more than 1200

Type 2: Multiclass Y variable for “shares” where,

- bucket 1 : shares less than 1000,   
 - bucket 2 : shares between 1000 and 2000,  
 - bucket 3 : shares greater than 2000

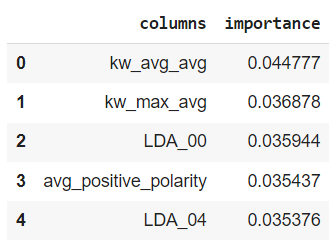
**Experimental Process:**Our initial results before presentation day were quite different from our outputs after finding the accurate hyperparameters. Our accuracies were low regardless of various adjustments. We ran the hyper parameter tuning grid for our best performing model- XGBoost to improve this.

**Outcomes and visualizations based on metric:**

Following are the accuracy and results for each model.

1. **Multi-Class Models:**
2. **DT Classifier:**

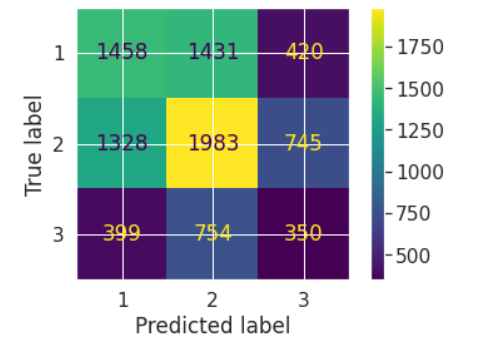
Feature Importance:



Classification Report:

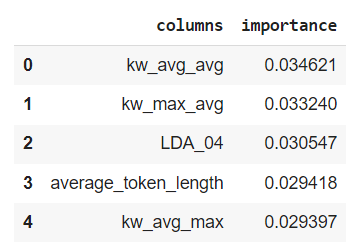


Confusion Matrix:

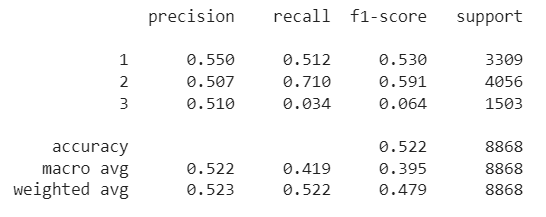


**2. Random Forest:**

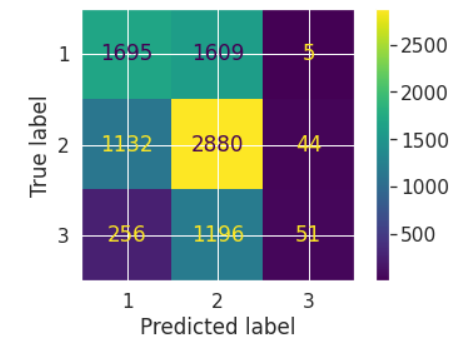
Feature Importance:



Classification Report:

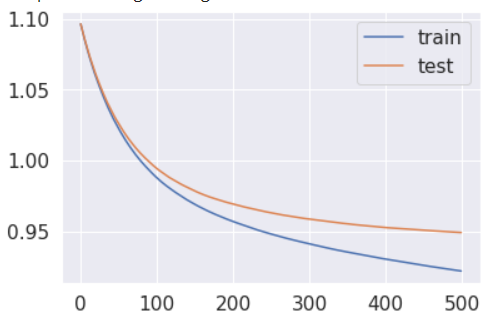


Confusion Matrix:

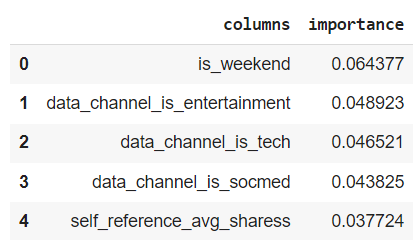


**3. X Gradient Boost:**

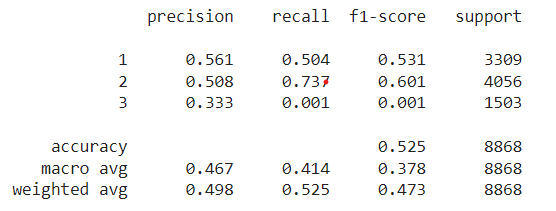
Train. V. Test:



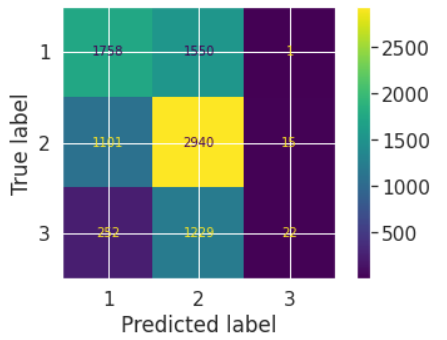
Feature Importance:



Classification Report:

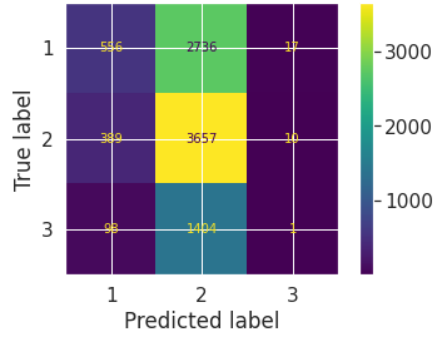


Confusion Matrix:

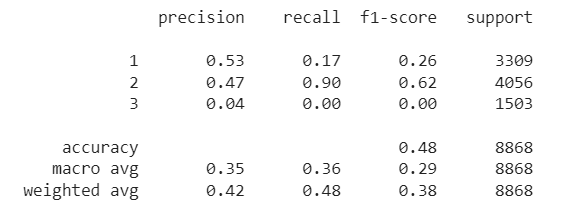


**4. Logistic Regression:**

Confusion Matrix:

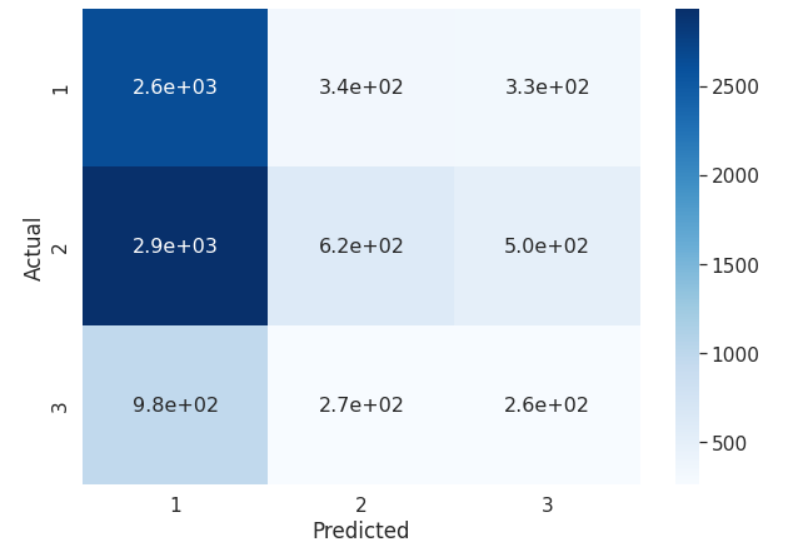


Classification Report:

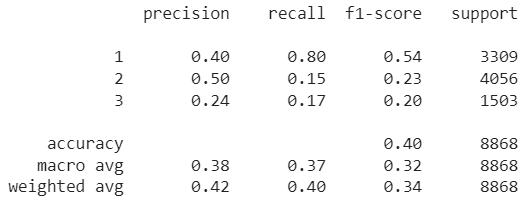


**5. Naive Bayes**

Confusion Matrix:

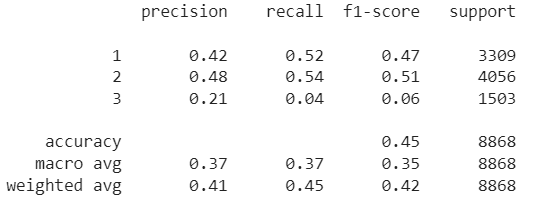


Classification Report:

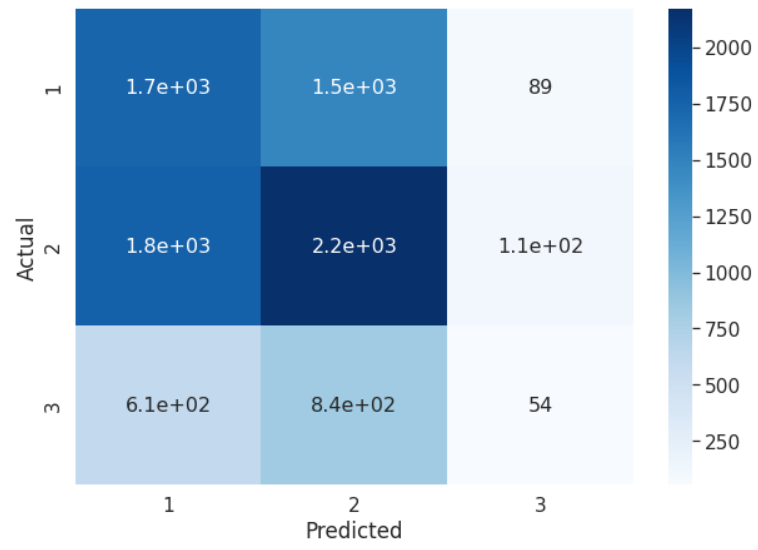


**6. KNN:**

Classification Report:

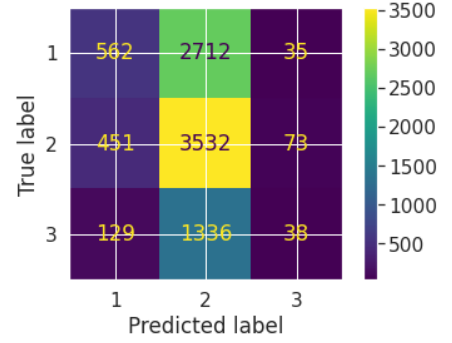


Confusion Matrix:

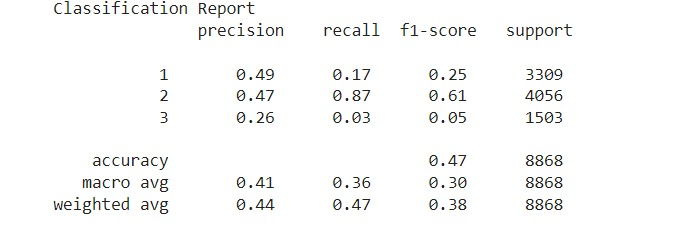


**7: SVM:**

Confusion Matrix:



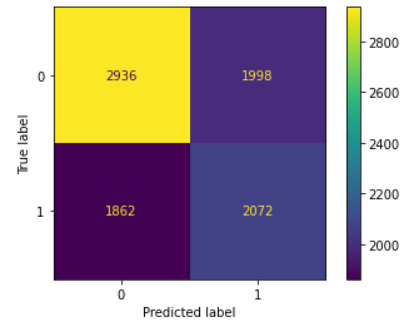
Classification Report:



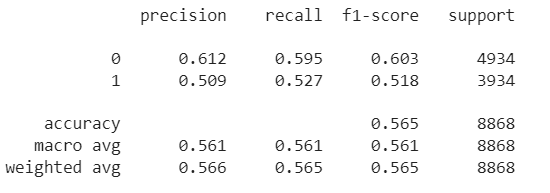
**B. Binary Classification Models:**

1. **DT CLassifier:**

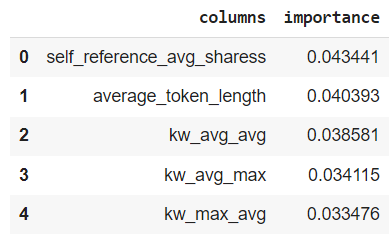
Confusion Matrix :



Classification Report:

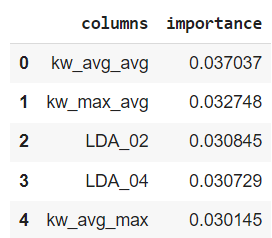


Feature importance:

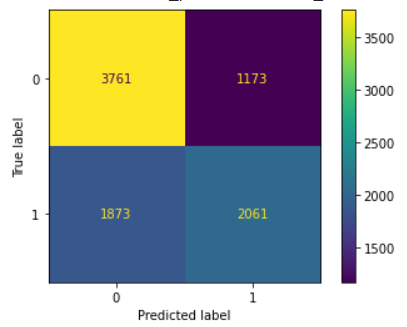


1. **Random Forest:**

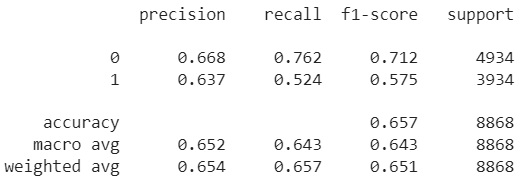
Feature importance:



Confusion Matrix:

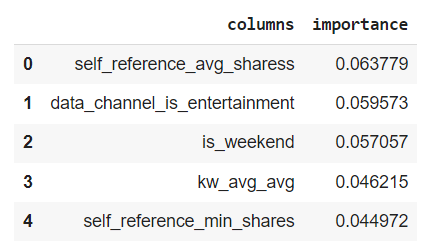


Classification Report:

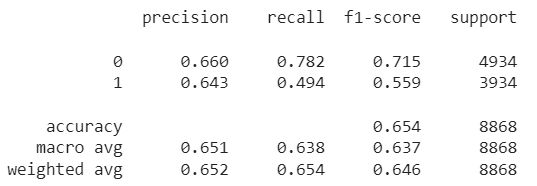


1. **XGBoost:**

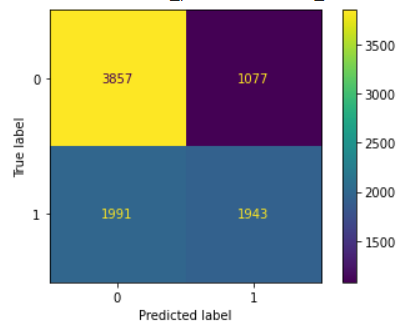
Feature Importance:



Classification Report:

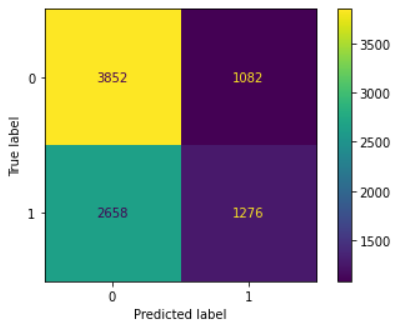


Confusion Matrix:

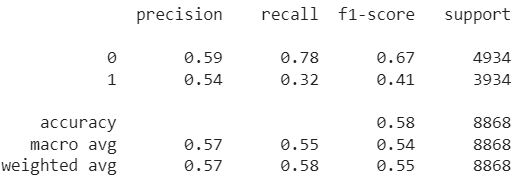


1. **Logistic Regression:**

Confusion Matrix:

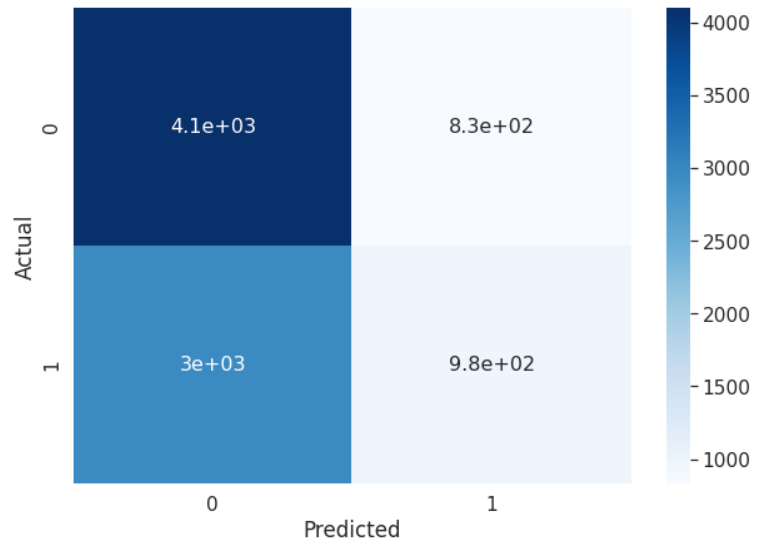


Classification Report:

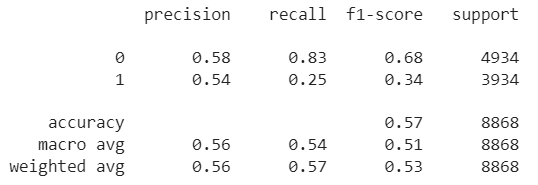


1. **Naive Bayes:**

Confusion Matrix:

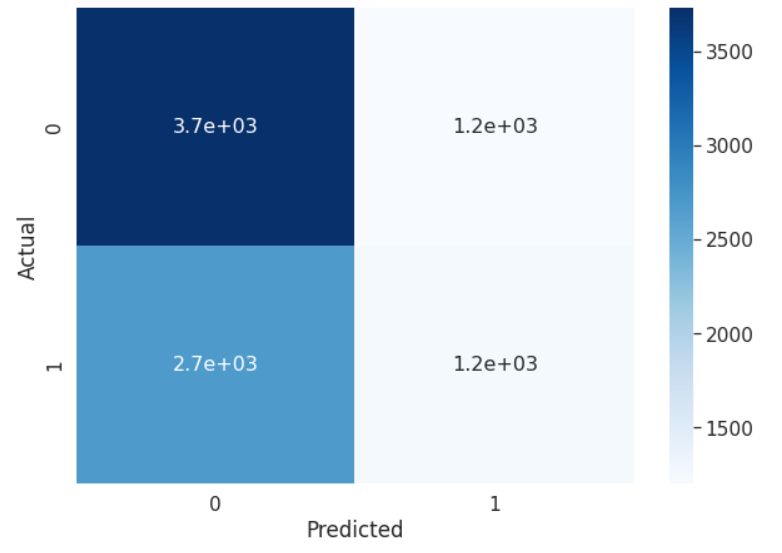


Classification report:

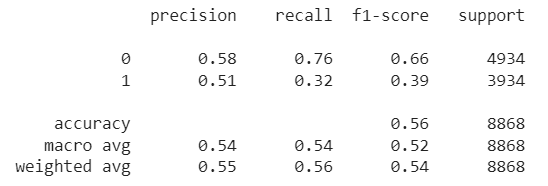


1. **KNN:**

Confusion Matrix:

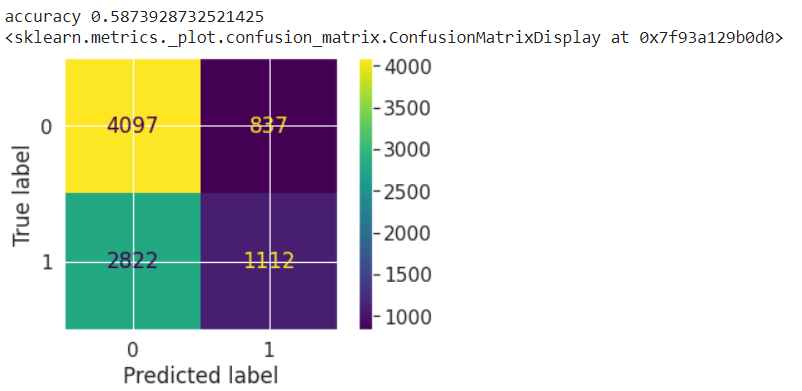


Classification Report:

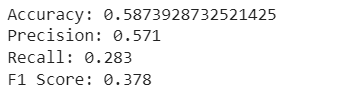


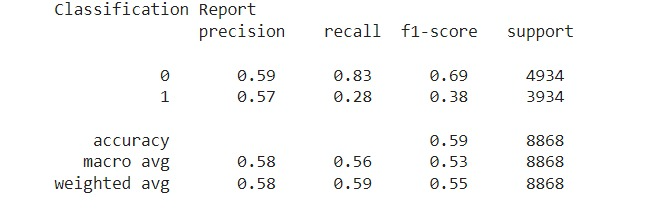
1. **SVM:**

Confusion Matrix:



Classification report:





**Conclusion:**

When it comes to accuracy, XBoost is seemingly our best model for both Multiclass and Binary classifications. What we have learnt is that even with adjustments made like splitting the testing and training sets differently, altering our parameters and threshold, trying out multiclass and binary class variables that the accuracies for our data are quite low in relevancy. The model comparisons for the accuracy, recall, precision and F1 score are also similar. It could be possible to improve prediction accuracy if there was further data and documentation. For example, if there was data of the monthly distribution of the articles, we could evaluate how different seasons, historical events and festivals affect what type of genre of an article to be the most popular. There could be a possibility that the lack of data or removal data impurities/outliers could possibly have essential information required for prediction. It is also noteworthy that the concept of “popularity is relative”, hence its prediction cannot be fully accurate.

**References:**

1. Chakure, Afroz. “Random Forest Classification and Its Implementation.” *Medium*, The Startup, 6 Nov. 2020, https://medium.com/swlh/random-forest-classification-and-its-implementation-d5d840dbead0.
2. Gandhi, Rohith. “Support Vector Machine - Introduction to Machine Learning Algorithms.” *Medium*, Towards Data Science, 5 July 2018, https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47.
3. “How the Random Forest Algorithm Works in Machine Learning.” *Dataaspirant*, 26 Aug. 2020, https://dataaspirant.com/random-forest-algorithm-machine-learing/.
4. “How the Random Forest Algorithm Works in Machine Learning.” *Dataaspirant*, 26 Aug. 2020, https://dataaspirant.com/random-forest-algorithm-machine-learing/.
5. “Introduction to Random Forest in Machine Learning.” *Section*, <https://www.section.io/engineering-education/introduction-to-random-forest-in-machine-learning/>.
6. “Knn Classifier, Introduction to K-Nearest Neighbor Algorithm.” *Dataaspirant*, 27 June 2017, https://dataaspirant.com/k-nearest-neighbor-classifier-intro/#:~:text=Knn%20Algorithm%20Pseudocode%3A,distances%20from%20this%20sorted%20list.
7. *Pseudocode of Naïve Bayes Algorithm | Download Scientific Diagram*. https://www.researchgate.net/figure/Pseudocode-of-naive-bayes-algorithm\_fig2\_325937073.
8. Sekhar, Sidharth. “Math behind Logistic Regression Algorithm.” *Medium*, Analytics Vidhya, 30 Aug. 2019, <https://medium.com/analytics-vidhya/logistic-regression-b35d2801a29c>.
9. Shin, Terence. “A Mathematical Explanation of Naive Bayes in 5 Minutes.” *Medium*, Towards Data Science, 6 June 2020, <https://towardsdatascience.com/a-mathematical-explanation-of-naive-bayes-in-5-minutes-44adebcdb5f8>.
10. “SVM Hyperparameter Tuning Using GRIDSEARCHCV: ML.” *GeeksforGeeks*, 4 Oct. 2021, <https://www.geeksforgeeks.org/svm-hyperparameter-tuning-using-gridsearchcv-ml/>.
11. Yiu, Tony. “Understanding Random Forest.” *Medium*, Towards Data Science, 29 Sept. 2021, <https://towardsdatascience.com/understanding-random-forest-58381e0602d2>.
12. [https://www.researchgate.net/figure/Pseudocode-of-naive-bayes-algorithm\_fig2\_325937073#:~:text=The%20pseudocode%20of%20the%20Na%C3%AFve,and%20perform%20regression.%20…](https://www.researchgate.net/figure/Pseudocode-of-naive-bayes-algorithm_fig2_325937073#:~:text=The%20pseudocode%20of%20the%20Na%C3%AFve,and%20perform%20regression.%20%E2%80%A6)