Machine Learning Project
SC-20

SC	Section 2	2021170161	حبيبة علاء الدين عادل الصناديدي
SC	Section 7	2021170538	ملك رافت فؤاد فاضل
SC	Section 6	2021170691	مريم محمد يسري محمد
SC	Section 2	2021170195	رنا وحيد تمام عثمان

Preprocessing:

First, we import the necessary libraries. Then, we read the CSV file using **read_csv** and found that our dataset contains 38,643 rows and 46 columns.

Categorical Data (Objects):

- **url**: The URL of the article.
- **title**: The title of the article.
- **channel_type**: The type of channel where the article was published.
- **weekday**: The day of the week when the article was published.
- **isWeekEnd**: Binary indicator of whether the article was published on a weekend.

Numerical Data (Integers):

- **timedelta**: The time difference between the publication of the article and the dataset acquisition.
- **n_tokens_content**: Number of words in the content of the article.
- **num_hrefs**: Number of links in the content of the article.
- **num_self_hrefs**: Number of links to other articles published by the same publisher.
- **num_imgs**: Number of images in the content of the article.
- **num_videos**: Number of videos in the content of the article.
- **num_keywords**: Number of keywords in the metadata of the article.
- **kw_min_min**: Minimum number of shares of the articles for the 5 most shared keywords.
- **kw_min_max**: Maximum number of shares of the articles for the 5 most shared keywords.
- **kw_max_max**: Maximum number of shares of the articles for the 5 most shared keywords.
- **shares**: Number of shares of the article (target variable).

Numerical Data (Floats):

- **n_unique_tokens**: Rate of unique words in the content of the article.
- **n_non_stop_words**: Rate of non-stop words in the content of the article.
- **n_non_stop_unique_tokens**: Rate of unique non-stop words in the content of the article.
- **average_token_length**: Average length of the words in the content of the article.
- **kw_max_min**: Minimum number of shares of the articles for the 5 least shared keywords.
- **kw_avg_min**: Average number of shares of the articles for the 5 least shared keywords.
- **kw_avg_max**: Average number of shares of the articles for the 5 most shared keywords.
- **kw_min_avg**: Minimum average shares of the articles for the 5 most shared keywords.
- **kw_max_avg**: Maximum average shares of the articles for the 5 most shared keywords.
- **kw_avg_avg**: Average average shares of the articles for the 5 most shared keywords.
- **self_reference_min_shares**: Minimum number of shares of articles referenced in the content of the article.
- **self_reference_max_shares**: Maximum number of shares of articles referenced in the content of the article.
- **self_reference_avg_sharess**: Average number of shares of articles referenced in the content of the article.
- **LDA_00** to **LDA_04**: Latent Dirichlet Allocation (LDA) topic features.
- **global_subjectivity**: Text subjectivity score.
- **global_sentiment_polarity**: Text sentiment polarity score.
- **global_rate_positive_words**: Rate of positive words in the content of the article.
- **global_rate_negative_words**: Rate of negative words in the content of the article.
- **rate_positive_words**: Rate of positive words among non-neutral tokens.
- **rate_negative_words**: Rate of negative words among non-neutral tokens.
- **avg_positive_polarity**: Average polarity of the positive words in the content of the article.
- **min_positive_polarity**: Minimum polarity of the positive words in the content of the article.
- **max_positive_polarity**: Maximum polarity of the positive words in the content of the article.
- **avg_negative_polarity**: Average polarity of the negative words in the content of the article.
- **min_negative_polarity**: Minimum polarity of the negative words in the content of the article.
- **max_negative_polarity**: Maximum polarity of the negative words in the content of the article.

- Our target is to predict the number of **shares** in regression.
- Our target is to classify **Article_Popularity** in classfication.
- We found that the "channel type" column was not named correctly; it should be **channel_type** and “Article Popularity it should be “Article_Popularity”.
- Additionally, some column names contained whitespaces, which we removed.
- We observed that our dataset has no null values or duplicates.
- In the **channel_type** column, we found 7 unique values, so We determined that **[]** is not a valid value. Approximately 5,960 instances of **[]** were found, representing 15.42% of the entire dataset. To address this, we replaced these instances with the mode of the **channel_type** column, which is **data_channel_is_world**.
- **Encoding**: the categorical data. We applied label encoding to the **url**, **title**, **channel_type**, and **weekday** columns, resulting in new columns: **url_encoded**, **title_encoded**, **channel_type_encoded**, and **weekday_encoded**. For the **isWeekEnd** column, we used one-hot encoding.
- **In classification**: we encoded the target column.

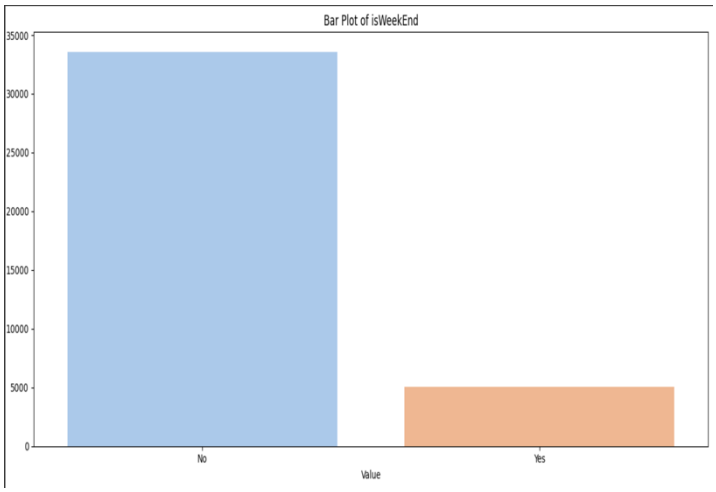
Analysis (EDA):

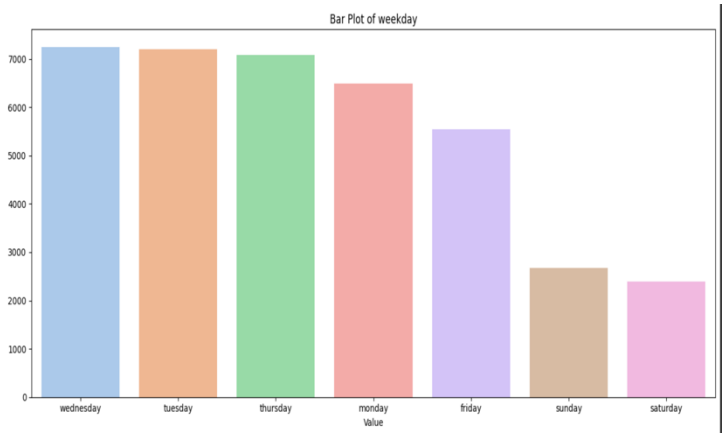
Graph [1,2,3]:

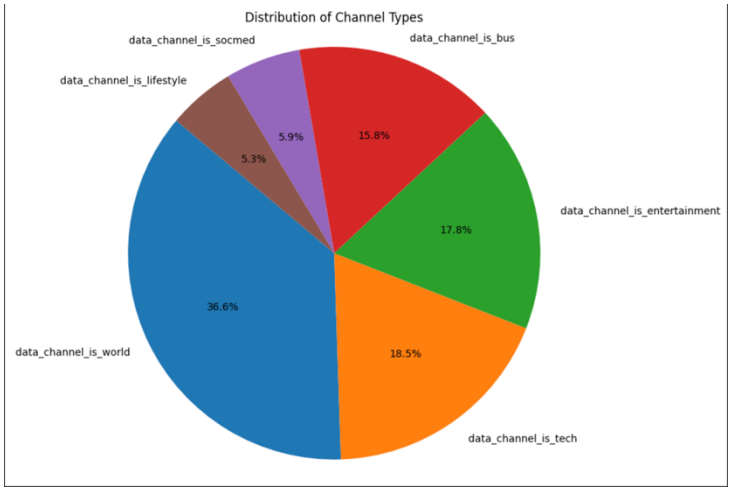
Objective: Create a bar plot and a pie chart to visualize the distribution of **isWeekEnd**, **weekday**, and **channel_type**.

Findings:

- For **isWeekEnd**, the number of 'no' values is greater than 'yes'.
- Among weekdays, 'Wednesday' is the most frequently occurring day.
- For **channel_type**, 'data_channel_is_world' is the most frequently occurring channel.



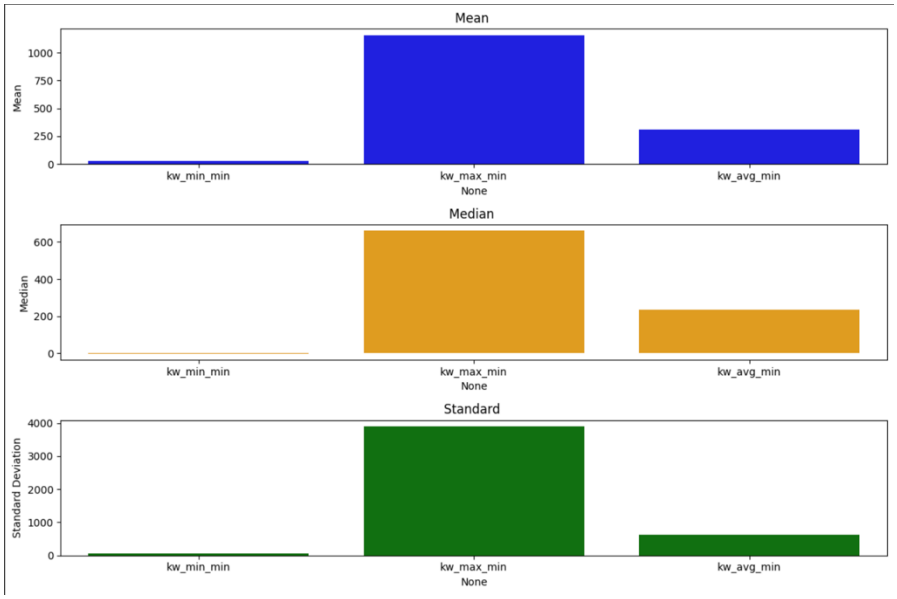




Graph [4]:

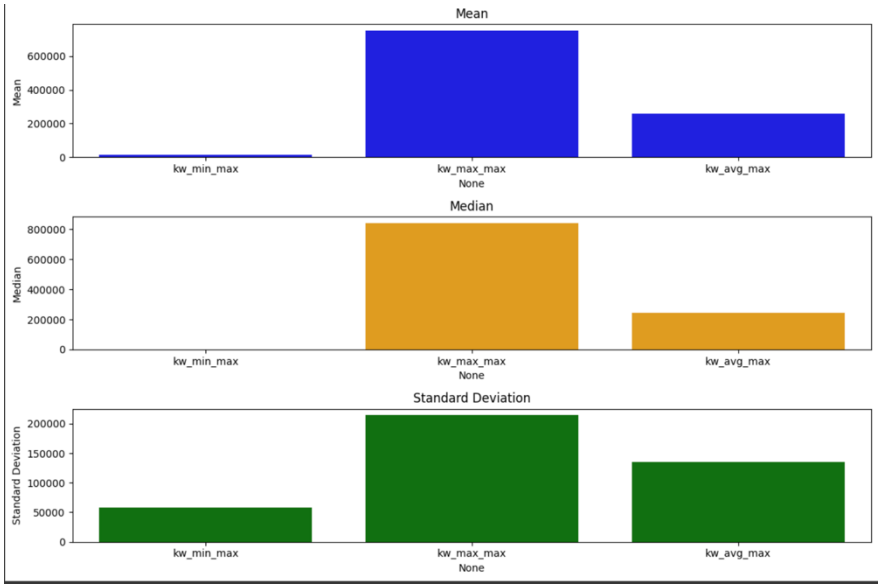
Objective: Calculate the mean, standard deviation, and median for the columns **kw_min_min**, **kw_max_min**, and **kw_avg_min**.

Findings: **kw_max_min** has the highest mean, median, and standard deviation among the mentioned columns.



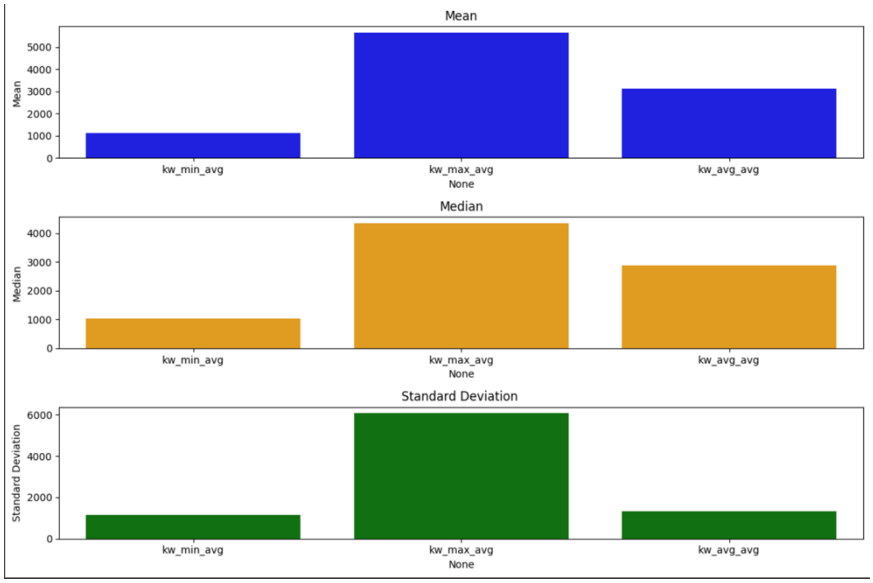
Graph [5]:
Objective: Calculate the mean, standard deviation, and median for the columns **kw_min_max**, **kw_max_max**, and **kw_avg_max**.
Findings:

- **kw_max_max** has the highest mean, median, and standard deviation among the mentioned columns.



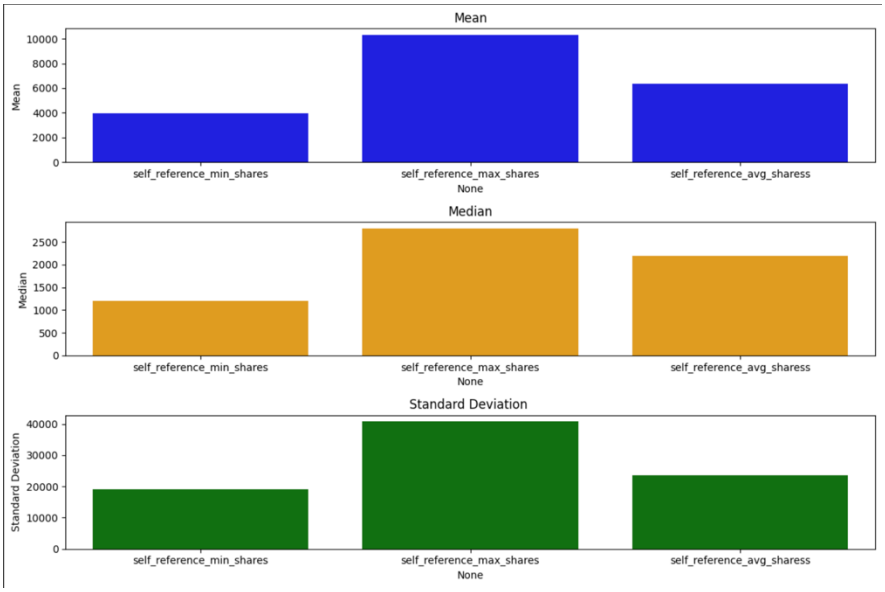
Graph [6]:
Objective: Calculate the mean, standard deviation, and median for the columns **kw_min_avg**, **kw_max_avg**, and **kw_avg_avg**.
Findings:

- **kw_max_avg** has the highest mean, median, and standard deviation among the mentioned columns.



Graph [7]:
Objective: Calculate the mean, standard deviation, and median for the column's **self_reference_min_shares**, **self_reference_max_shares**, and **self_reference_avg_shares**.
Findings:

- **self_reference_max_shares** has the highest mean, median, and standard deviation among the mentioned columns.

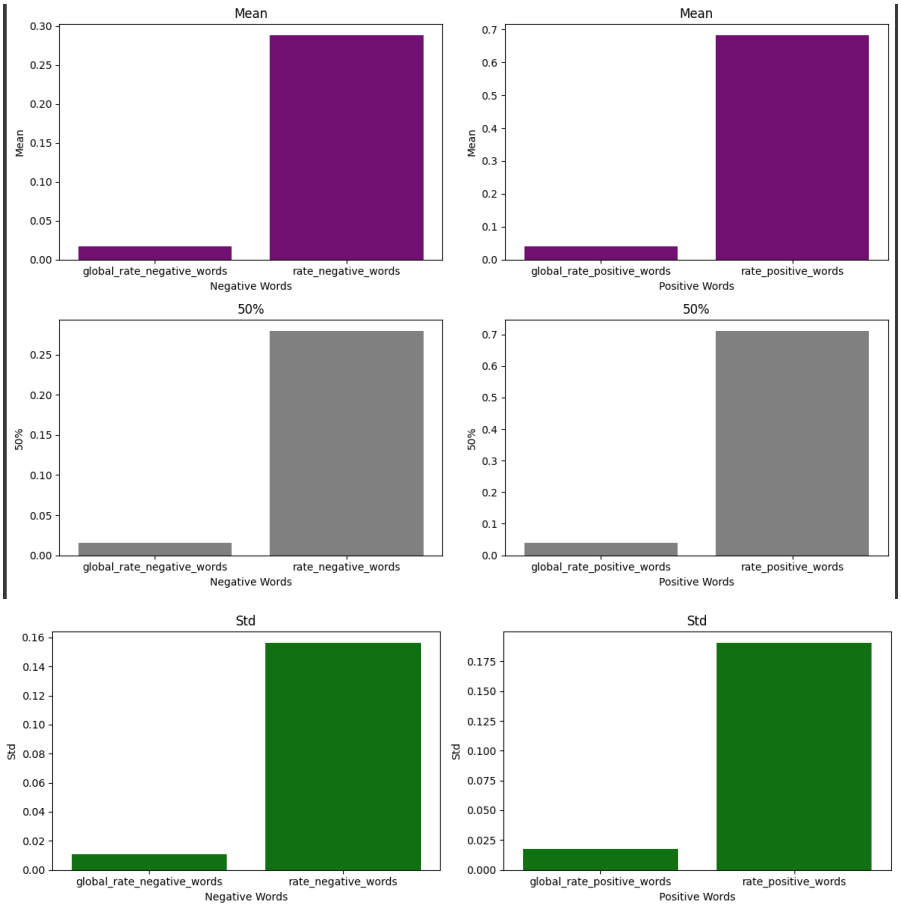


Graph [8]:

Objective: Calculate the mean, standard deviation, and median for the columns related to Negative Words (**global_rate_negative_words**, **rate_negative_words**) and Positive Words (**global_rate_positive_words**, **rate_positive_words**).

Findings:

- Among Negative Words, **rate_negative_words** have the highest mean, median, and standard deviation.
- Among Positive Words, **rate_positive_words** have the highest mean, median, and standard deviation.

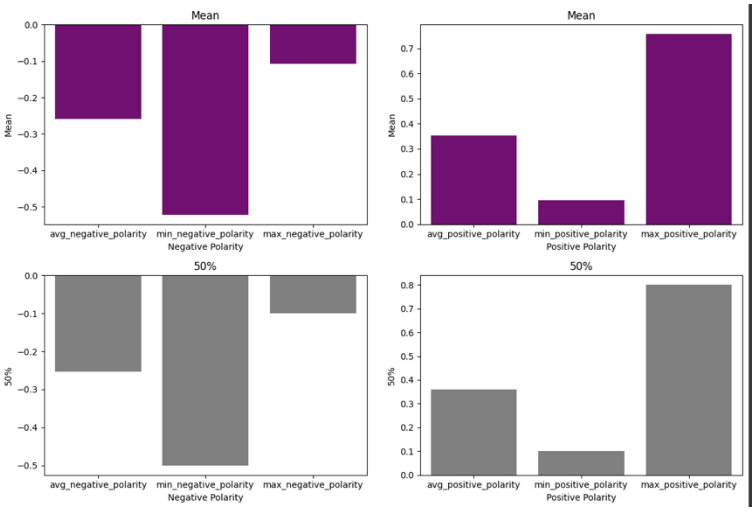


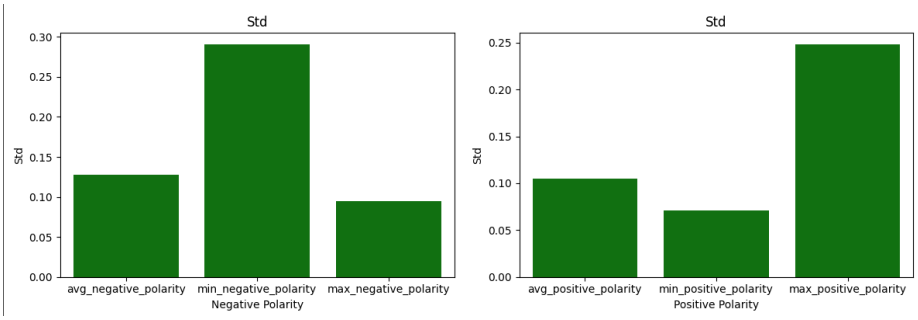
Graph [9]:

Objective: Calculate the mean, standard deviation, and median for the columns related to Negative Polarity (**avg_negative_polarity**, **min_negative_polarity**, **max_negative_polarity**) and Positive Polarity (**avg_positive_polarity**, **min_positive_polarity**, **max_positive_polarity**).

Findings:

- For Negative Polarity, **min_negative_polarity** has the highest mean, median, and standard deviation.
- For Positive Polarity, **max_positive_polarity** has the highest mean, median, and standard deviation.





Regression

Feature Selection:

Data Splitting:

We divided the dataset into two parts:

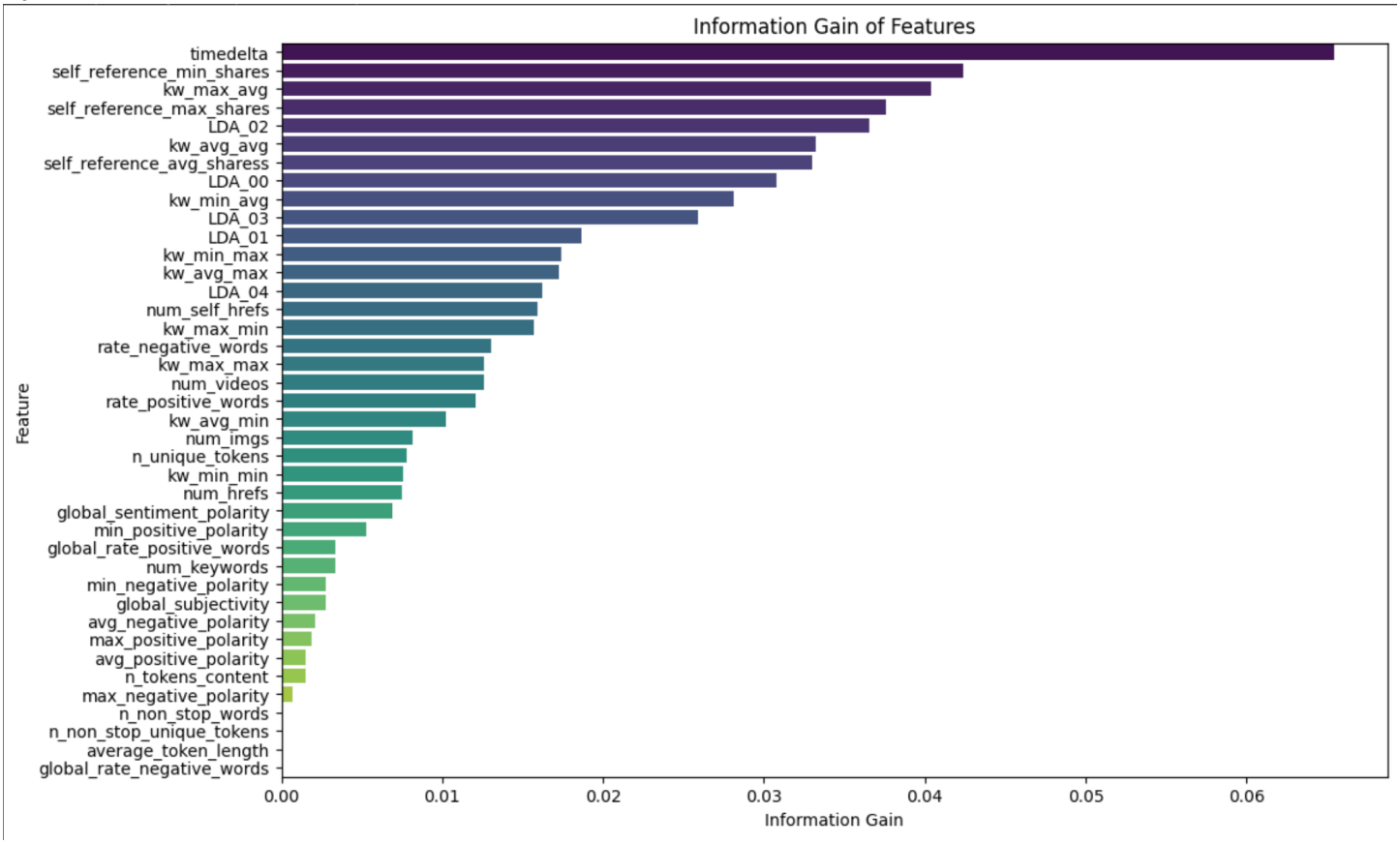
- **X**: This contains the independent variables. We decided to exclude all categorical data for feature selection, thus removing the following columns: **shares**, **channel_type**, **channel_type_encoded**, **url**, **url_encoded**, **title_encoded**, **title**, **weekday_encoded**, **weekday**, **isWeekEnd_0**, **isWeekEnd_1**, **isWeekEnd**.
- **Y**: This consists of the dependent variable or target column. We removed the **shares** column from **X** and placed it in **Y**.

Feature Selection Techniques Used:

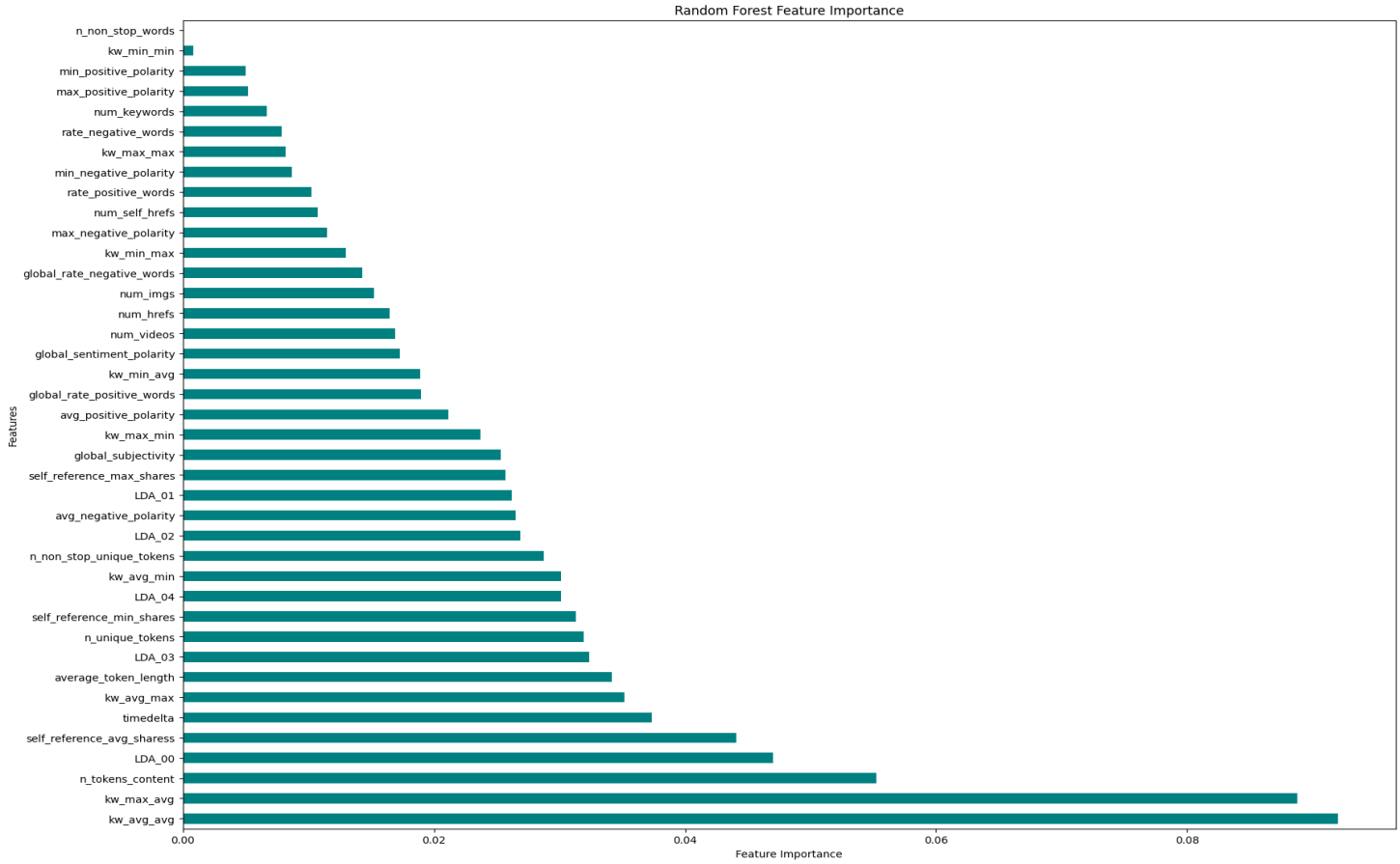
We employed five different techniques for feature selection:

1. **Information Gain**: Used to determine the importance of numerical features based on the entropy of the target variable.
2. **Random Forest Regressor**: Utilized the feature importance provided by a Random Forest model to rank the numerical features.
3. **Gradient Boosting**: Like the Random Forest Regressor, Gradient Boosting was used to rank numerical features based on their importance.
4. **Correlation Matrix**: Assessed the linear relationship between numerical features and the target variable.
5. **ANOVA (Analysis of Variance)**: Applied to the categorical features to determine their significance in explaining the variance in the target variable.

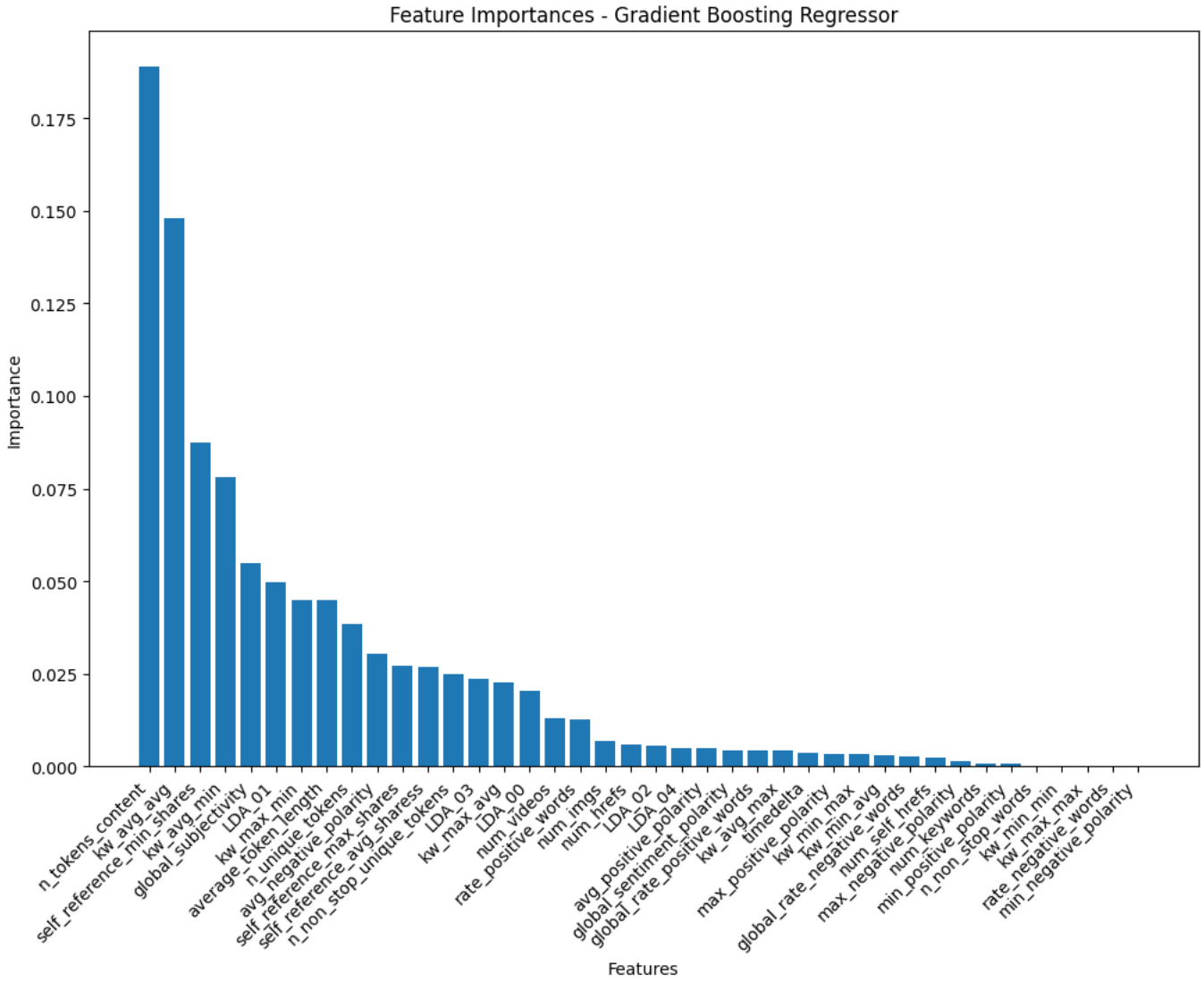
1)Information Gain:



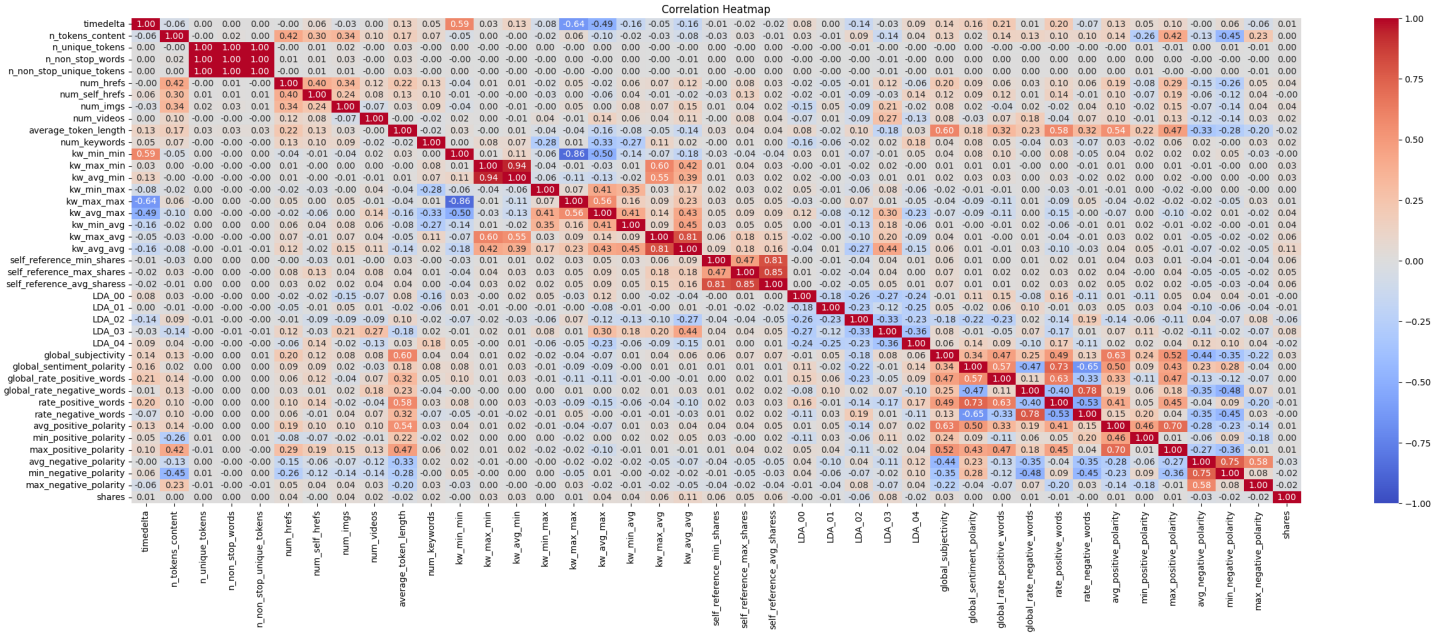
2)Random Forest Regressor:



3)Gradient Boosting:



4)Correlation Matrix:



How we select the numerical features:

1. Took the top 20 feature from each technique.
2. Count the occurrence of each feature in the 4 techniques.
3. Selected the repeated features in four techniques more than twice.

Selected Numerical Features:

- self_reference_max_shares
- LDA_01
- LDA_03
- kw_max_avg
- LDA_00
- self_reference_min_shares
- kw_min_avg
- LDA_02
- rate_positive_words
- kw_avg_max
- self_reference_avg_sharess
- kw_max_min
- kw_avg_avg
- num_videos

5)ANOVA Feature Selection for Categorical Data:

For the features **URL** and **TITLE**, each record in the dataset has a unique value. These features are not affected by changes in the dataset, and their values do not influence the mean squared error (MSE) or accuracy.

ANOVA values were calculated as follows:

- **channel_type**: 1.3975e-07
- **weekday**: 0.0023
- **isWeekEnd**: 0.0008

The small p-value indicates that the categorical variables are statistically significant in explaining the variability in the target variable 'shares'. Therefore, it suggests that they are likely to influence the number of shares.

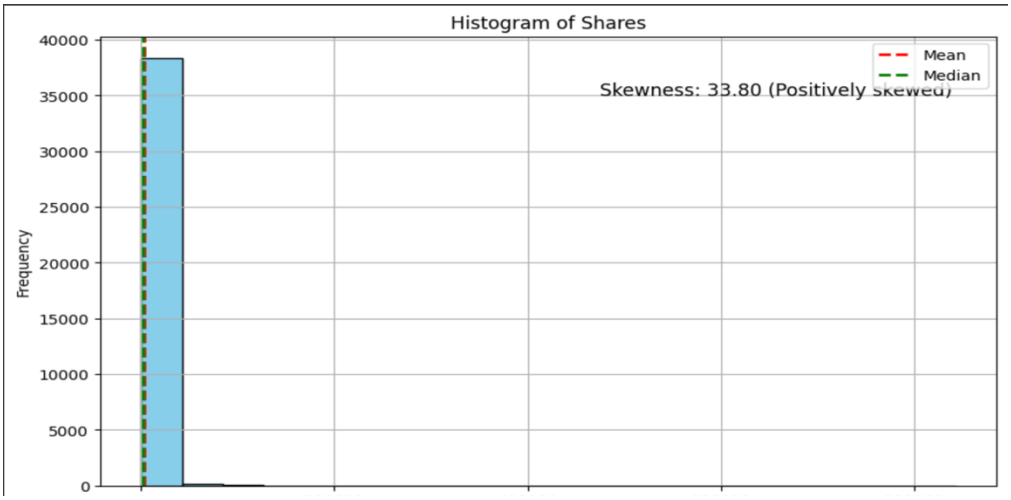
Selected Categorical Features:

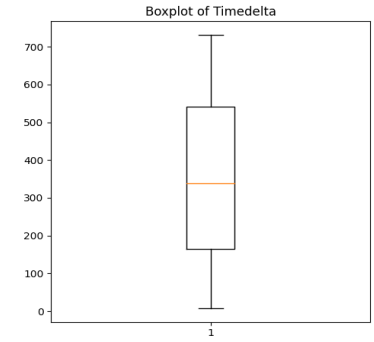
1. **channel_type**:
2. **weekday**
3. **isWeekEnd**

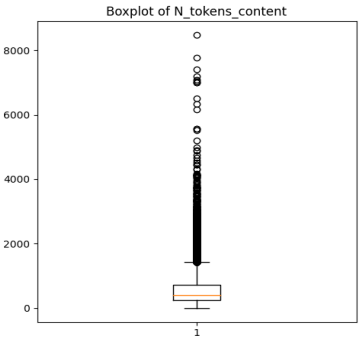
Feature Scaling:

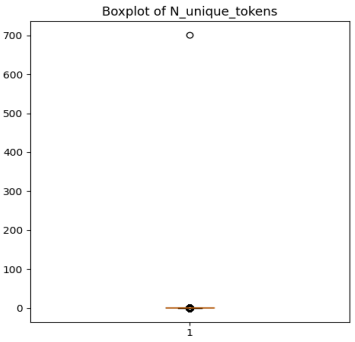
Standardization:

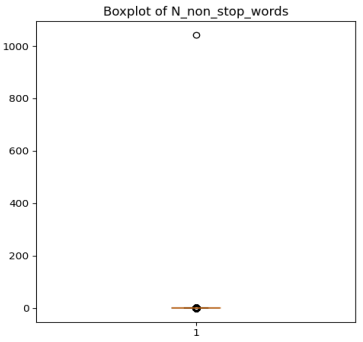
As our data have a lot of outliers and standardization is useful when the features in the dataset have varying scales so we applied it on the selected features and target column.

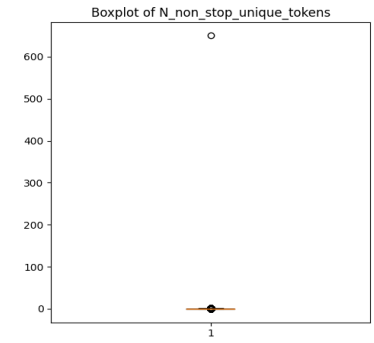


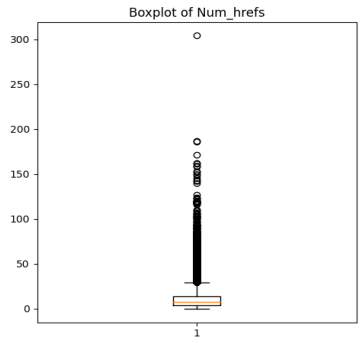


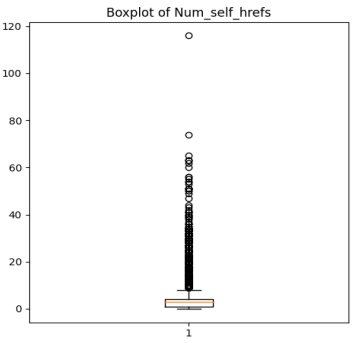


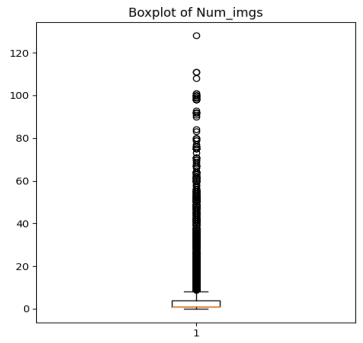


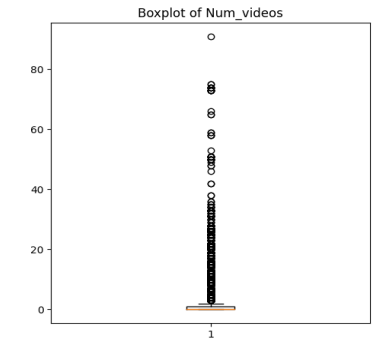


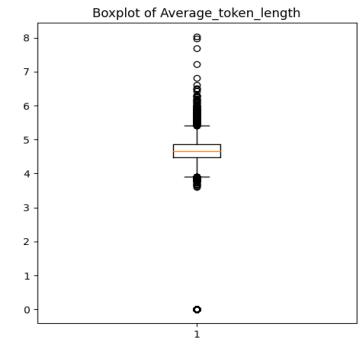


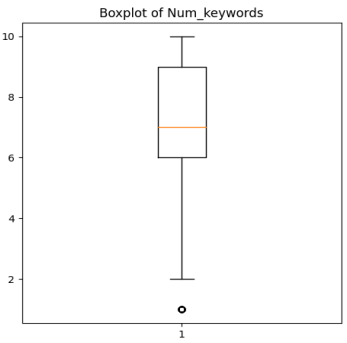


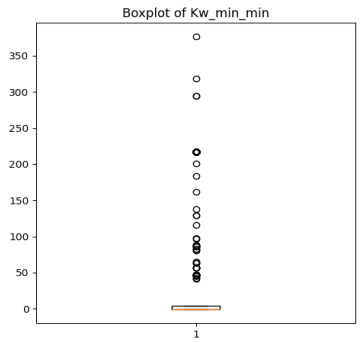


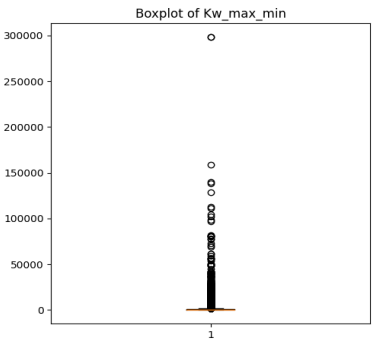


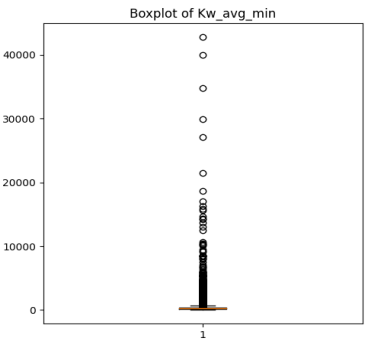


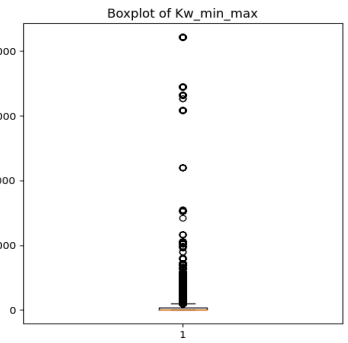


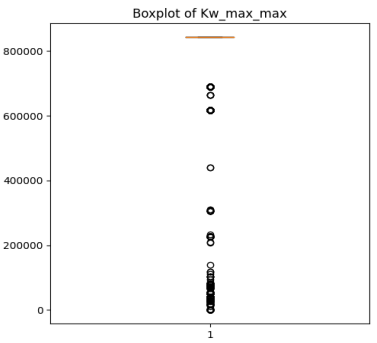












Modeling:

Regression techniques:

- **Train Without Cross Validation**:

	Linear Regression	Lasso regression	XGBoost	Random Forest
MSE	0.92	0.93	0.89	0.82
MAE	0.25	0.26	0.26	0.26
R2	2.13%	1.96%	5.99%	13.26%

- **Test Without Cross Validation**:

	Linear Regression	Lasso regression	XGBoost	Random Forest
MSE	1.1838	1.1850	1.1945	1.1785
MAE	0.2633	0.2659	0.2621	0.2596
R2	1.4%	1.3%	1.8%	2.19%

- **Test Cross Validation:**

	Linear Regression	Lasso regression	XGBoost	Random Forest
Mean MSE	0.9336	0.9347	1.121	0.9697
Mean MAE	0.2603	0.2619	0.2923	0.2619

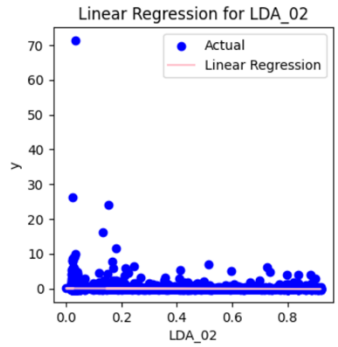
We Conclude from this that the best model to use is **Random Forest.**

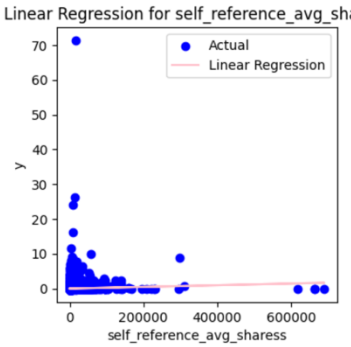
Sizes of the data (How we Split the data):

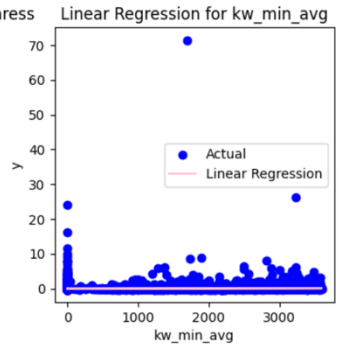
- Train Data: 80%
- Test Data: 20%
- Cross-validation: 5-fold:
  - Divided it to 5-fold.
  - Calculated The MSE and MAE for every fold separately.
  - Calculated the Mean of MSE and MAE.

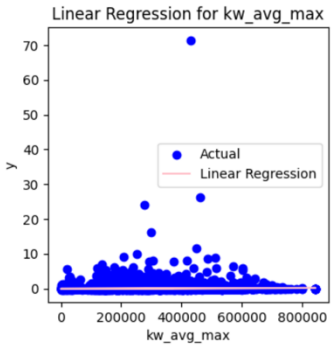
Screenshots of the resultant's regression line plots:

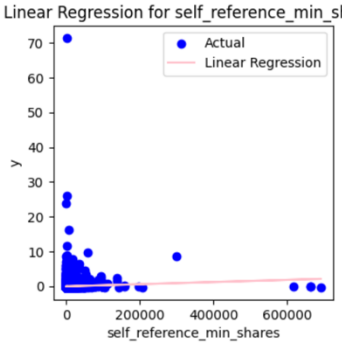
Linear Regression:

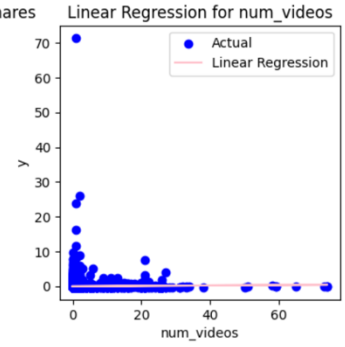


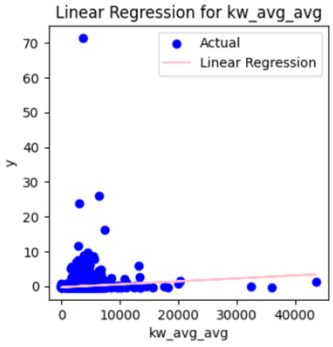


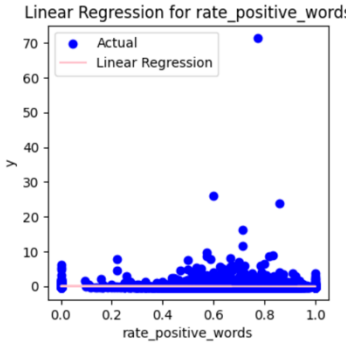


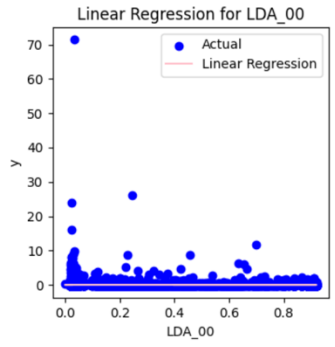


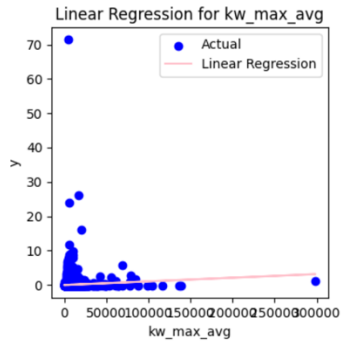


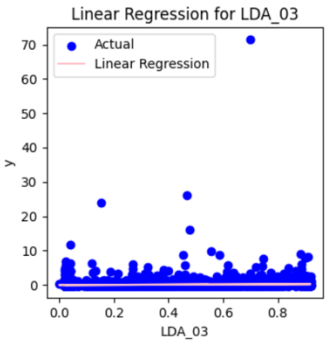


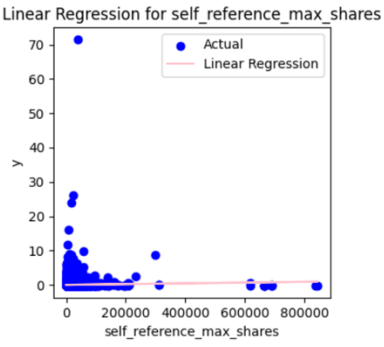


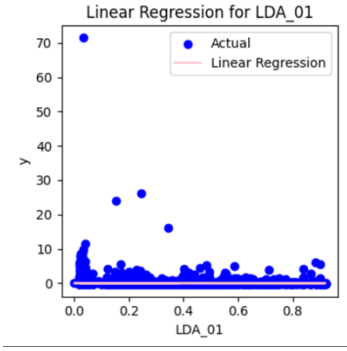


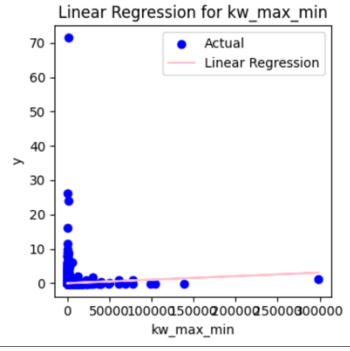




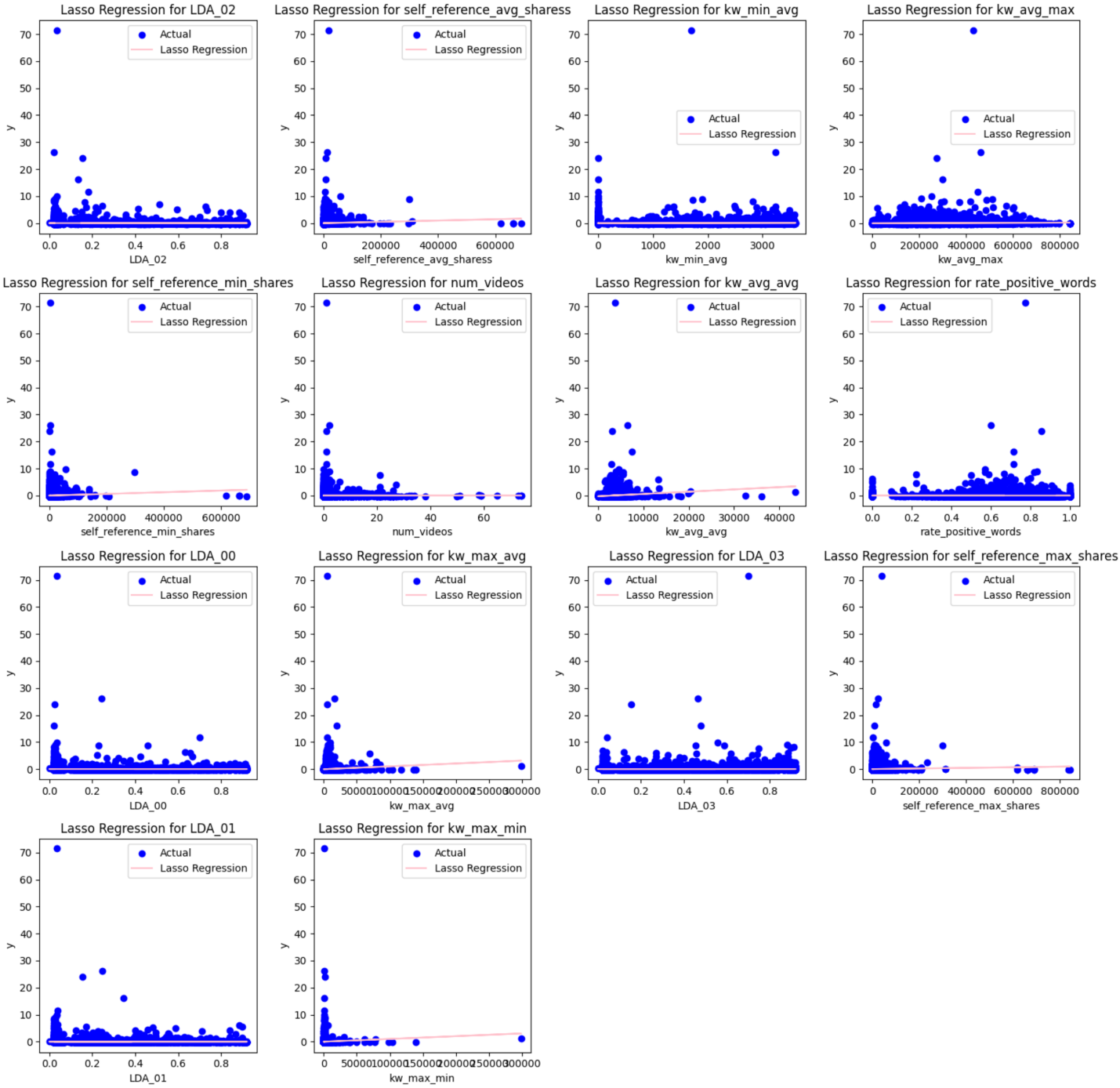








Lasso Regression:



Conclusion:

In the beginning, we had a lot of features to work with, so we tried different methods to pick out the most important ones. We focused on things like the content of the article, where it was published, and when it was published, thinking these would affect how many shares it got.

After a lot of trial and error, we narrowed down our features to a smaller set that seemed to have a big impact on predicting article shares. We looked at things like the number of shares of other articles by the same publisher, certain topics in the content, and how positive or negative the language was.

We tried out different regression techniques to see which one worked best. In the end, we found that using Random forest gave us the most accurate predictions.

This phase of the project validated our initial ideas regarding the importance of specific features in predicting article shares. It underscored the significance of employing cross-validation techniques to ensure the robustness and generalizability of our model.

Classification

Feature Selection:

Data Splitting:

We divided the dataset into two parts:

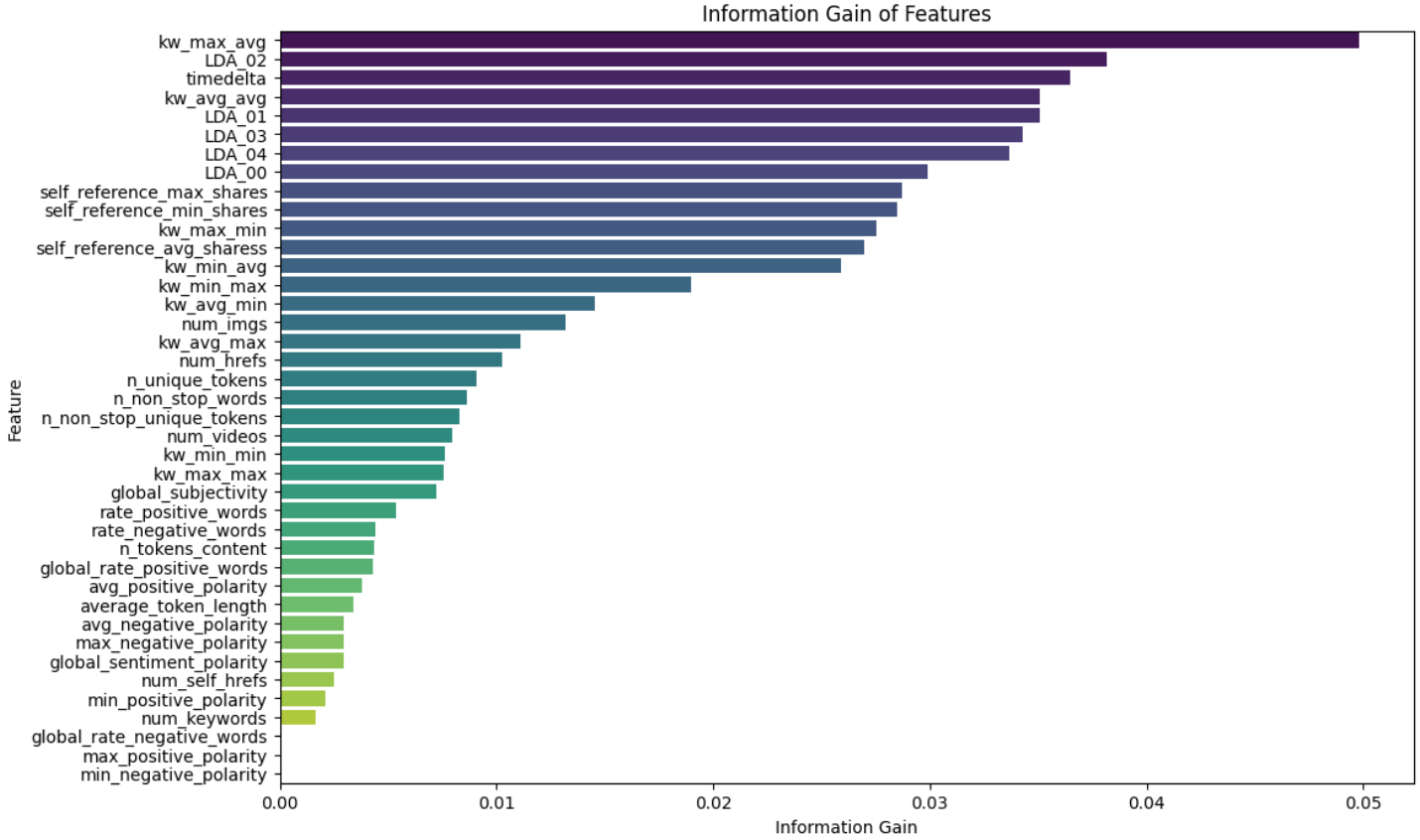
- **X**: This contains the independent variables. We decided to exclude all categorical data for feature selection, thus removing the following columns: **'Article_Popularity'**, **'Article_Popularity_encoded', channel_type**, **channel_type_encoded**, **url**, **url_encoded**, **title_encoded**, **title**, **weekday_encoded**, **weekday**, **isWeekEnd_0**, **isWeekEnd_1**, **isWeekEnd**.
- **Y**: This consists of the dependent variable or target column. We removed the **'Article_Popularity'** column from **X** and placed it in **Y**.

Feature Selection Techniques Used:

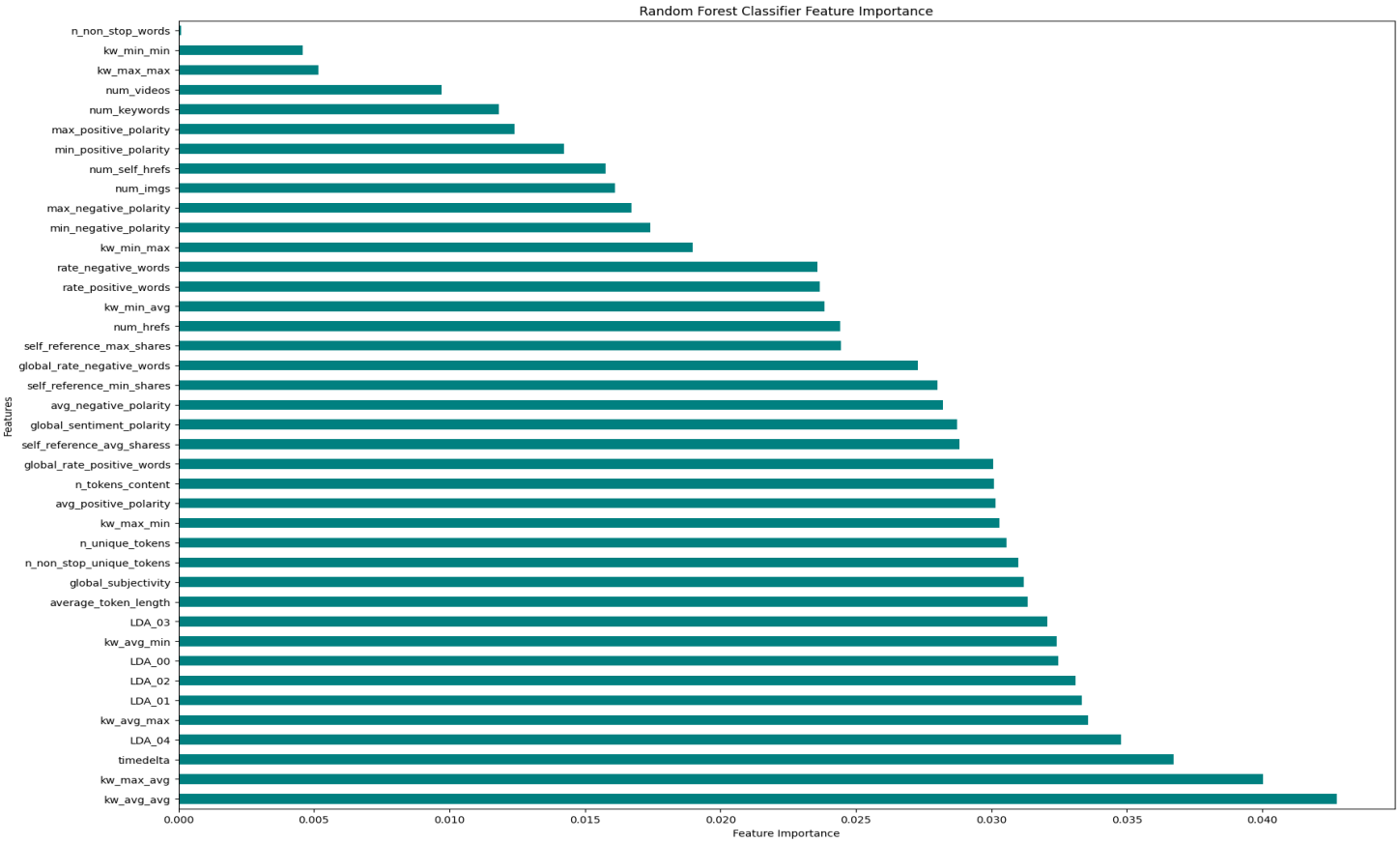
We employed five different techniques for feature selection:

1. **Information Gain Classifier**: Used to determine the importance of numerical features based on the entropy of the target variable.
2. **Random Forest Classifier**: Utilized the feature importance provided by a Random Forest model to rank the numerical features.
3. **Gradient Boosting Classifier**: Like the Random Forest Regressor, Gradient Boosting was used to rank numerical features based on their importance.
4. **Chi-Square**: assesses the association between categorical variables, determining if there's a significant relationship by comparing observed and expected frequencies.

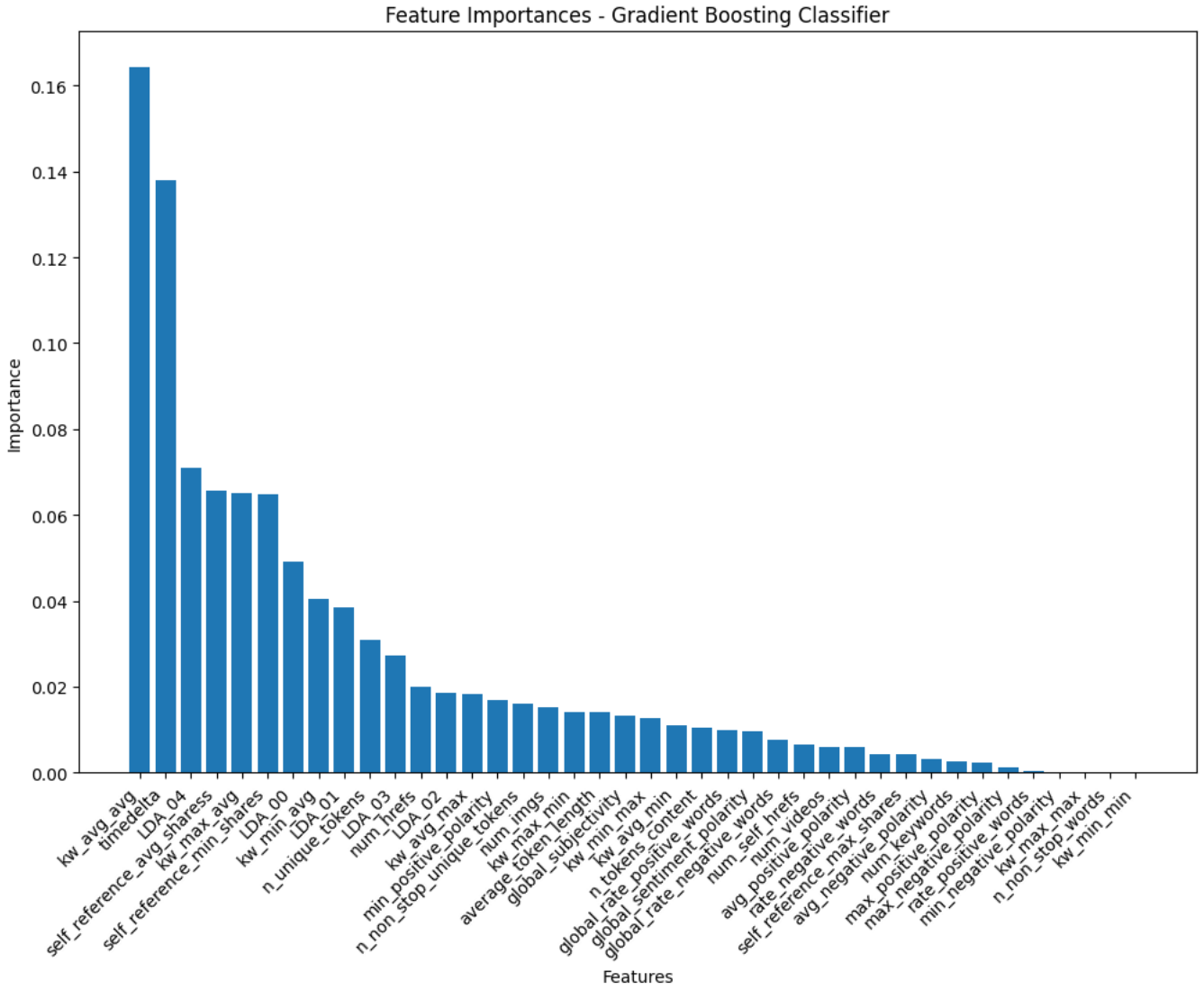
1)Information Gain:



2)Random Forest Classifier:



3)Gradient Boosting Classifier:



4)Chi-square:

```
Chi-Square Test Results:
                Feature  Chi2 Score        P-value
0  channel_type_encoded  130.545953   4.124940e-28
1      weekday_encoded  207.282377   1.125914e-44
2          isWeekEnd_0  122.979692   1.760538e-26
3          isWeekEnd_1  814.358419  3.328100e-176
```

A small p-value (< 0.05) typically indicates a significant association between the variables.

How we select the numerical features:

1. Took the top 20 feature from each technique.
2. Count the occurrence of each feature in the 3 techniques.
3. Selected the repeated features in four techniques more than twice.

Selected Features:

1. n_unique_tokens
2. timedelta
3. LDA_03
4. kw_min_avg
5. self_reference_avg_sharess
6. LDA_02
7. self_reference_min_shares
8. LDA_01
9. num_hrefs
10. kw_avg_max
11. LDA_04
12. kw_max_min
13. num_imgs
14. kw_avg_avg
15. kw_max_avg
16. LDA_00

Feature Scaling:

Standardization:

As our data have a lot of outliers and standardization is useful when the features in the dataset have varying scales so we applied it on the selected features.

Modelling:

- Logistic Regression:

C	Solver	Training Accuracy	Test Accuracy
0.1	liblinear	0.4323	0.4305
0.1	lbfgs	0.4326	0.4316
0.1	saga	0.4312	0.4302
1.0	liblinear	0.4322	0.4307
1.0	lbfgs	0.4326	0.4294
1.0	saga	0.4312	0.4302
10.0	liblinear	0.4323	0.4305
10.0	lbfgs	0.4329	0.4293
10.0	saga	0.4312	0.4302

Effect of C parameter:

- The C parameter in Logistic Regression controls the regularization strength, where smaller values specify stronger regularization.
- Lower values of C result in stronger regularization, which can help prevent overfitting by penalizing large coefficients. However, too much regularization might lead to underfitting.
- We observe that as C increases (e.g., from 0.1 to 1.0 to 10.0), there is a slight increase in training set accuracy in some cases, but the test set accuracy remains relatively stable or slightly decreases.
- This suggests that moderate regularization (C=1.0) might be more suitable for the given dataset, as it balances between preventing overfitting and maintaining model flexibility.

Effect of Solver algorithm:

- The Solver parameter in Logistic Regression determines the optimization algorithm used to fit the model to the training data.
- We see that three solvers were used: 'liblinear', 'lbfgs', and 'saga'.
- The 'liblinear' solver is suitable for small to medium datasets and supports both L1 and L2 regularization. It performed consistently across different C values.
- The 'lbfgs' solver is efficient for large-scale datasets and is suitable when L2 regularization is preferred. However, it showed convergence warnings for some combinations of hyperparameters, indicating potential convergence issues.
- The 'saga' solver is a variant of 'lbfgs' that supports L1 regularization and is well-suited for large datasets with L1 regularization. It also showed convergence warnings in some cases.

● Random Forest:

n_estimators	max_depth	Training Set Accuracy	Test Set Accuracy
50	None	1.0000	0.4787
50	10	0.5916	0.4865
50	20	0.9973	0.4743
100	None	1.0000	0.4842
100	10	0.5927	0.4857
100	20	0.9987	0.4844
200	None	1.0000	0.4865
200	10	0.5927	0.4888
200	20	0.9989	0.4902

Effect of n_estimators:

- The n_estimators hyperparameter determines the number of trees in the Random Forest ensemble.
- Increasing the number of trees can lead to better generalization performance by reducing overfitting, as the model learns from more diverse sets of trees.
- From the provided results, we can observe that as n_estimators increase (e.g., from 50 to 100 to 200), there's a slight improvement in test set accuracy in some cases, while in other cases, it remains relatively stable or even slightly decreases.
- The models with higher values of n_estimators tend to have better training set accuracy (close to 1.0), indicating that they may have overfit the training data to some extent.

Effect of max_depth:

- The max_depth hyperparameter controls the maximum depth of each tree in the Random Forest.
- Deeper trees can capture more complex patterns in the data, potentially leading to better performance on the training set. However, overly deep trees may also lead to overfitting.
- From the provided results, we observe that setting max_depth to None (allowing the trees to grow without restriction) results in very high training set accuracy (close to 1.0), indicating potential overfitting. However, this doesn't necessarily translate to better performance on the test set.
- Models with lower values of max_depth (e.g., 10) tend to have lower training set accuracy but may generalize better to unseen data, as indicated by their test set accuracy.

● Gradient Boosting:

n_estimators	max_depth	Training Set Accuracy	Test Set Accuracy
50	3	0.5039	0.4885
50	5	0.5603	0.4867
50	7	0.6754	0.4832
100	3	0.5219	0.4904
100	5	0.6046	0.4845
100	7	0.7537	0.4821
200	3	0.5475	0.4857
200	5	0.6730	0.4851
200	7	0.8633	0.4769

Effect of n_estimators:

- The n_estimators hyperparameter specifies the number of boosting stages, i.e., the number of weak learners (trees) to be included in the ensemble.
- Increasing the number of estimators can enhance the model's predictive performance by allowing it to capture more complex patterns in the data.
- From the provided results, we observe that as n_estimators increases (e.g., from 50 to 100 to 200), there's a mixed effect on the test set accuracy:
- In some cases, increasing n_estimators results in a slight improvement in test set accuracy.
- However, in other cases, further increasing n_estimators leads to diminishing returns or even a decrease in test set accuracy.
- The models with higher values of n_estimators tend to have better training set accuracy, indicating that they can capture more complex relationships in the training data. However, this doesn't always translate to improved generalization performance on the test set.

Effect of max_depth:

- The max_depth hyperparameter controls the maximum depth of each tree in the Gradient Boosting ensemble.
- Deeper trees can capture more complex interactions in the data but may also lead to overfitting.
- From the provided results, we observe that increasing max_depth generally results in higher training set accuracy.
- Models with higher values of max_depth (e.g., 7) tend to have higher training set accuracy but may also exhibit signs of overfitting, as indicated by lower test set accuracy.
- Shallower trees (lower max_depth) may lead to slightly lower training set accuracy but may generalize better to unseen data, as indicated by higher test set accuracy.

- KNN:

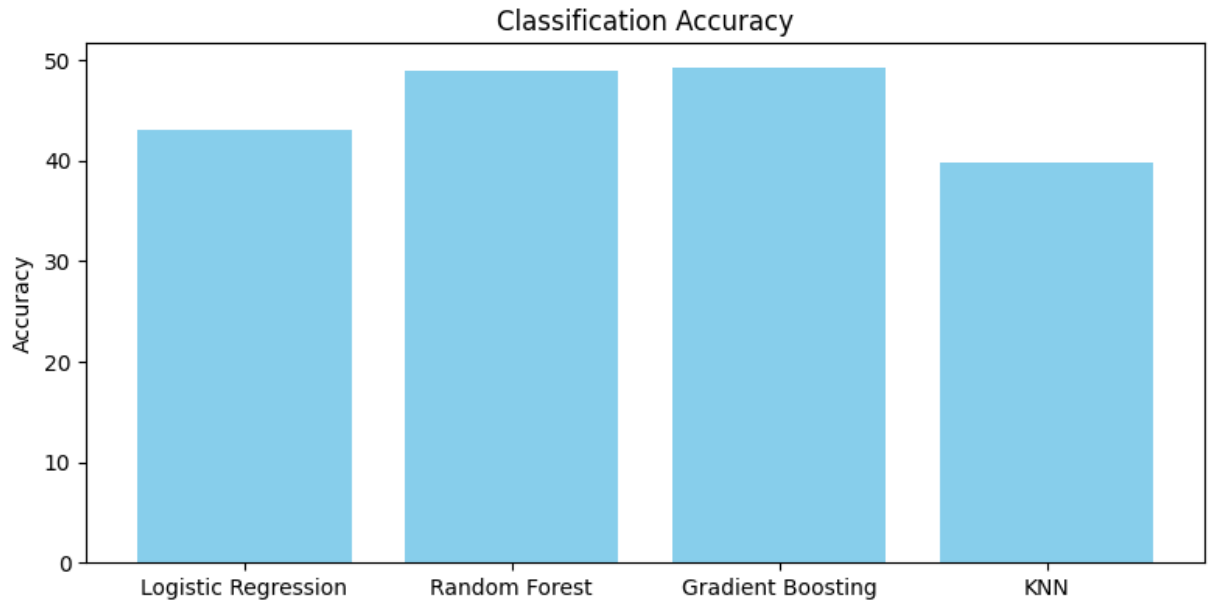
n_neighbors	weights	Training Set Accuracy	Test Set Accuracy
3	uniform	0.6430	0.3687
3	distance	1.0000	0.3713
5	uniform	0.5820	0.3885
5	distance	1.0000	0.3878
7	uniform	0.5532	0.3977
7	distance	1.0000	0.3973

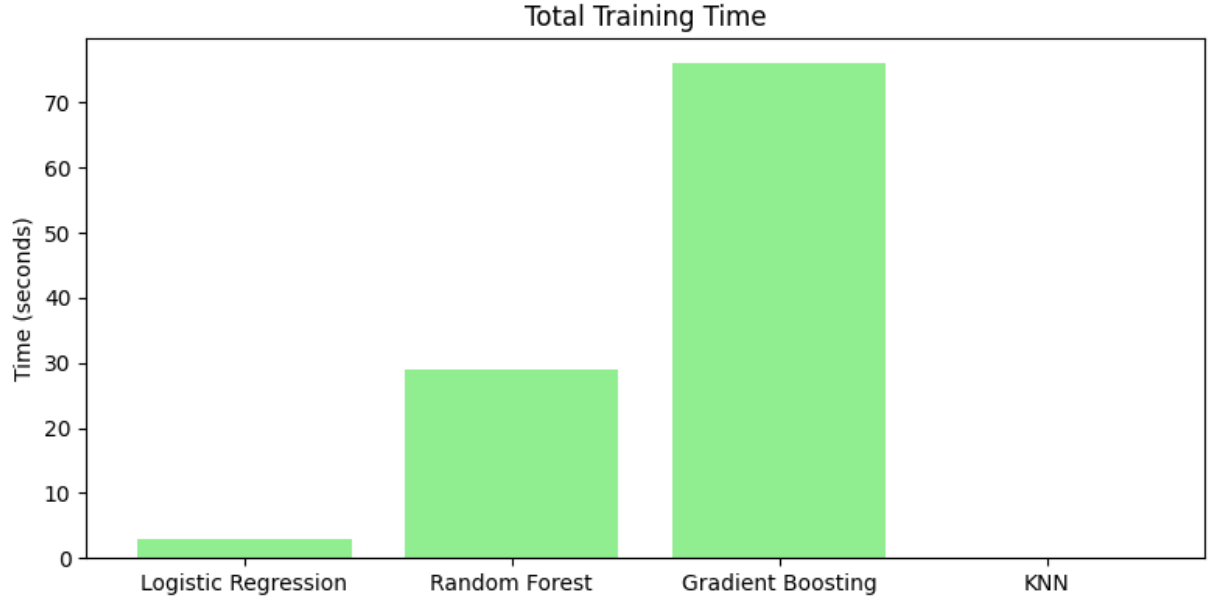
Effect of n_neighbors**:**

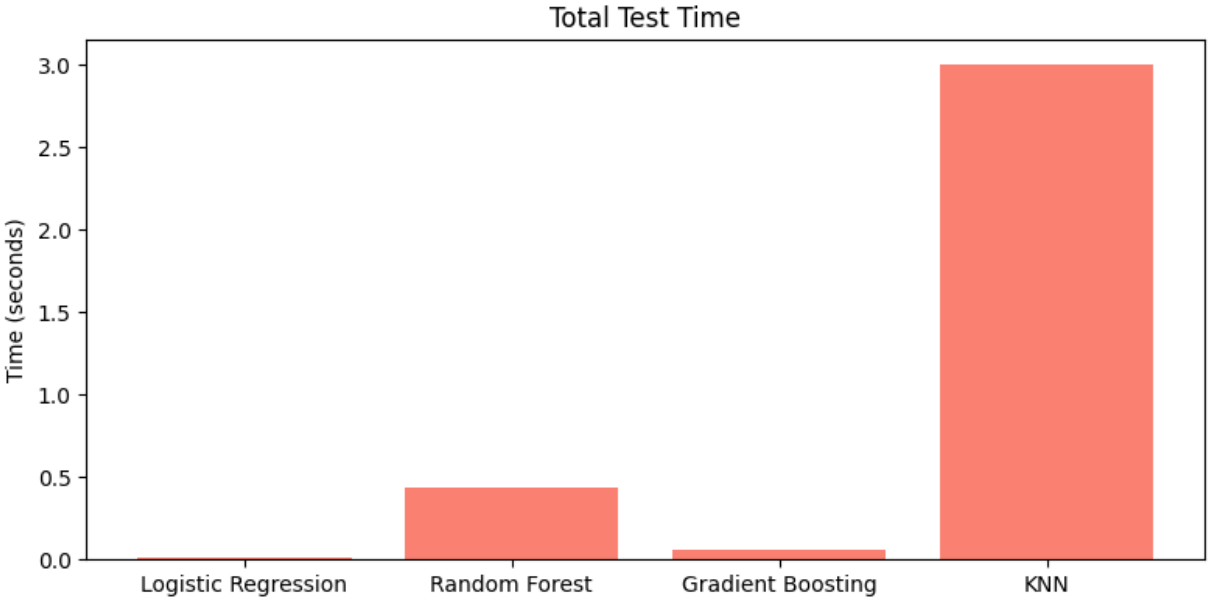
- The n_neighbors hyperparameter determines the number of nearest neighbors used in the classification.
- A smaller value of n_neighbors implies a more flexible decision boundary, potentially capturing intricate patterns in the data. However, it may also lead to overfitting, especially with noisy data.
- Conversely, a larger value of n_neighbors results in a smoother decision boundary, which might generalize better to unseen data but may underfit the training data if it's too large.
- From the provided results, we can observe that as the value of n_neighbors increases (from 3 to 5 to 7), there's a slight improvement in test set accuracy in some cases, indicating better generalization.
- However, it's essential to balance the value of n_neighbors to prevent underfitting or overfitting. For instance, in some cases, increasing n_neighbors beyond a certain point may not lead to further improvement and might even degrade performance.

Effect of weights:

- The weights hyperparameter determines the weight assigned to each neighbor during prediction. It can be set to 'uniform', where all neighbors have equal weight, or 'distance', where closer neighbors have a higher influence.
- When set to 'uniform', each neighbor contributes equally to the decision-making process, regardless of its distance from the query point.
- Conversely, when set to 'distance', closer neighbors have a higher weight, and their influence on the prediction is stronger.
- From the provided results, we can observe that the choice of weights has a relatively minor impact on model performance compared to n_neighbors. However, in some cases, using 'distance' weighting may lead to slightly better performance, as it assigns more importance to nearby neighbors, which may contain more relevant information.


Classification Accuracy
Accuracy
0
10
20
30
40
50
Logistic Regression
Random Forest
Gradient Boosting
KNN


Total Training Time
Time (seconds)
0
10
20
30
40
50
60
70
Logistic Regression
Random Forest
Gradient Boosting
KNN


Total Test Time
Time (seconds)
0.0
0.5
1.0
1.5
2.0
2.5
3.0
Logistic Regression
Random Forest
Gradient Boosting
KNN

Conclusion:

At the first, we were confused with a multitude of features demanding our attention. We kept in mind that factors like the article's content, where it was published, and when it appeared could significantly influence the number of shares it garnered.

We hypothesized that the same variables impacting regression outcomes would also have relevance in classification tasks, potentially resulting in improved accuracy.

In pursuit of this theory, we embarked on a journey of exploration, testing various classification techniques to get the most effective one. Our trial and error led us to discover that employing Gradient Boosting with specific configurations yielded the most accurate predictions.

This phase of our project not only affirmed our initial assumptions regarding the importance of certain features but also underscored the advantages of classification over regression in terms of predictive accuracy.