PROJECT REPORT ON

Online News Popularity

Submitted by
Angelin Sorna Swetha
Balaji M
C G Chandira Prawin
Karthik Raj
Kazi Tanveer Rehaman
Sruthi P

Mentored By
Mr. Romil Gupta

ACKNOWLEDGEMENT

We would like to thank our mentor **Mr. Romil Gupta**, SME(SQL Server), Scrum Master at IBM for providing his valuable guidance and suggestions over the course of our Project Work. We also thank him for his continuous encouragement and interest towards our Project work.

We are extremely grateful to our all teaching and non-teaching staff members of **GREAT LEARNING**, who showed keen interest and inquired us about our development and progress.

We would like to express our gratitude to all teaching and non-teaching staff members **of Great Lakes Institute of Management**, for providing their support and guidance for our project.

We greatly admire and acknowledge the constant support we received from our friends and team members for all the effort and hard work that they have put into completing this project.

Table of Contents

1 INTRODUCTION

Predictive analysis using machine learning has been gaining popularity in recent times. It has been vital to the growth of companies like Target, Amazon and Netflix in identifying customer behavior through Business Intelligence models. These instances suggest clearly that there are business opportunities that needs to be harnessed in the field of predictive analysis.

Applying predictive analysis to articles on the web can be vital in extending a creator into a large-scale distributor of his/her content. According to Kelwin et. al[1], current marketing techniques make use of a business intelligence approach called Decision Support Systems (DSS) to predict popularity before the actual marketing of the product. DSS is an information system that supports business or decision-making activities for organizations. Adaptive Business Intelligence (ABI) provides the power of prediction and optimization working together to increase popularity of articles on the web. Thus, knowing in advance the articles, which are likely to become popular, based on the article content, is vital for business intelligence approaches. A low share count is a clear indication to make changes to the articles content to improve popularity. Social media platforms have given new opportunities for users to become large-scale distributors of their content.

According to the Pew Research Centre, social media usage between 2005-2015 shows that 65% of adults now use social networking sites - a nearly tenfold jump in the past decade. Although social media platforms provide new opportunities for users to distribute their content, they have also led to a huge bias in terms of number of articles getting popular. A minuscule percentage of the articles become popular, and their share counts increase rapidly, almost exponentially, leaving the other articles virtually unknown to social media users. Understanding what makes one article more popular than other, observing its popularity dynamics, and being able to predict its popularity has thus attracted a lot of interest in the past few years.

Prediction of article popularity, especially those from Mashable, have been studied by Kelwin et.al[1] and Dumont et.al[2]. However, a detailed comparison and analysis of the most popular regression models applied on Mashable articles has not been studied yet. Hence, there is a need to experiment further to produce new and interesting results in predictive analysis using machine learning regression models, applied to articles from Mashable. This paper aims to harness machine learning algorithms to predict share counts of Mashable articles. The experiments focus on predicting a single value, the share count. Share count will refer to the value obtained from Mashable for the article under scrutiny and represents the number of times the article is shared across any social media platform.

1.1 Need for the project

For content writers/editors, advertisers, content creators and website owners to make it easier for them to decide whether to publish an article or not and to make them understand the scope of improvement in the article.

1.2 Scope

This project helps content writers and editors to be able to understand how popular their article would become prior to publication, which would provide a foresight to their articles popularity status and based on the results, improvisation of the articles can be done.

2 LITERATURE REVIEW

Md. Taufeeq Uddin et.al[3] in their work 'Predicting the popularity of online news from content metadata' mentioned about Popularity prediction of online news to predict the future popularity of news article prior to its publication estimating the number of shares, likes, and comments. Yet, popularity prediction is a challenging task due to various issues including difficulty to measure the quality of content and relevance of content to users; prediction difficulty of complex online interactions and information cascades; inaccessibility of context outside the web; local and geographic conditions; social network properties. This paper focused on popularity prediction of online news by predicting whether users share an article or not, and how many users share the news adopting before publication approach. This paper proposes the gradient boosting machine for popularity prediction using features that are known before publication of articles. The proposed model showed around 1.8% improvement over previously applied techniques on a benchmark dataset. This model also indicates that features extracted from articles keywords, publication day, and the data channel are highly influential for popularity prediction.

Xuandong Lei et.al[4] in their paper titled 'Is Your Story Going to Spread Like a Virus? Machine Learning Methods for News Popularity Prediction' tried different methods to predict the popularity before its publication. Using over 39000 sample articles from Mashable.com, it tried to address the problem both as a numerical and a multinomial classification problem. Specifically, they applied linear regression, polynomial regression, GAM with smoothing splines and Lasso to predict the exact shares. The best CV error result was 0.7649 acquired by GAM with smoothing splines. And then they used SVM, Random Forest and Bagging to predict popularity, which is divided into four categories, for each article. In such case, Random Forest gives the best result, which achieves 50.4% prediction accuracy.

K.Fernandes et.al[1] in their work 'A Proactive Intelligent Decision Support System for Predicting the Popularity of Online News', proposed a novel and proactive Intelligent Decision Support System (IDSS) that analysed articles prior to their publication. Using a broad set of extracted features (e.g., keywords, digital media content, earlier popularity of news referenced in the article) the IDSS first predicted if an article will become popular. Then, it optimized a subset of the articles features that can more easily be changed by authors, searching for an enhancement of the predicted popularity probability. Using a large dataset, with 39,000 articles from the Mashable website, they performed a robust rolling windows evaluation of five state of the art models. The best result was provided by a Random Forest model with a discrimination power of 73%. Moreover, several stochastic hill climbing local searches were explored. When optimizing 1000 articles, the best optimization method obtained a mean gain improvement of 15 percentage points in terms of the estimated popularity probability. These results attest the proposed IDSS as a valuable tool for online news authors.

E Hensinger et.al[5] in the work 'Modelling and predicting news popularity' explored the problem of learning to predict the popularity of an article in online news media.By "popular" it meant an article that was among the "most read" articles of a given day in the news outlet that published it. It showed that this cannot be modelled simply as the binary classification task of separating popular from unpopular articles, thereby assuming that popularity is an absolute property. Instead, they proposed to view popularity in the perspective of a competitive situation where the popular articles are those which were the most appealing on that particular day. This leads to the notion of an "appeal" function, to model which we use a linear

function in the bag of words representation. The parameters of this linear function are learnt from a training set formed by pairs of documents, one of which was popular and the other which appeared on the same page and date, without becoming popular. To learn the appeal function they used Ranking Support Vector Machines, using data collected from six different outlets over a period of 1 year. It showed that method can predict which articles will become popular, as well as extracting those keywords that mostly affect the appeal function. This also enables to compare different outlets from the point of view of their readers' preference patterns.

R Shreyas et.al[6] in the paper 'Predicting popularity of online articles using Random Forest regression's, the Random Forest regression model is used to predict popularity of articles from the Online News Popularity data set. The performance of the Random Forest model was investigated and compared with other models. Impact of standardization, regularization, correlation, high bias/high variance and feature selection on the learning models were also studied. Results indicate that, the Random Forest approach predicted popular/unpopular articles with an accuracy of 88.8%. Predictive analysis has been vital to the growth of companies like Target, Amazon and Netflix in identifying customer behavior through Business Intelligence models. These instances suggest clearly that there are business opportunities to be harnessed in the field of predictive analysis. Applying predictive analysis to articles on the web can be vital in extending a creator into a large-scale distributor of his/her content.

3 OVERVIEW

3.1 Dataset Overview:

Online News Popularity Data Set from UCI Machine Learning Repository

3.2 Problem Statement

The dataset summarizes a heterogeneous set of features about articles published by Mashable in a period of two years. The objective is to build a model which can predict the number of shares of an online article at highest possible accuracy.

3.3 Tools Used

- Programming Language: Python
- Visualizations: Python

4 DATA PRE-PROCESSING

4.1 Shape of the Data

The dataset has 39644 rows with 61 features. The features are already dummied from the beginning.

4.2 Feature Information

Following are the main features of the dataset, and the rest are the dummies of these features.

Columns	Descriptions
url	URL of the article (non-predictive)
timedelta	Days between the article publication and the dataset acquisition (non-predictive)
n_tokens_title	Number of words in the title
n_tokens_content	Number of words in the content
n_unique_tokens	Rate of unique words in the content
n_non_stop_words	Rate of non-stop words in the content
n_non_stop_unique_tokens	Rate of unique non-stop words in the content
num_hrefs	Number of links
num_self_hrefs	Number of links to other articles published by Mashable
num_imgs	Number of images
num_videos	Number of videos
average_token_length	Average length of the words in the content
num_keywords	Number of keywords in the metadata
data_channel_is_lifestyle	Is data channel 'Lifestyle'?
data_channel_is_entertainment	Is data channel 'Entertainment'?
data_channel_is_bus	Is data channel 'Business'?
data_channel_is_socmed	Is data channel 'Social Media'?
data_channel_is_tech	Is data channel 'Tech'?
data_channel_is_world	Is data channel 'World'?
kw_min_min	Worst keyword (min. shares)
kw_max_min	Worst keyword (max. shares)
kw_avg_min	Worst keyword (avg. shares)
kw_min_max	Best keyword (min. shares)
kw_max_max	Best keyword (max. shares)
kw_avg_max	Best keyword (avg. shares)
kw_min_avg	Avg. keyword (min. shares)
kw_max_avg	Avg. keyword (max. shares)
kw_avg_avg	Avg. keyword (avg. shares)
self_reference_min_shares	Min. shares of referenced articles in Mashable
self_reference_max_shares	Max. shares of referenced articles in Mashable
self_reference_avg_sharess	Avg. shares of referenced articles in Mashable

weekday_is_monday	Was the article published on a Monday?
weekday_is_tuesday	Was the article published on a Tuesday?
weekday_is_wednesday	Was the article published on a Wednesday?
weekday_is_thursday	Was the article published on a Thursday?
weekday_is_friday	Was the article published on a Friday?
weekday_is_saturday	Was the article published on a Saturday?
weekday_is_sunday	Was the article published on a Sunday?
is_weekend	Was the article published on the weekend?
LDA_00	Closeness to LDA topic 0
LDA_01	Closeness to LDA topic 1
LDA_02	Closeness to LDA topic 2
LDA_03	Closeness to LDA topic 3
LDA_04	Closeness to LDA topic 4
global_subjectivity	Text subjectivity
global_sentiment_polarity	Text sentiment polarity
global_rate_positive_words	Rate of positive words in the content
global_rate_negative_words	Rate of negative words in the content
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	Avg. polarity of positive words
min_positive_polarity	Min. polarity of positive words
max_positive_polarity	Max. polarity of positive words
avg_negative_polarity	Avg. polarity of negative words
min_negative_polarity	Min. polarity of negative words
max_negative_polarity	Max. polarity of negative words
title_subjectivity	Title subjectivity
title_sentiment_polarity	Title polarity
abs_title_subjectivity	Absolute subjectivity level
abs_title_sentiment_polarity	Absolute polarity level
shares	Number of shares (target)

4.3 Data Types

The different datatypes available in the dataset were analyzed to find out non-numerical features. The information of the dataset is as follows:

	No of Features
float64	59
int64	1
object	1

From this, we can see that there is only 1 feature of the object datatype. When checked we could see that is the “url” column and it can be dropped. All the input features are “float” type and the output is of type “int”.

4.4 Missing Value Analysis

There are no null values in the dataset.

```
print("Number of Null Values : ",df.isnull().sum().sum() )
```

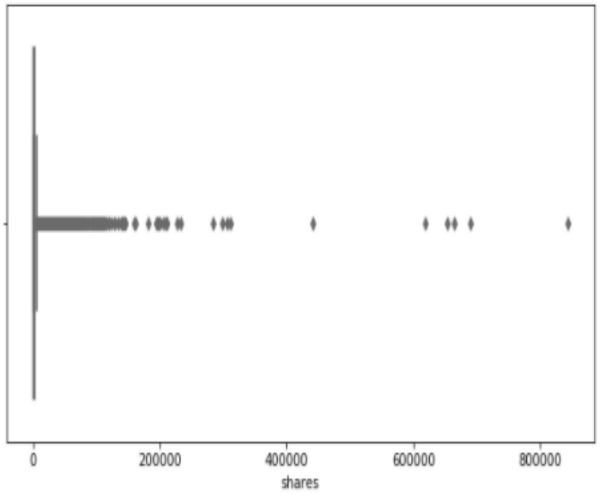
```
Number of Null Values :  0
```

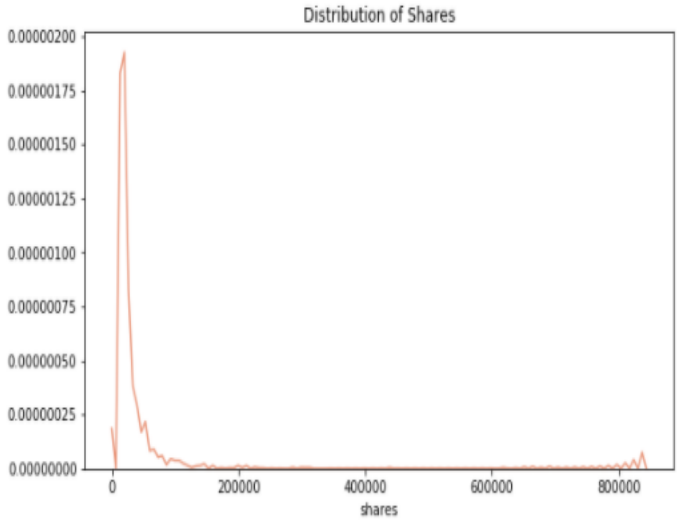
5 EXPLORATORY DATA ANALYSIS (EDA)

5.1 Univariate Analysis

5.1.1 Response Variable

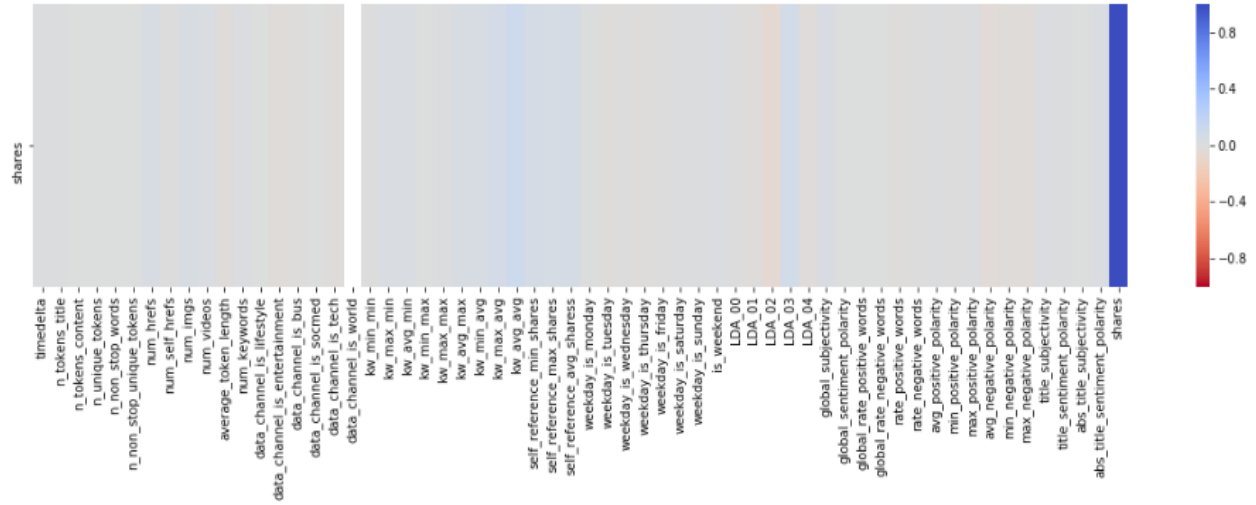
The target column ‘shares’ is a continuous variable column. It has values starting from 1 up to 843300.The distribution of this column is right skewed with many outliers. Most of the values are in between the range of 1-100000.





5.1.2 Correlation with Response

Correlation of Response with all the features in dataset was checked to find out which variables have more impact / relation with the target column.

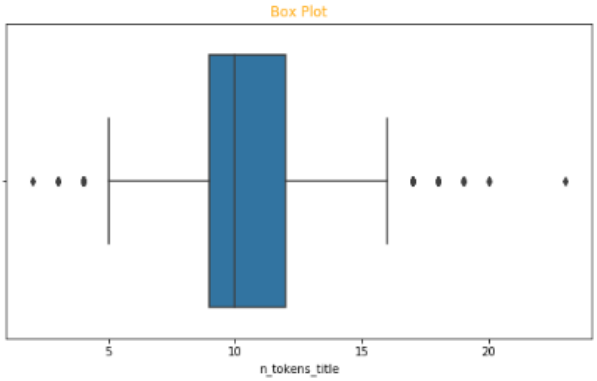


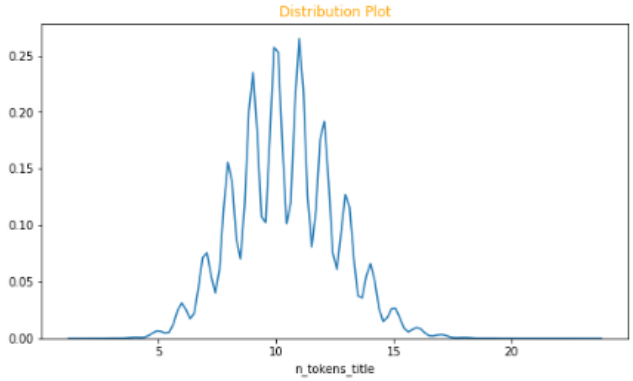
Top 8 features highly correlated with target variable (Shares) are:

num_hrefs	0.045404
self_reference_max_shares	0.047115
self_reference_min_shares	0.055958
self_reference_avg_sharess	0.057789
LDA_02	0.059163
kw_max_avg	0.064306
LDA_03	0.083771
kw_avg_avg	0.110413
shares	1.000000

5.1.3 *n_tokens_title*

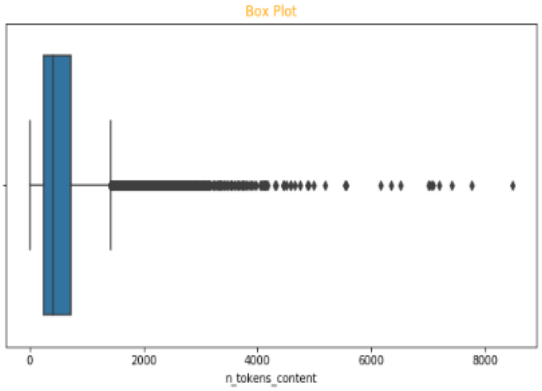
- We can observe many peaks in the model. It is multi modal.
- Most of the articles published have average of 8 to 12 words in the title.

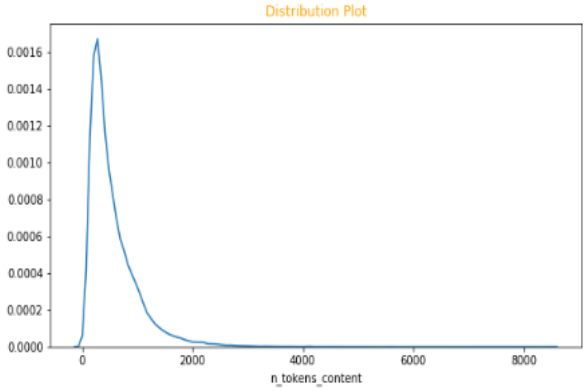




5.1.4 *n_tokens_content*

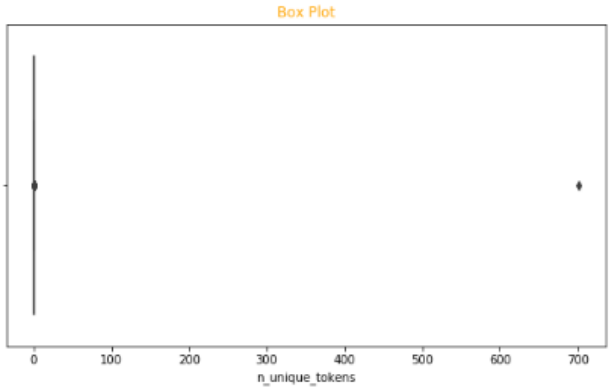
- The plot indicates that the distribution of *"n_tokens_content"* is right skewed.
- Most of the articles are in range Between 1000 to 2000 words.

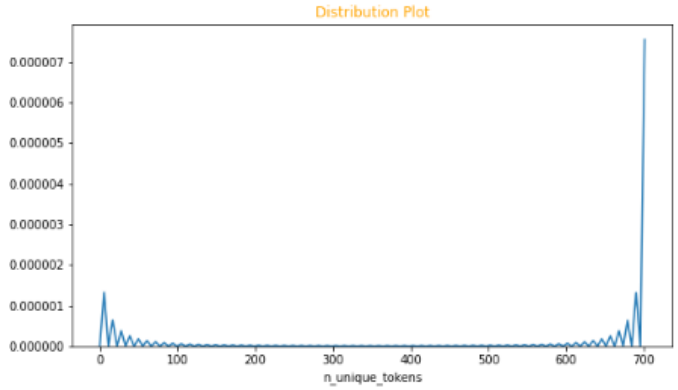




5.1.5 *n_unique_tokens*

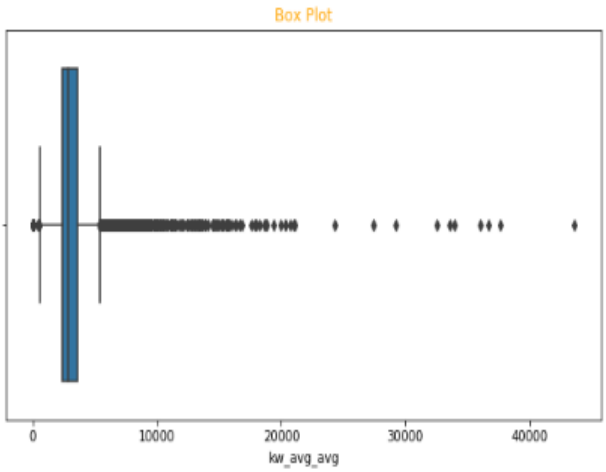
- The distribution plot shows that the "*n_unique_tokens*" has Multi mode.
- High peak at value is 700.These refer to common words and tokens.
- There are 2 noticeable concentration of values between 0 to 100 and 600 to 700.
- Unique tokens are quite less.

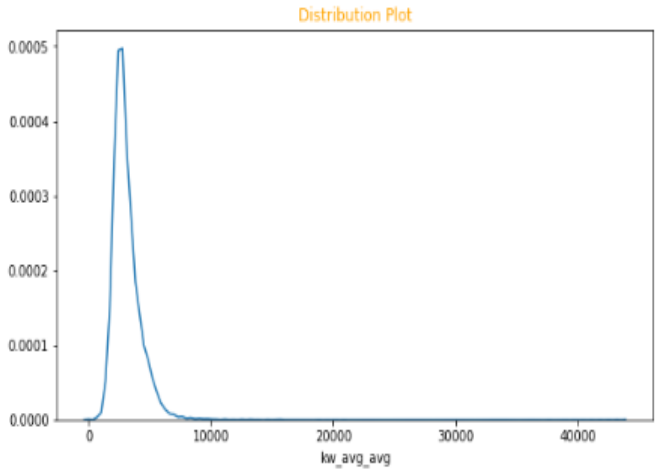




5.1.6 *kw_avg_avg*

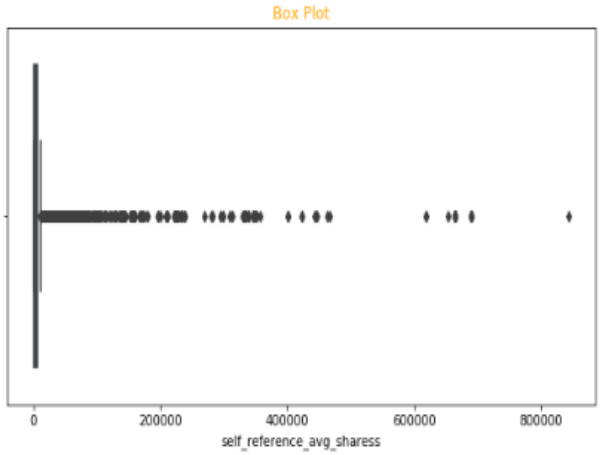
- Distribution plot is right skewed.
- There are nearly 1000 keywords, which does influence shares. These keywords are generated in topics of high interest.

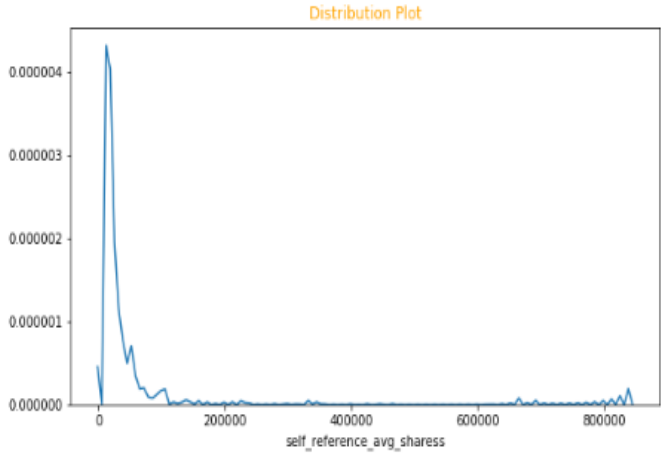




5.1.7 self_reference_avg_shares

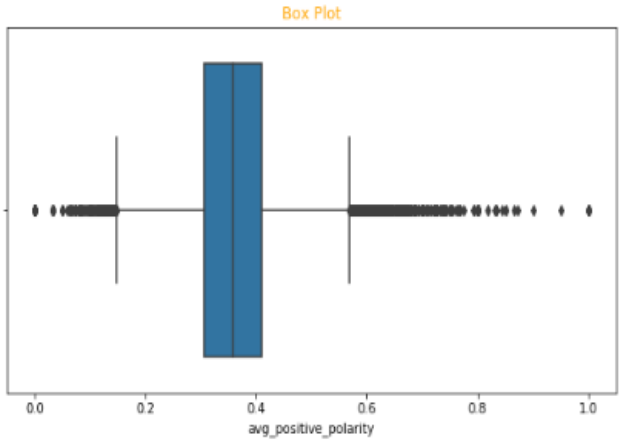
- Distribution plot is right skewed.
- Leptokurtic plot.
- Most of the self-ref articles have a mean share of 100000.So referenced articles are not preferred by readers. They want to read the news only in brief.

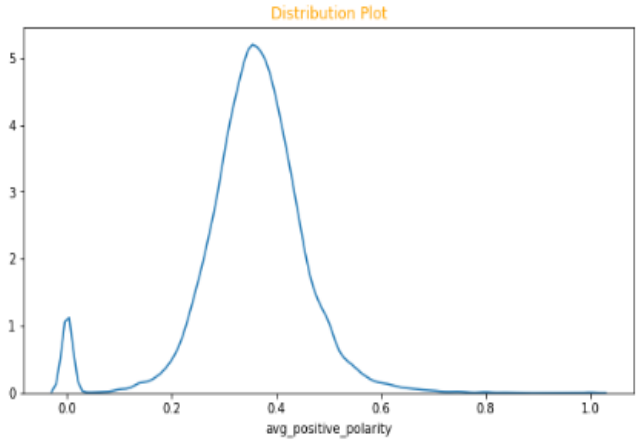




5.1.8 avg_positive_polarity

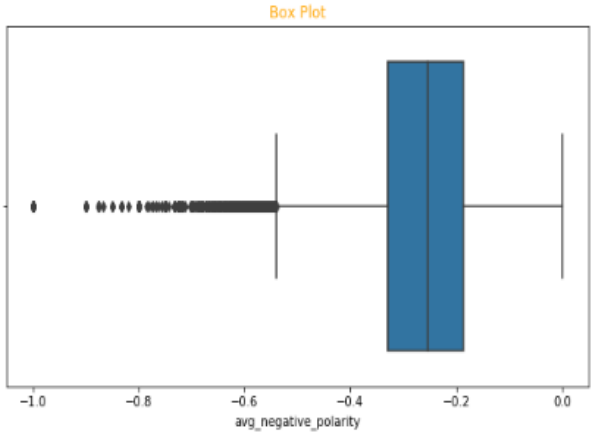
- Around mean value, we can observe a bell shape curve.
- Also, we can see a peak at 0 and also right skewness.
- The plot shows that the distribution has two modes.

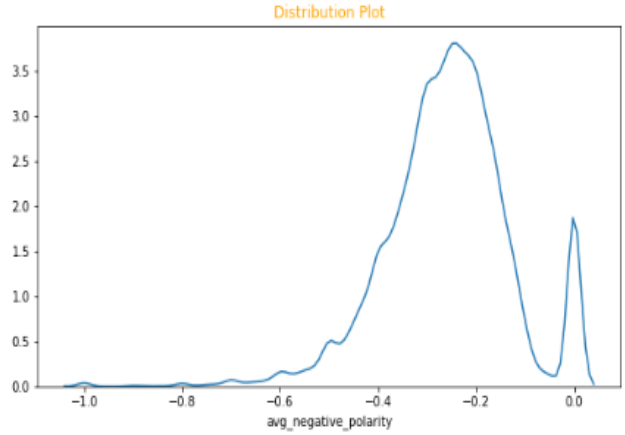




5.1.9 *avg_negative_polarity*

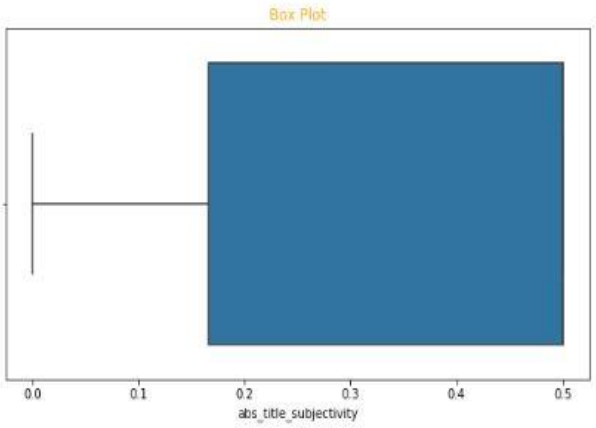
- Around mean value, we can observe a bell curve.
- In addition, we can see a peak at 0 and left skewness.

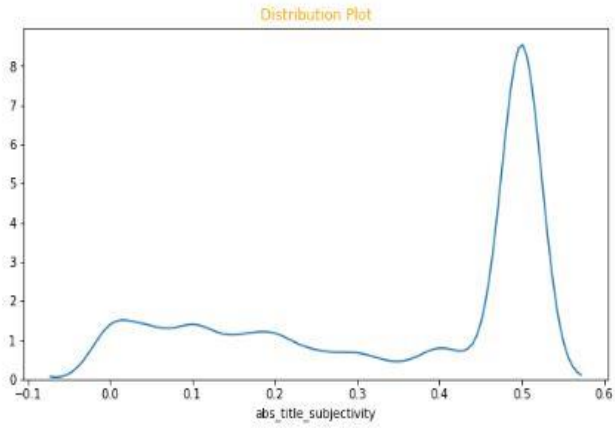




5.1.10 *abs_title_subjectivity*

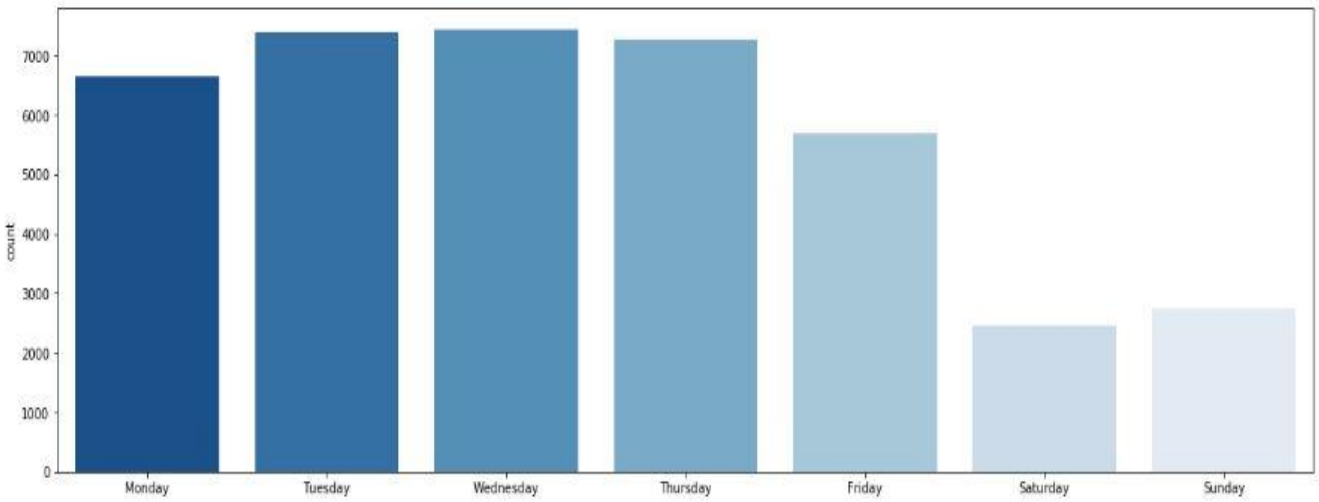
- Between 0.4 and 0.6 we can see a bell shaped curve.
- Distribution plot is left skewed





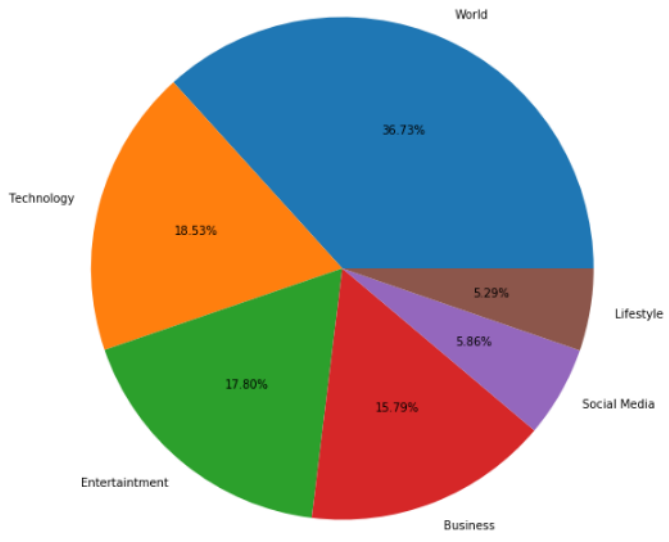
5.1.11 *Articles Published per Day of the Week*

- Most of the articles are published during Weekdays, Highest on Wednesday, followed by Tuesday and Thursday.
- There is no high margin of difference in the articles published during weekdays.
- During weekends, articles published are less than 3000.



5.1.12 Articles published per Data Channel

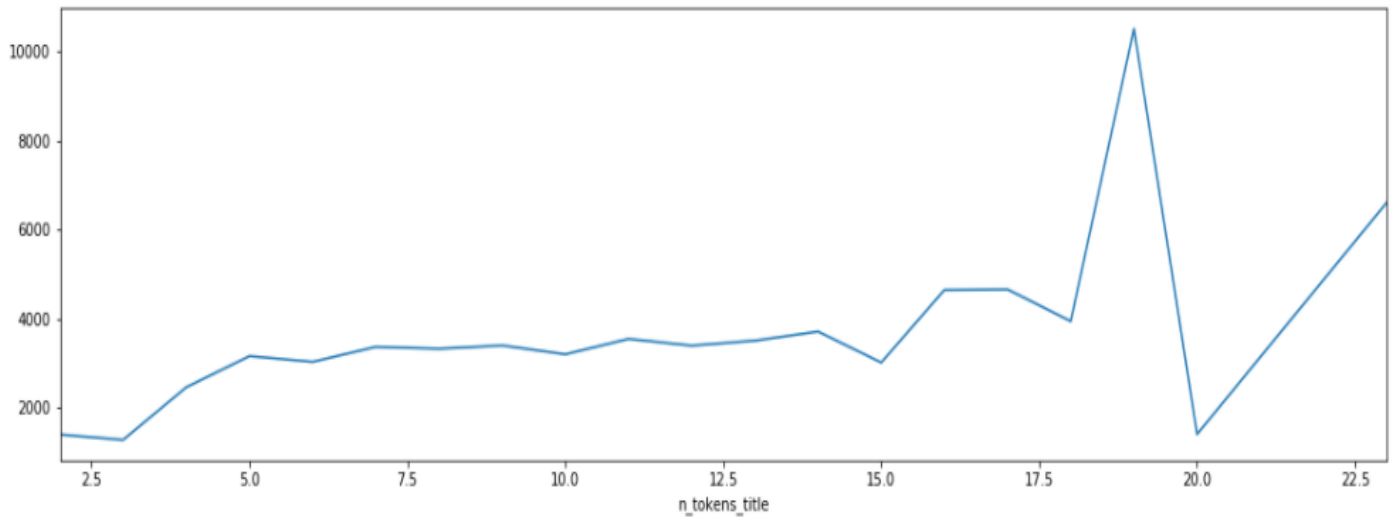
- World and Technology account for more than 50% of the articles published.
- Fields such as Lifestyle and Social Media has the least share.



5.2 Bivariate Analysis

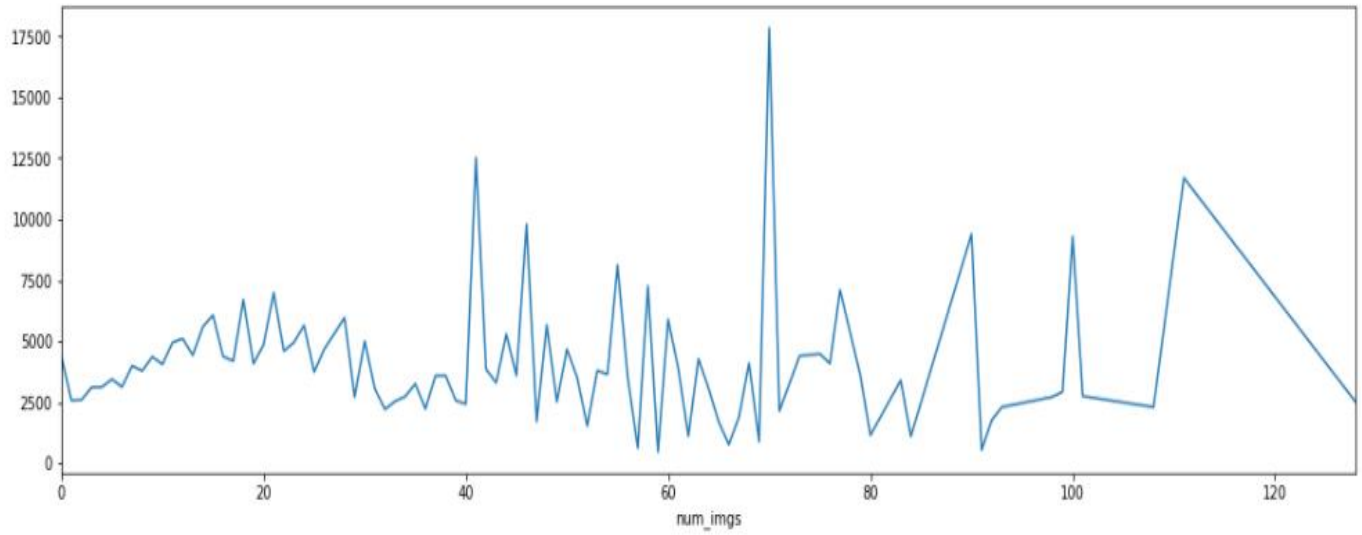
5.2.1 n_tokens_title

- The articles which contain 18-19 words in the title have received the maximum amount of shares.



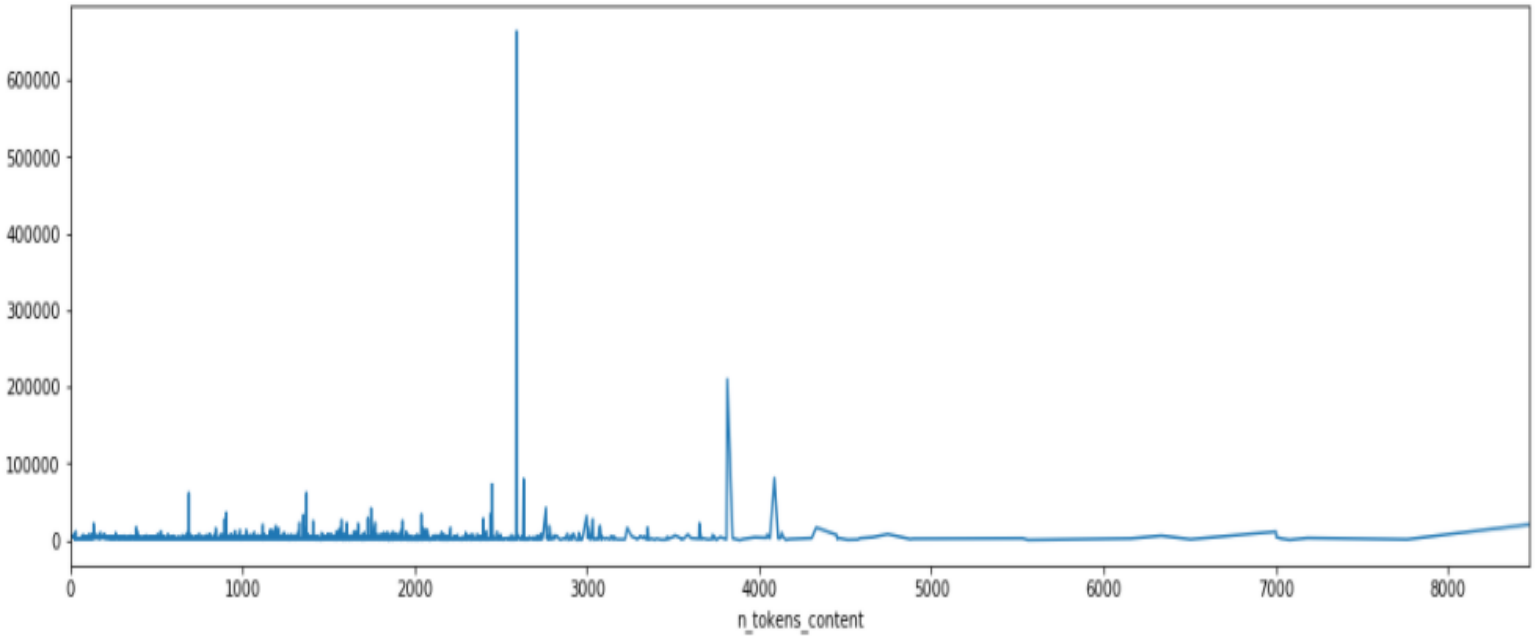
5.2.2 Number of images vs shares

- The nature of the above graph is very fluctuating. So we cannot conclude that increase in images will surely lead to increase in no of shares or otherwise.



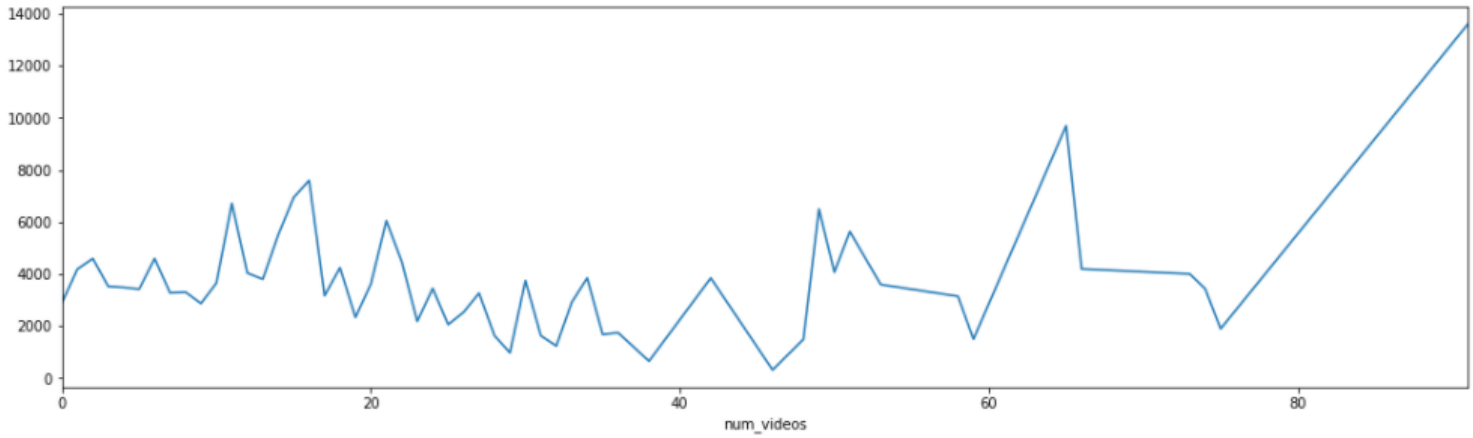
5.2.3 n_tokens_content vs. shares

- There are two heavily noticeable peaks in this `n_tokens_content` vs `shares` line graph.
- One is very high that smaller peaks are not even visible. This comes at the range of 2500-3000 average.
- The other peak is high only to a certain level. This is at the range of 3800-4000 average.
- Generally, people do not like to share long articles. Articles having 2000 to 4000 words are most popular and shared.



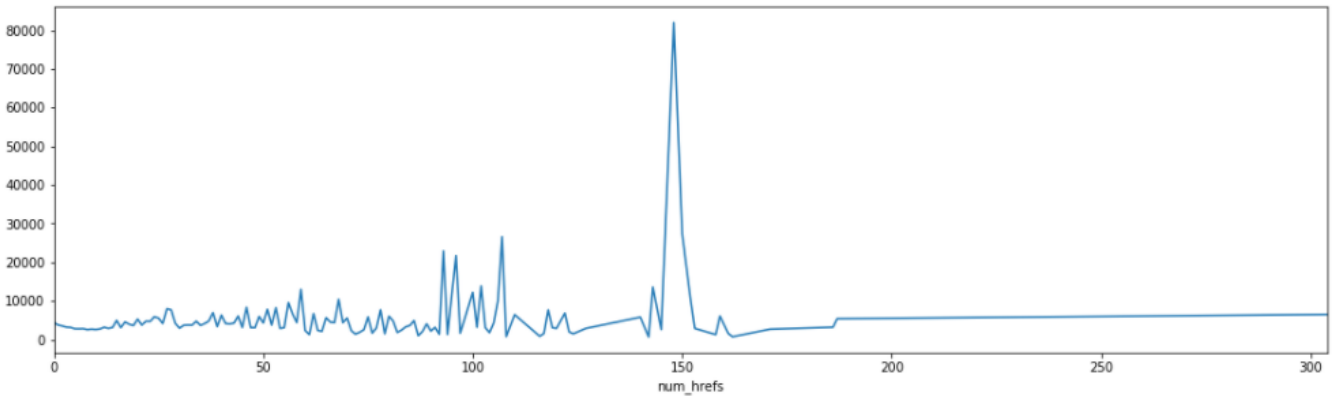
5.2.4 Number of videos vs. Shares

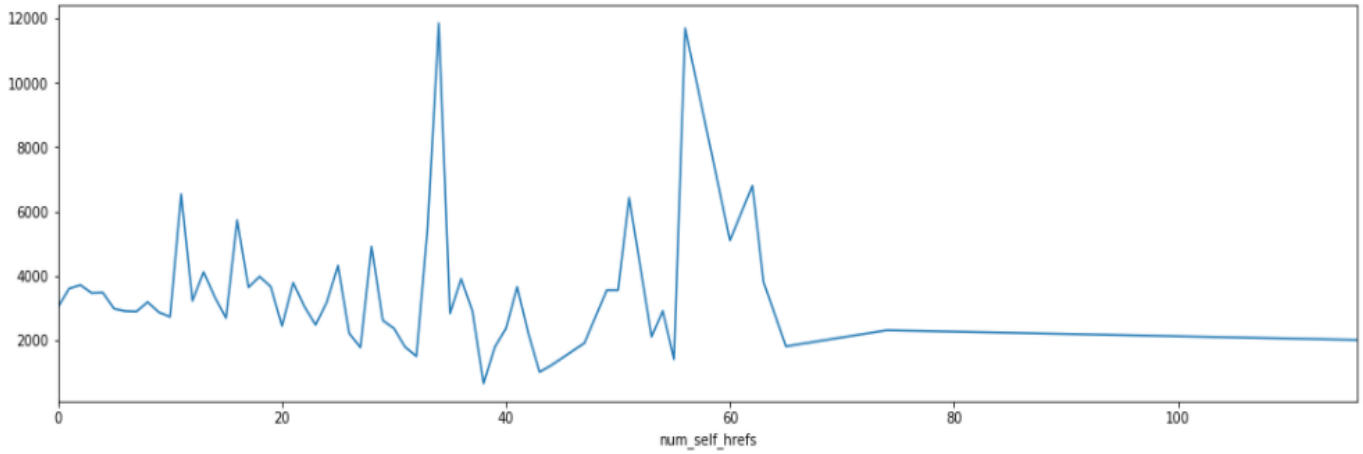
- We can conclude to a certain level of confidence that with increase in number of videos there is an increase in number of shares.



5.2.5 Number of Links vs. Shares & Number of Links to other Articles vs. Shares

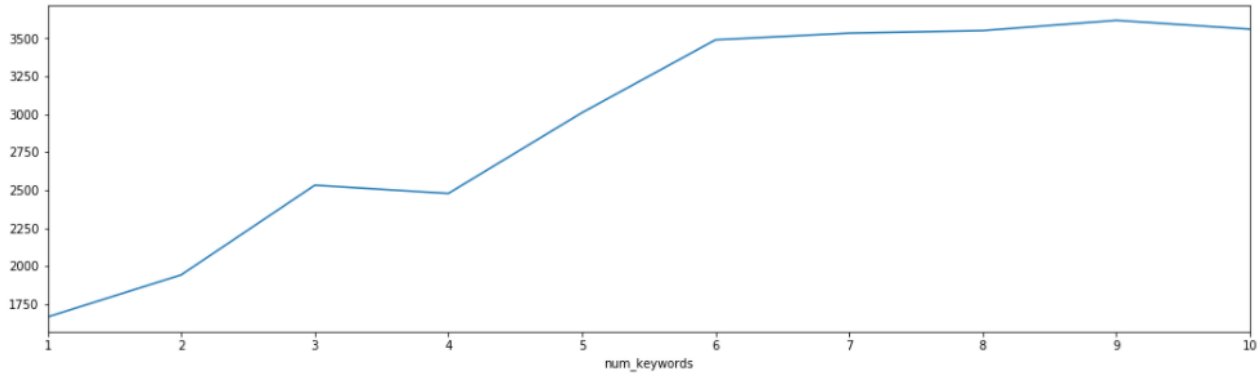
- People do like to share articles containing links. However, if an article contains a too many links it is generally not shared and the shares are generally lowest.
- So having minimum no of links is preferred.





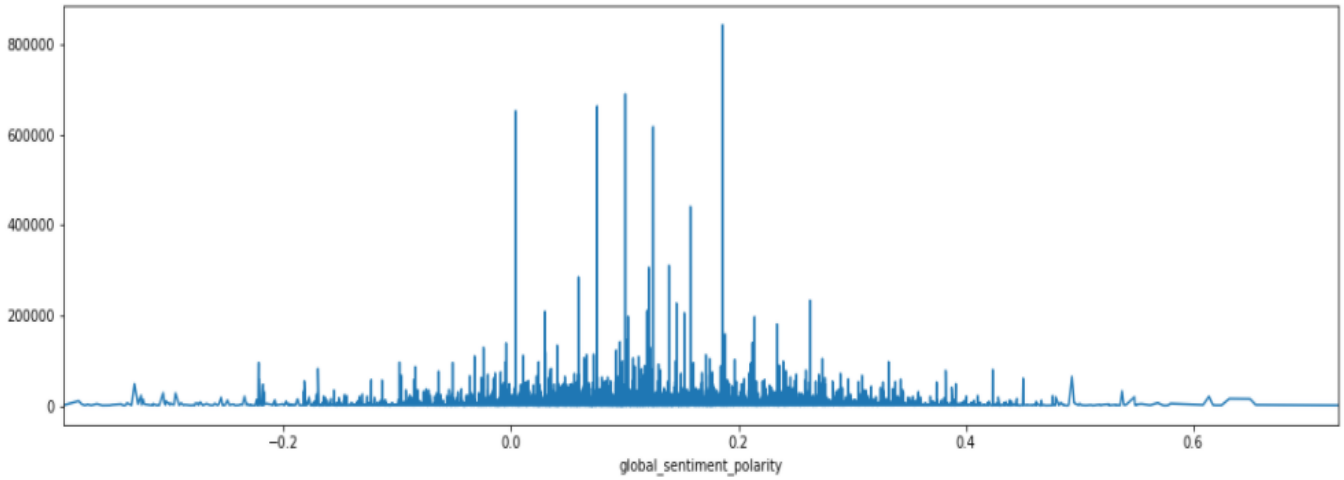
5.2.6 Number of keywords vs. Shares

- There's a noticeable trend in the line of `num_keywords` vs mean `shares`.
- More the keywords in the metadata of the article, more the ariticle is popular (shared more).
- One can also see that there is no considerable increase in mean shares after reaching 6 keywords position.



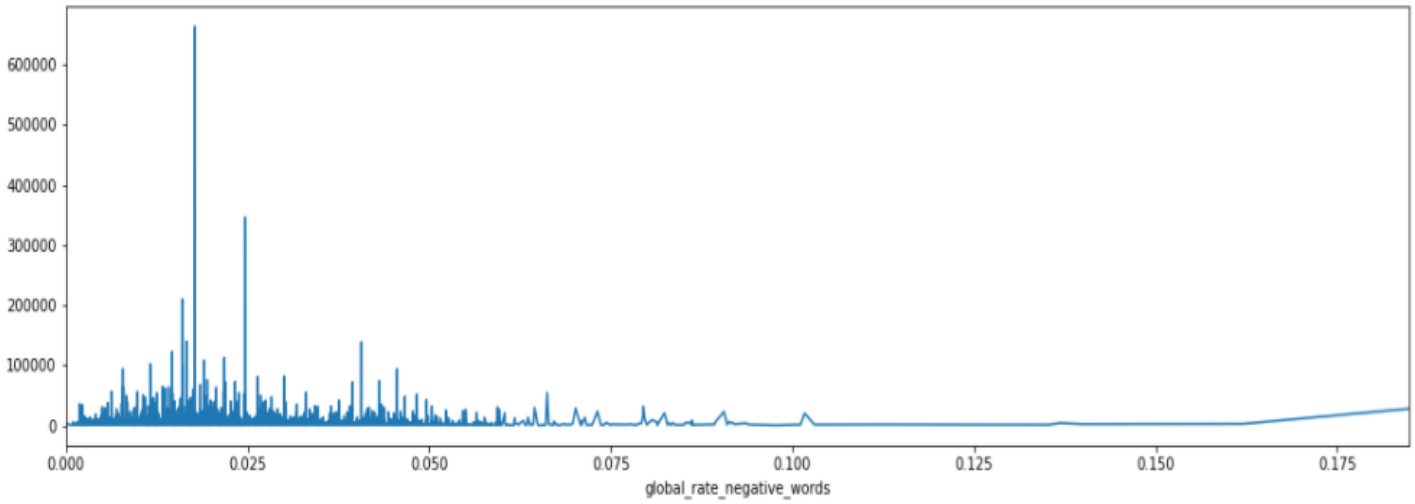
5.2.7 Global Sentiment Polarity vs. Shares

- The line graph of mean `shares` vs `global_sentiment_polarity` shows the sentiment of the whole article which gets popular.
- Positive sentiment range of above 0.0 sees most of the spikes in average `shares`.
- We can also see that spikes are not present after 0.4 in the positive sentiment range too.
- People like to share articles, which are generally positive or neutral.



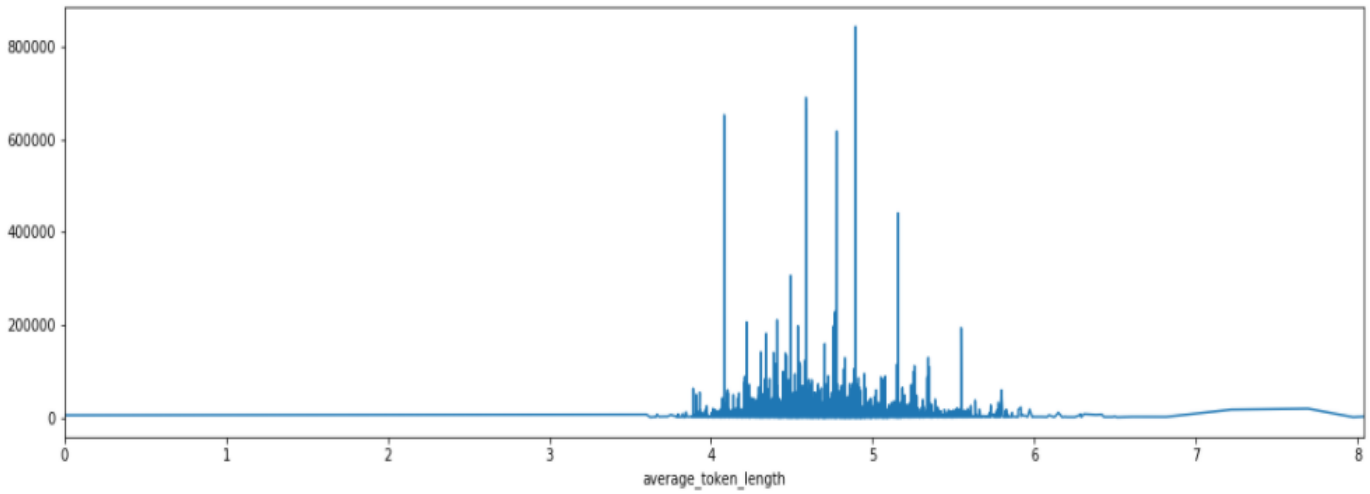
5.2.8 Global Rate of Negative Words vs. Shares

- Lower the occurrences of Negative words more the number of shares.



5.2.9 Average token Length vs. Shares

- The line graph above shows the mean ‘shares’ vs ‘average_token_length’ (average length of words in the article).
- We can clearly see that the mean ‘shares’s spikes more at the level of 4-6 length of words.
- Neither below 4 nor above 6 has any shift in the spike of mean `shares`.



6. OUTLIER TREATMENT

6.1 Detection

Checking the outlier percentage of all the columns in the dataset by the Inter Quartile Range method. This is done in python by creating a function to return the outlier percent and looping this function created over all the columns of the dataset.

In [7]:
```
#creating a function to check outlier percentage
def outlier_per(col):
    values=col
    q1=np.quantile(values,0.25)
    q3=np.quantile(values,0.75)
    iqr=q3-q1
    ul=q3+(1.5*iqr)
    ll=q1-(1.5*iqr)
    per=((len(values[values>ul])+len(values[values<ll]))/len(values))*100
    return(ul,ll,per)
```

6.2 Transformation / Removal

By using the Square Root transformation and Log(X+1) transformation, we can reduce the outliers to a certain level. We can also choose elimination of the outliers by dropping the respective rows from our dataset. Percentage of outliers after transformation/ removal has been checked for all the columns.

Out[10]:

	Column Name	Before Transformation	After Log Transformation	After Square Root Transformation	After removal
0	timedelta	0.000000	2.353505	0.000000	0
1	n_tokens_title	0.393512	0.572611	0.628106	0
2	n_tokens_content	4.873496	3.291880	2.010443	1.81929
3	n_unique_tokens	4.043589	3.864491	3.932598	0.260252
4	n_non_stop_words	7.128623	7.128623	7.128623	2.00179
5	n_non_stop_unique_tokens	4.368993	4.457281	4.502686	0.429954
6	num_hrefs	5.471332	0.335494	2.313145	2.81259
7	num_self_hrefs	5.269531	1.218374	2.282875	0
8	num_imgs	19.428398	5.950609	11.527886	43.2735
9	num_videos	7.431325	5.483944	4.815478	0
10	average_token_length	4.240345	4.076382	4.172237	0.239713
11	num_keywords	0.128648	1.843957	0.242161	0
12	kw_min_min	11.797795	0.000000	0.000000	0
13	kw_max_min	9.234922	10.009333	9.265192	3.25163
14	kw_avg_min	5.249350	6.291149	0.000000	1.009
15	kw_min_max	12.768963	0.000000	5.206468	9.14953

6.3 Treatment

Creating a function that checks which transformation gives the best effective reduction in outliers or if both doesn't have that much effect (at least 20% reduction) in reducing the outliers, the function checks whether removal of those rows can be effective and whether there is huge loss of data (>5%) if the rows are dropped. This also checks whether the column has any negative values which could result in imaginary / infinity values while applying any of the two transformations.

	Column Name	Transformation Done
0	timedelta	No need for transformation.
1	n_tokens_title	Removal
2	n_tokens_content	Square Root Transformation
3	n_unique_tokens	Removal
4	n_non_stop_words	Removal
5	n_non_stop_unique_tokens	Removal
6	num_hrefs	Log (X+1) transformation
7	num_self_hrefs	Log (X+1) transformation
8	num_imgs	Log (X+1) transformation
9	num_videos	Square Root Transformation
10	average_token_length	Removal
11	num_keywords	Removal
12	kw_min_min	NO transformation/removal is effective.
13	kw_max_min	NO transformation/removal is effective.
14	kw_avg_min	NO transformation/removal is effective.
15	kw_min_max	Log (X+1) transformation

7. STATISTICAL SIGNIFICANCE OF VARIABLES

7.1 Normality test of the target

We performed “kstest” test to check the normality of Target column- Shares. Since the Distribution of the target column is not normal, we need to perform only non parametric tests.

```
stat,pval=kstest(df['shares'],'norm')
print('Stat Value:',stat,'P Value:',pval)
if pval>0.05:
   print('Normal Distribution (Gaussian)')
else:
   print('There is no Normal Distribution')
```

```
Stat Value: 0.9997662334628614 P Value: 0.0
There is no Normal Distribution
```

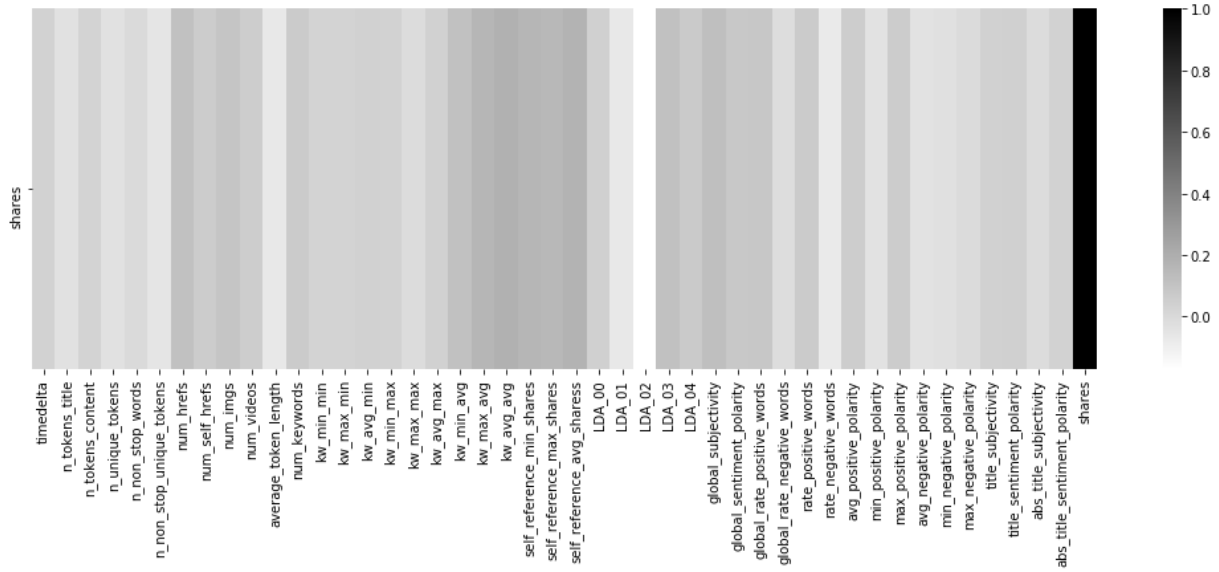
7.2 Kruskal-Wallis H-test tests

The Kruskal-Wallis H-test tests the null hypothesis that the population median of all of the groups are equal. It is a non-parametric version of ANOVA.

	Column	Stat Value	P Value	Significance
0	data_channel_is_entertainment	384.445796	1.339850e-85	Significant
1	data_channel_is_lifestyle	59.399517	1.287007e-14	Significant
2	data_channel_is_bus	1.164061	2.806250e-01	Not Significant
3	data_channel_is_socmed	480.727108	1.484318e-106	Significant
4	data_channel_is_tech	416.826891	1.197063e-92	Significant
5	data_channel_is_world	958.156708	2.237093e-210	Significant
6	weekday_is_monday	10.455867	1.222605e-03	Significant
7	weekday_is_tuesday	54.041685	1.962810e-13	Significant
8	weekday_is_wednesday	60.081599	9.100518e-15	Significant
9	weekday_is_thursday	39.403854	3.446161e-10	Significant
10	weekday_is_friday	3.080485	7.923692e-02	Not Significant
11	weekday_is_saturday	360.427797	2.272126e-80	Significant
12	weekday_is_sunday	302.104892	1.146019e-67	Significant
13	is_weekend	709.523112	2.540158e-156	Significant

7.3 Statistical Test for Continuous Independent Variables and the Target Variable

Correlation was calculated between Shares and Independent features.



Top 10 positively correlated features:	
shares	1.000000
kw_avg_avg	0.187171
self_reference_avg_sharess	0.180348
self_reference_min_shares	0.167511
kw_max_avg	0.163650
self_reference_max_shares	0.158326
global_subjectivity	0.126159
kw_min_avg	0.116028
num_hrefs	0.115922
LDA_03	0.114358

Top 10 negatively correlated features:	
LDA_02	-0.170575
rate_negative_words	-0.070795
average_token_length	-0.064787
LDA_01	-0.063297
n_non_stop_unique_tokens	-0.051209
avg_negative_polarity	-0.037260
min_positive_polarity	-0.035873
n_tokens_title	-0.031780
n_unique_tokens	-0.031134
min_negative_polarity	-0.025590

8. MODEL BUILDING

The data with no null values and insights from EDA was then split into Train and Test set as 30% of total dataset for Test Dataset. We applied OLS (Ordinary Least Squares) model initially to understand the performance of the base model. Assumptions of Linear regression were checked.Various other regressor models like Random Forest Regressor, Decision tree Regressor and other ensemble model regressors were used.

8.1 Linear Regression Model

We got R square value of 0.142 for the base OLS model and MSE value of 0.7 and 0.6 for test and train data respectively

```
R^2 score of the model is: 0.14134748688071797
MSE of the model is: 0.7094276182838636
RMSE of the model is: 0.8422752627756935
```

8.2 Assumptions of Linear Regression

8.2.1 *Linearity :*

- To check whether the relationship between the input and output variable is linear
- In our dataset residuals follow linearity

8.2.2 *Normality* :

- Normality tests are used to determine if a data set is well-modeled by a normal distribution and to compute how likely it is for a random variable underlying the data set to be normally distributed.
- Normality between residuals is not established.

8.2.3 *Homoscedasticity :*

- This assumption requires that the variance of the errors be constant for all values of X. In other words, the variability of Y values is the same when X is a low value as when X is a high value.
- Plot of residuals indicating that a fitted model is Appropriate and data is homoscedastic.

8.2.4 *No Autocorrelation*

- Autocorrelation refers to the degree of correlation between the values of the

same variables across different observations in the data.

- We can observe Positive Correlation.

8.2.5 No Multicollinearity :

- In statistics, multicollinearity (also collinearity) is a phenomenon in which one predictor variable in a multiple regression model can be linearly predicted from the others with a substantial degree of accuracy.
- We observed high multi-collinearity in the features.

8.3 Other Proposed Models

Now that we have our dataset prepared and features treated for outliers and false values, we will do some training/cross-validation to get a baseline understanding of how our models perform and which model perform well on the data. The models to be used are:

- DecisionTreeRegressor
- RandomForestRegressor
- KNeighborsRegressor
- AdaBoostRegressor
- BaggingRegressor
- GradientBoostingRegressor
- XGBRegressor
- LGBMRegressor
- SVR

8.3.1 DecisionTreeRegressor:

Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

8.3.2 RandomForestRegressor

Random forests are a way of averaging multiple deep decision trees, trained on different parts of the same training set, with the goal of reducing the variance.

8.3.3 KNeighborsRegressor

The k-nearest neighbors (KNN) algorithm is a simple, supervised machine learning algorithm that can be used to solve both classification and regression problems. It's easy to implement and understand, but has a major drawback of becoming significantly slows as the size of that data in use grows.

8.3.4 *AdaBoostRegressor:*

AdaBoost regressor is a meta-estimator that begins by fitting a regressor on the original dataset and then fits additional copies of the regressor on the same dataset but where the weights of instances are adjusted according to the error of the current prediction. As such, subsequent regressors focus more on difficult cases.

8.3.5 *BaggingRegressor:*

Bagging regressor is an ensemble meta-estimator that fits base regressors each on random subsets of the original dataset and then aggregate their individual predictions (either by voting or by averaging) to form a final prediction. Such a meta-estimator can typically be used as a way to reduce the variance of a black-box estimator (e.g., a decision tree), by introducing randomization into its construction procedure and then making an ensemble out of it.

8.3.6 *GradientBoostingRegressor*

Gradient boosting classifiers are a group of machine learning algorithms that combine many weak learning models together to create a strong predictive model. Decision trees are usually used when doing gradient boosting.

8.3.7 *XGBRegressor:*

XGBoost stands for "Extreme Gradient Boosting" and it is an implementation of gradient boosting trees algorithm. The name xgboost, though, actually refers to the engineering goal to push the limit of computations resources for boosted tree algorithms. Which is the reason why many people use xgboost.

8.3.8 *LGBMRegressor*

LightGBM is a gradient boosting framework that uses tree-based learning algorithms. It is designed to be distributed and efficient with Faster training speed and higher efficiency, Lower memory usage, Better accuracy, Support of parallel and GPU learning and Capable of handling large-scale data.

8.3.9 *SVR:*

A support-vector machine constructs a hyperplane or set of hyperplanes in a high- or infinite-dimensional space, which can be used for classification, regression, or other tasks like outliers' detection.
Our objective, when we are moving on with SVR, is to basically consider the points that are within the decision boundary line. Our best fit line is the hyperplane that has a maximum number of points.

9. MODEL EVALUATION

We are going to evaluate the models based on Mean Squared Error.

The mean squared error (MSE) of an estimator measures the average of the squares of the errors—that is, the average squared difference between the estimated values and the actual value.

MSE is a risk function, corresponding to the expected value of the squared error loss. The fact that MSE is almost always strictly positive (and not zero) is because of randomness or because the estimator does not account for information that could produce a more accurate estimate.

The mathematical equation that will give us the mean squared error is,

$$\text{MSE}=\frac{1}{n}\sum_{1}^{n}(y-y')^2$$

Where:
n-no. of observations
y-actual value of y
y′-predicted value of

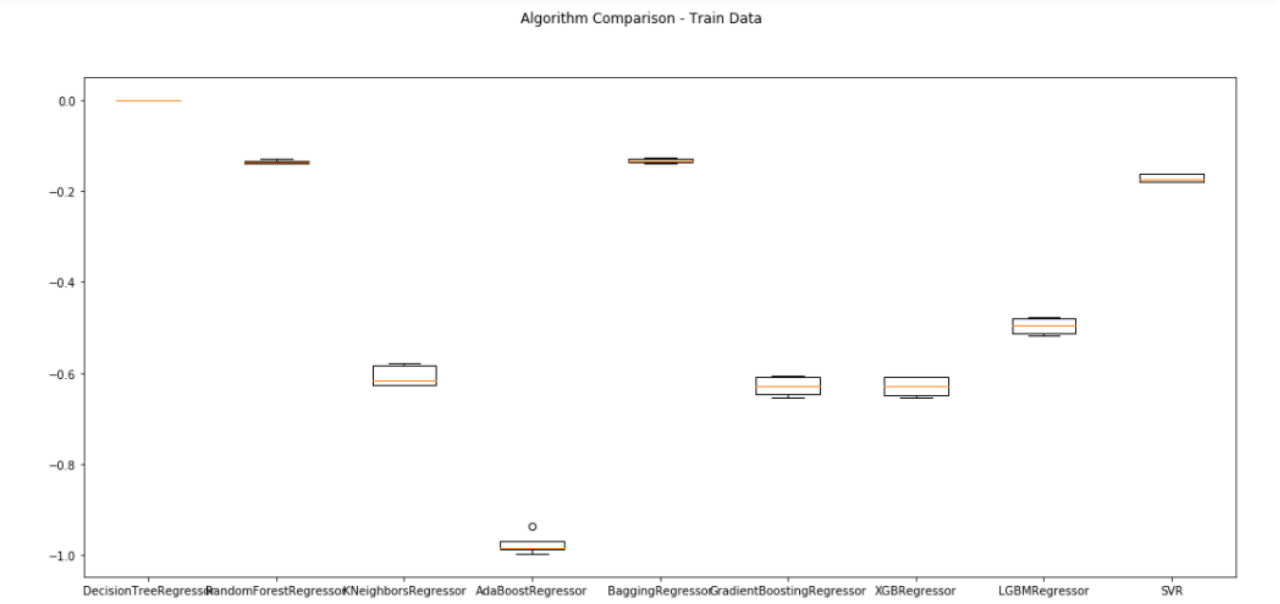
Negative Mean Square Error:

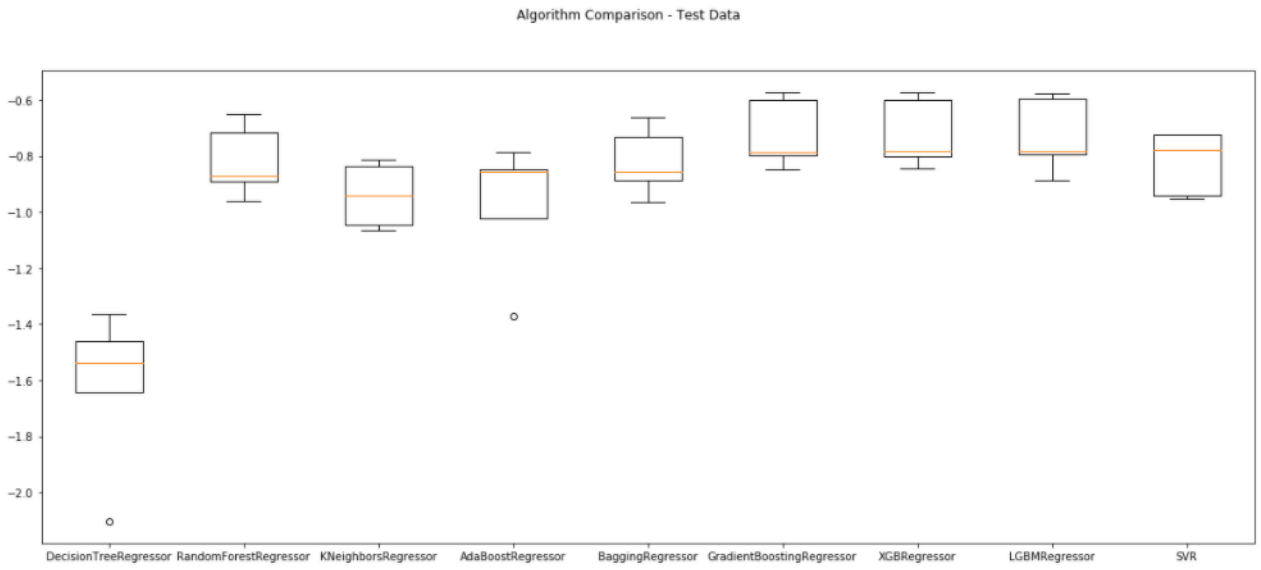
The Mean Square Error returned by sklearn.cross_validation.cross_val_score is always a negative. The actual MSE is the absolute value of the Negative MSE.

Below listed are the values of the Average MSE of the various models.

	Train_mean_score	Test_mean_score
DecisionTreeRegressor	-1.993758e-33	-1.620499
RandomForestRegressor	-1.356091e-01	-0.817309
KNeighborsRegressor	-6.058626e-01	-0.940220
AdaBoostRegressor	-9.742227e-01	-0.977136
BaggingRegressor	-1.333719e-01	-0.820163
GradientBoostingRegressor	-6.284071e-01	-0.720668
XGBRegressor	-6.296427e-01	-0.719909
LGBMRegressor	-4.967486e-01	-0.726665
SVR	-1.714207e-01	-0.824166

Below are the boxplots which compare the various algorithms.





9.1 Final Model Selection

By comparing the evaluation metrics of all the modes in both Train and test set, we identified that XGBoost model was better than others as it gave the least MSE value.

Here the MSE for both train and test are better than any other model and also the difference between train and test data accuracies are very low, which shows that the model is neither an overfit nor an underfit. Comparing to the evaluation metric MSE of XGBoost without tuning, it is proved that the model works well with data.

9.1.1 XGB Regressor

XGBoost is an implementation of gradient boosted decision trees designed for speed and performance. XGBoost stands for eXtreme Gradient Boosting.

The XGBoost library implements the gradient boosting decision tree algorithm.

Boosting is an ensemble technique where new models are added to correct the errors made by existing models. Models are added sequentially until no further improvements can be made.

PROS AND CONS OF XGBRegressor

Pros:

The two reasons to use XGBoost are also the two goals of the project:

- Execution Speed: Generally, XGBoost is fast. Really fast when compared to other implementations of gradient boosting.
- Model Performance: XGBoost dominates structured or tabular datasets on classification and regression predictive modeling problems.

Cons:

- Difficult interpretation, visualization tough
- Overfitting possible if parameters not tuned properly.
- Harder to tune as there are too many hyperparameters.

In our model, without any hyper parameter tuning we were able to get a good MSE value, in both Training and Test dataset.

```
XGBRegressor Train MSE : 0.629643
XGBRegressor Test MSE : 0.719909
```

9.1.2 Hyper Parameter Tuning

The hyper parameters of XGBRegressor were tuned using RandomisedSearchCV to get the optimum results for the data set.

```
params = { 'n_estimators': st.randint(0,200),
        'max_depth': st.randint(0,50),
        'learning_rate': st.uniform(0,1),
        'gamma':st.uniform(0,1),
        'reg_alpha':st.uniform(0,1),
         'reg_lambda':st.uniform(0,1),
        'colsample_bytree':st.uniform(0,1) }
xg_boost=XGBRegressor()
rsearch=RandomizedSearchCV(xg_boost,params,cv=7,random_state=42,n_jobs=1,
                  scoring='neg_mean_squared_error',n_iter=100)
rsearch.fit(X,y)
rsearch.best_params_
```

The results of the hyper parameters that are tuned are listed below.

```
colsample_bytree= 0.4148195023376652,
gamma= 0.27340707193070624,
learning_rate= 0.056375496650927115,
max_depth= 5,
n_estimators= 196,
reg_alpha= 0.9997176732861306,
reg_lambda= 0.9966368370739054
```

9.1.3 Optimised Results

The following is the result of the evaluation metrics of the model after the hyper parameter tuning.

```
MSE Of train: 0.5280755330949964
MSE Of test: 0.6672382031476866
```

We were able to reduce the MSE after the parameter tuning.

10. OUTCOMES

10.1 Recommendations

Key Preferences:

- Increase in the Number of Keywords.
- Words in the title between should be crisp (not too short and not too long).
- Increase in the Number of Images.
- Words in the content better kept minimal.
- Restriction in the number of links to other articles up to 150.
- Global Sentiment somewhere between neutral to positive polarity.
- Lower the occurrence of negative words in the article content and title.
- Length of the words in the content between 4-6 letters on average.
- Articles Published in the weekend are shared more.
- **Data Channel:**
    - Editors may want to consider the importance of channels when it comes to popularity and required emphasis must be given.
    - Lifestyle > Social Media > Technology > Business > Entertainment > World.
- **Time of Publication:**
    - Articles are published in certain days are shared more than others.
    - Saturday > Sunday > Monday > Friday > Tuesday > Thursday > Wednesday.
    - We can see that the weekend published articles are usually shared more.

10.2 Limitations

- Our model focuses on the content part of the article more and therefore misses out on the other factors such as reader's attributes, time relevance of the article's subject which could also heavily influence the shares of the articles.
- Reader Demographics such as Age, Location could vary with respect to the kind of articles (content, channel) they are sharing more which we lack in our dataset and therefore model.
- The model is quite low in the precision point of view in predicting the number of shares and subsequently it tends to the increase in the error in the predicted values.

10.3 Improvements

Various other variables on the reader's front such as reader demographics, socio-economic features such as Age, Income, Location, Career and time period relevance of the article's news can be an influencing factor in the articles which he chooses to share more. These variables help us in better reader/customer segmentation and further help us in understanding and predicting the shares better.

11. CONCLUSIONS

- We can see that the feature variables have very low correlation to our target variable.Infact the highest correlated value is only 0.11.

- Among the various models that were applied on the dataset, XG Boost regressor gave us the best MSE value of 0.667.

- We can observe that the MSE value for any of the model applied lies within the range of 0.65 to 0.8. MSE can be low in situation of under fitting, where there are too many degrees of freedom available to Residual space and too few to the Regressor space.

- The best R-squared value obtained is never above 20%.This value is quite low. A low R-squared value indicates that our independent variables is not explaining much in the variation of your dependent variable - regardless of the variable significance.

- We can see that none of the features greatly impact the target variable.The prediction can be improved by taking into account other features which aren't provided in the dataset.

12. REFERENCES

[1] Fernandes, K., Vinagre, P., & Cortez, P. (2015, September). A proactive intelligent decision support system for predicting the popularity of online news. In *Portuguese Conference on Artificial Intelligence* (pp. 535-546). Springer, Cham.

[2] Dumont, F., Macdonald, E. B., & Rincon, A. F. (2016). Predicting Mashable and Reddit Popularity Using Closed-form and Gradient Descent Linear Regression: COMP598.

[3] Uddin, M. T., Patwary, M. J. A., Ahsan, T., & Alam, M. S. (2016, October). Predicting the popularity of online news from content metadata. In 2016 International Conference on Innovations in Science, Engineering and Technology (ICISET) (pp. 1-5). IEEE.

[4] Lei, X., Hu, X., & Fang, H. Is Your Story Going to Spread Like a Virus? Machine Learning Methods for News Popularity Prediction.

[5] Hensinger, E., Flaounas, I., & Cristianini, N. (2013). Modelling and predicting news popularity. *Pattern Analysis and Applications*, *16*(4), 623-635.

[6] Shreyas, R., Akshata, D. M., Mahanand, B. S., Shagun, B., & Abhishek, C. M. (2016, August). Predicting popularity of online articles using random forest regression. In *2016 Second International Conference on Cognitive Computing and Information Processing (CCIP)* (pp. 1-5). IEEE.