Online News Popularity Analysis

Yulin Hou

hou.yul@northeastern.edu

Percentage of Effort Contributed by Student:____100%_____

Signature of Student:_ _____Yulin Hou_____

Submission Date:_____04/18/2023_____

· Introduction

1. The business problem I am going to work on in this course

The expansion of social media has had a significant impact on various industries, especially the news media industry, providing both opportunities and challenges for news organizations and individuals. This has led to a shift in the way that news is produced and consumed. Therefore, it has increased the competition for attention, making it more challenging for news organizations and individuals to stand out. Additionally, the rise of social media has also led to the spread of misinformation and fake news, which makes people harder to discriminate.

To sum up, based on the dataset “Online News Popularity Data Set”, the first problem I am going to work on is to predict the online news popularity with a large list of characteristics known before publication that describe different aspects of the article. Secondly, I propose to explore the association between the news popularity and the characteristics of the article, such as the data channel and weekday.

2. Explain a few sentences about the business problem

For the first business problem, predicting online news popularity with characteristics known before publication can be beneficial in different aspects, such as resource allocation and reference. Secondly, by exploring the factors of popular news, individuals and news organizations could obtain insight into what types of content are more likely to be popular with readers. This information could be used to guide the production and distribution of content, which can help them increase audience engagement and revenue.

3. Explain one or two sentences on the dataset

The dataset collects the content of all the articles published from 2013 to 2015 from Mashable, which is one of the largest news websites. There is a total of 39644 articles with 47 features and without missing value. The nominal attributes were transformed with the common 1-of C-encoding.

```
#example of nominal attributes(data channel)
raw_data.iloc[:,13:19].head()
```

	data_channel_is_lifestyle	data_channel_is_entertainment	data_channel_is_bus	data_channel_is_socmed	data_channel_is_tech	data_channel_is_world
0	0.0	1.0	0.0	0.0	0.0	0.0
1	0.0	0.0	1.0	0.0	0.0	0.0
2	0.0	0.0	1.0	0.0	0.0	0.0
3	0.0	1.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	1.0	0.0

· Problem Statement

1. My hypothesis

After training the machine learning model, we could classify the article into suitable popularity groups with the characteristics of the articles with an acceptable error. Besides, we will summarize some significant factors of the popular article, which has the top share number in different topics and in general, such as the number of words in the title, data channel(topic), and published weekday.

2. How will the machine approach or automation solve the business use case?

First, the training data would be split into training(60-70%) and validation(30-40%) sets. Then, to predict the popularity of online news, we will utilize some classification models, such as K-Nearest Neighbors (KNN) , and random forest, searching for the best performance model to predict the share of the articles.

Moreover, to find out the significant factors of popular news, we will try to apply some methods to get the news characteristics that are strongly correlated with popularity.

3. What does the business get out of this solution?

For the first problem, predicting online news popularity with characteristics known before publication can be beneficial in different aspects, such as:

1. Resource allocation: individuals and news organizations could allocate resources more effectively and prioritize the production and promotion of high-performing content.
2. Brand reputation: individuals and news organizations can avoid publishing uninteresting news as well as controversial or sensitive content that could damage their brand reputation.

For the second business problem, by finding the factors of popular news, individuals and news organizations could:

1. Tailor content to better meet the needs and interests of the audience.
2. Optimize distribution of the content to maximize reach and engagement.

· Dataset used

Data dictionary

The data dictionary includes the attribution information, datatype, the number of missing values in the column as well as an example for each attribution.

]:

Attribute	Attribute Information	Datatype	Missing_value	Example
0. url	URL of the article	object	0	http://mashable.com/2013/01/07/amazon-instant-...
1. timedelta	Days between the article ...	float64	0	731.0
2. n_tokens_title	Number of words in the title	float64	0	12.0
3. n_tokens_content	Number of words in the content	float64	0	219.0
4. n_unique_tokens	Rate of unique words in the con...	float64	0	0.663594
...	...	...	...	...
56. title_subjectivity	Title subjectivity	float64	0	0.5
57. title_sentiment_polarity	Title polarity	float64	0	-0.1875
58. abs_title_subjectivity	Absolute subjectivity level	float64	0	0.0
59. abs_title_sentiment_polarity	Absolute polarity level	float64	0	0.1875
60. shares	Number of shares (target)	int64	0	593

61 rows × 4 columns

According to the dictionary, we could get detailed information about the attributes.

· Analysis

1. Descriptive statistics

	timedelta	n_tokens_title	n_tokens_content	n_unique_tokens	n_non_stop_words	n_non_stop_unique_tokens	num_hrefs	num_self_hrefs	num_im
count	39644.000000	39644.000000	39644.000000	39644.000000	39644.000000	39644.000000	39644.000000	39644.000000	39644.0000
mean	354.530471	10.398749	546.514731	0.548216	0.996469	0.689175	10.883690	3.293638	4.5441
std	214.163767	2.114037	471.107508	3.520708	5.231231	3.264816	11.332017	3.855141	8.3094
min	8.000000	2.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0000
25%	164.000000	9.000000	246.000000	0.470870	1.000000	0.625739	4.000000	1.000000	1.0000
50%	339.000000	10.000000	409.000000	0.539226	1.000000	0.690476	8.000000	3.000000	1.0000
75%	542.000000	12.000000	716.000000	0.608696	1.000000	0.754630	14.000000	4.000000	4.0000
max	731.000000	23.000000	8474.000000	701.000000	1042.000000	650.000000	304.000000	116.000000	128.0000

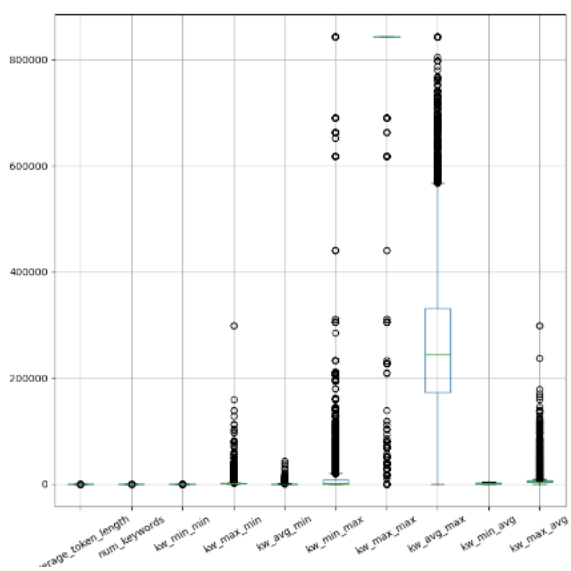
According to 'timedelta', we could see the time interval is great. We will select the online news published in more than 30 days and less than 90 days to solve our problem. Within this interval, we could ensure the validation of our data by excluding the impact of duration.

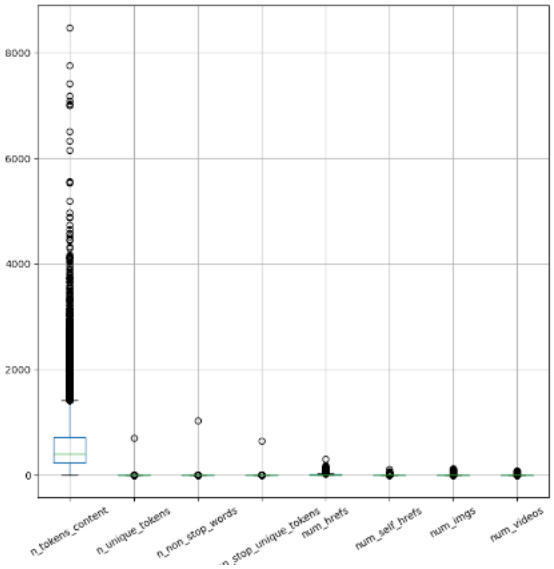
2. EDA

Outlier detection

Boxplot helps us to understand the numeric data distribution as well as find the outliers.

Taking the boxplots below as an example of outlier detection, we could know that some attributes have a great many outliers and the difference among the scales of attributes are significant, leading to the unintuitive plot. We can remove some outliers far away from the max value and scale the attributes.



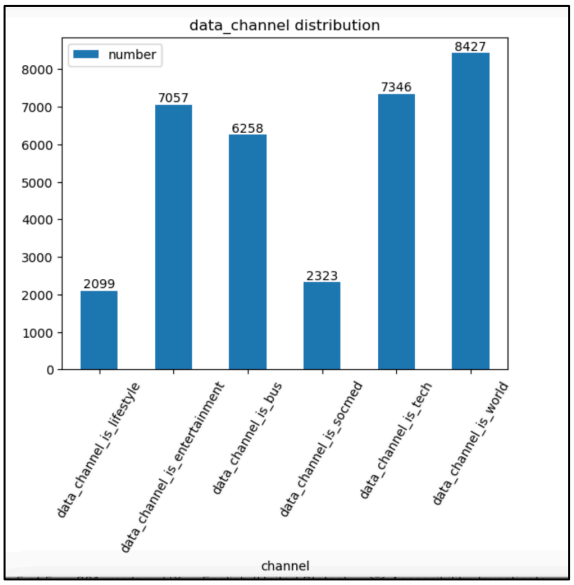


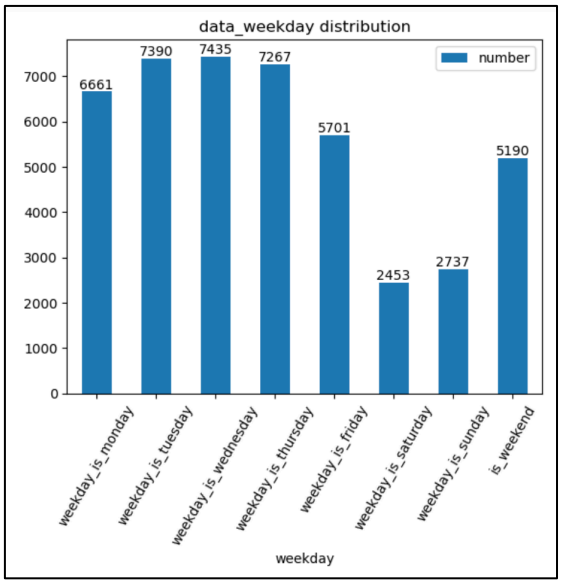
Univariate Analysis

In the OnlineNewspopularity dataset, there is some categorical information, such as the data channel and the published weekday. We could get some simple insights from category distribution.

From the left graph, we know that the world and tech channels have the most number of articles, so they might achieve great popularity among online news. Otherwise, there are fewer articles on lifestyle and social media channels, indicating less popularity. We will combine lifestyle and social media channels in feature engineering.

From the right graph, the attributes 'weekday_is_saturday', 'weekday_is_sunday', and 'is_weekend' are redundant, we will only keep 'is_weekend' to represent the other two as the weekend have a similar distribution. The dimension reduction operation will execute in the feature engineering.

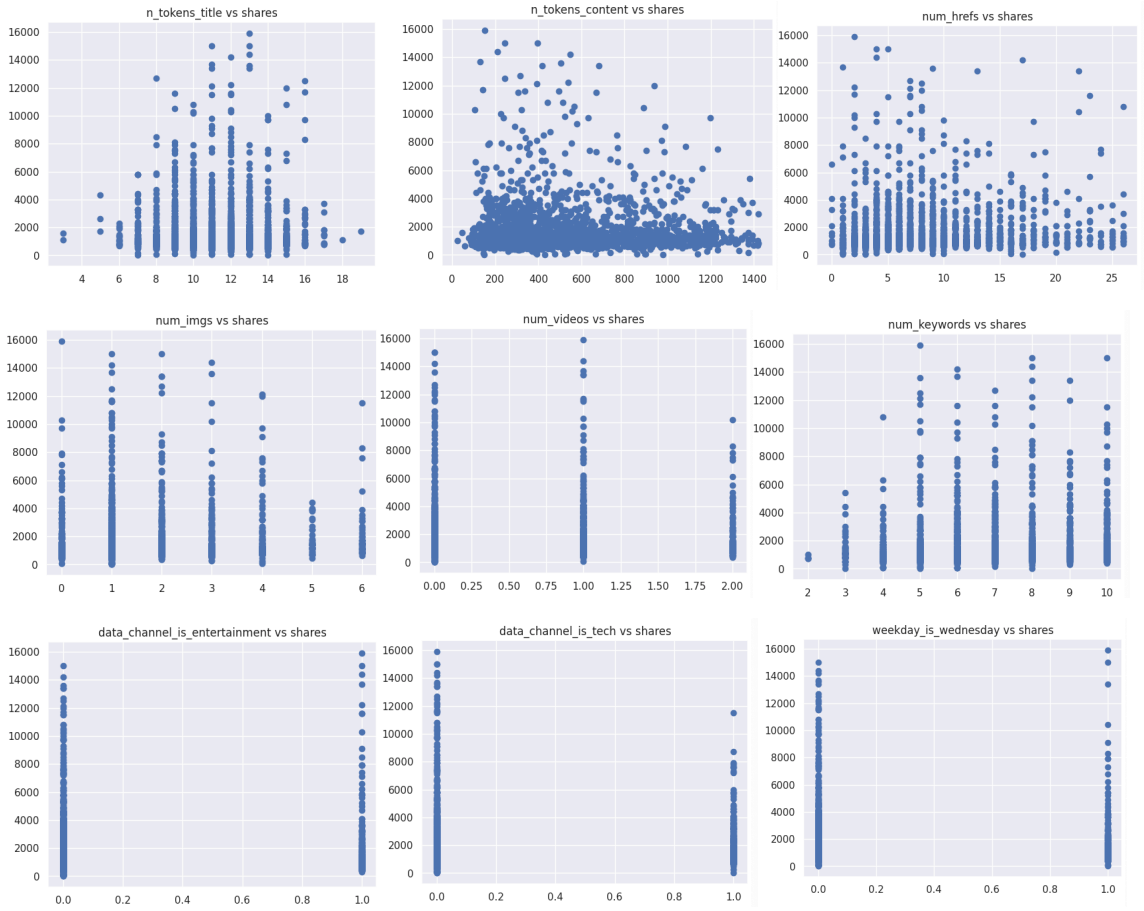


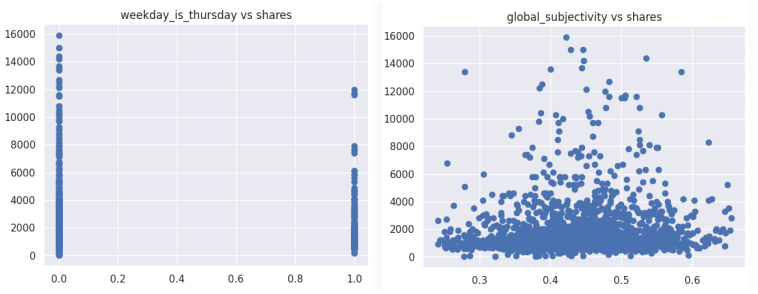


Bivariate Analysis

After removing outliers and selecting the time interval, we perform the bivariate analysis to explore the relation between features and the number of shares. According to the scatter plots below, we can obtain that:

- A moderate length of titles obtains more popularity, which is between 8 and 12 words. A too-long or too-short title led to uninterest.
- The number of words in news content lower than 700 is more attractive.
- News with 1-6 references, 0-3 images, and a video obtains more interest.
- Repeated keywords contribute to more popularity.
- Entertainment channel gains more popularity, but people have less interested in the tech channel.
- The news published on Wednesday obtains more popularity, but the news published on Thursday draws less attention.
- Neutral subjectivity tends to obtain more popularity.

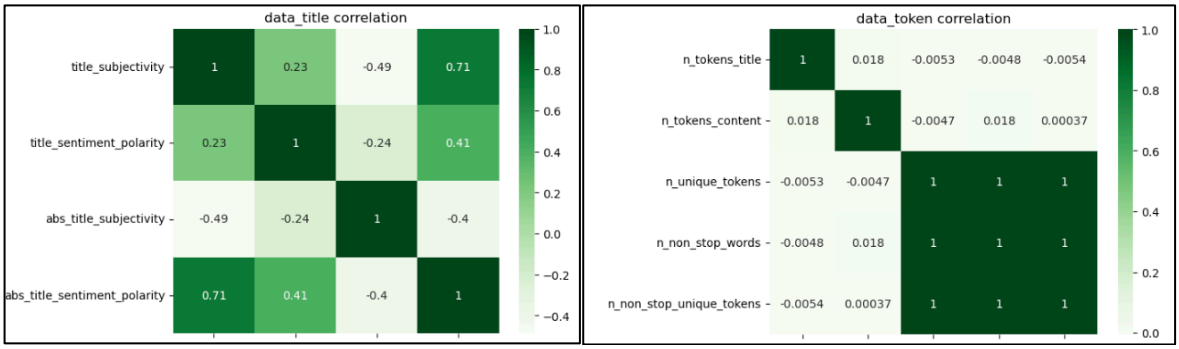


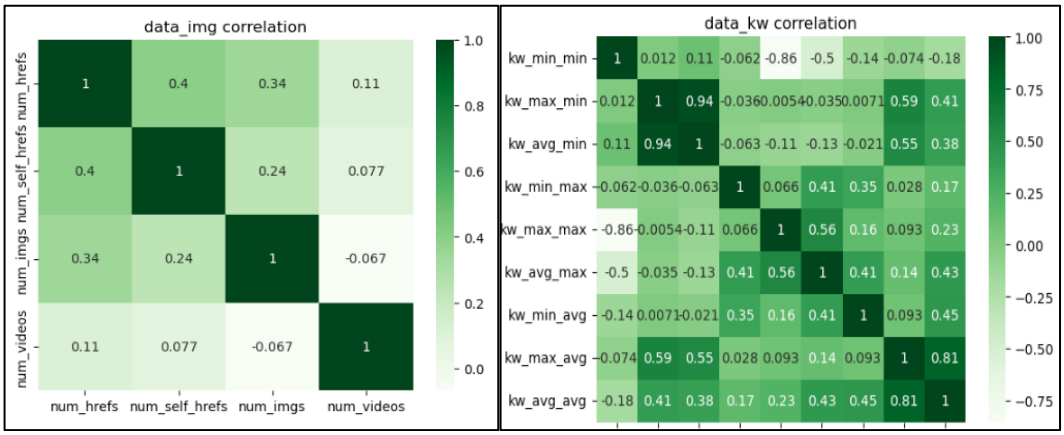


Correlation Analysis

As there are so many attributes, we will group similar attributes to compute the correlation.

From the heatmap below, we will notice that some attributes have high correlation with others. For example, in the ‘data_token correlation’ heatmap, the correlation among n_unique_tokens, 'n_non_stop_words', and 'n_non_stop_unique_tokens' is 1, indicating a perfect linear relationship among these attributes. In case of that, we can only keep one of the attributes which have high correlated variables.





PCA

Through PCA, we could extract the principle component from the original features. Before doing the Principal Components Analysis, We also utilize StandardScaler to standardize the dataset.

```
pca.explained_variance_ratio
[7.92452274e-02 7.20577403e-02 5.53316289e-02 5.25182421e-02
 4.93712913e-02 4.49425226e-02 4.39960492e-02 4.34894415e-02
 4.30050774e-02 4.22845462e-02 4.04254920e-02 3.97242506e-02
 3.90149537e-02 3.81728250e-02 3.77594467e-02 3.56993651e-02
 3.49006457e-02 3.31472183e-02 3.25648937e-02 2.99342513e-02
 2.72420969e-02 2.19005207e-02 2.01766097e-02 1.95146320e-02
 1.19989774e-02 7.39513882e-03 4.18691535e-03 1.56283655e-32]
cumsum(pca.explained_variance_ratio)
[0.07924523 0.15130297 0.2066346  0.25915284 0.30852413 0.35346665
 0.3974627  0.44095214 0.48395722 0.52624177 0.56666726 0.60639151
 0.64540646 0.68357929 0.72133873 0.7570381  0.79193875 0.82508596
 0.85765086 0.88758511 0.91482721 0.93672773 0.95690434 0.97641897
 0.98841795 0.99581308 1.         1.        ]
```

According to the explained variance ratio, we could cover 97.6% of information on the features with 24 principal components. However, because we still need the specific feature names for further analysis, we will keep all 28 selected attributes for future analysis.

Summary of Milestone2: Recommend features through EDA

After the EDA, we select 27 attributes for our problem, which covers significant information for each online news. Here is the latest dataset and new data dictionary.

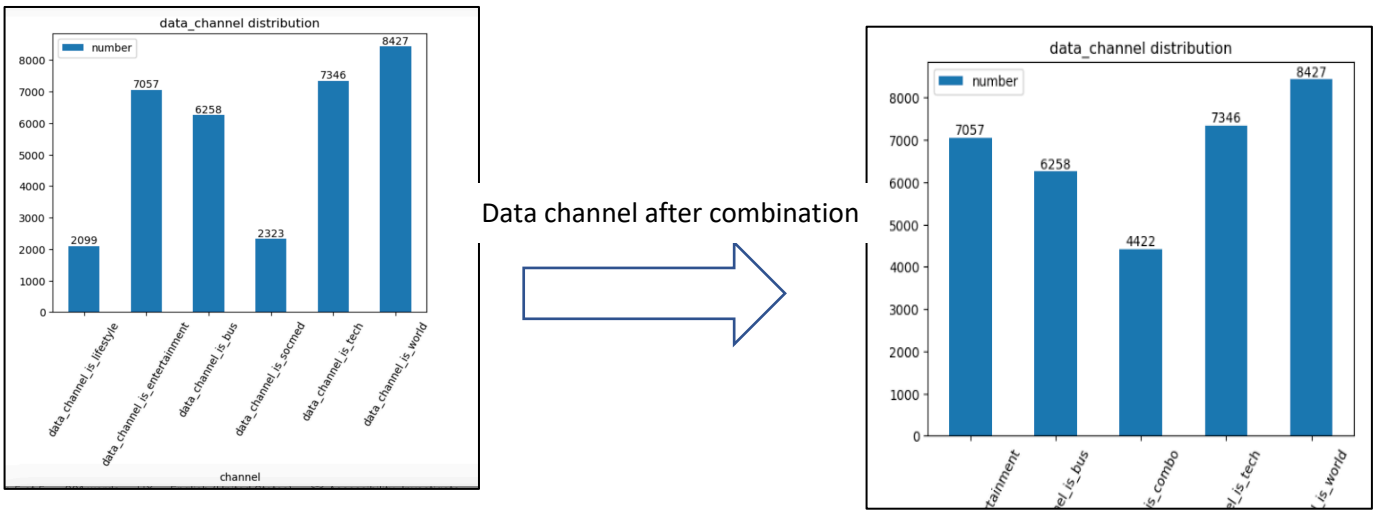
```
 0   timedelta                       39644 non-null  float64
 1   n_tokens_title                  39644 non-null  float64
 2   n_tokens_content                39644 non-null  float64
 3   n_unique_tokens                 39644 non-null  float64
 4   num_hrefs                       39644 non-null  float64
 5   num_imgs                        39644 non-null  float64
 6   num_videos                      39644 non-null  float64
 7   num_keywords                    39644 non-null  float64
 8   data_channel_is_lifestyle       39644 non-null  float64
 9   data_channel_is_entertainment   39644 non-null  float64
 10  data_channel_is_bus             39644 non-null  float64
 11  data_channel_is_socmed          39644 non-null  float64
 12  data_channel_is_tech            39644 non-null  float64
 13  data_channel_is_world           39644 non-null  float64
 14  kw_max_max                      39644 non-null  float64
 15  self_reference_avg_sharess      39644 non-null  float64
 16  weekday_is_monday               39644 non-null  float64
 17  weekday_is_tuesday              39644 non-null  float64
 18  weekday_is_wednesday            39644 non-null  float64
 19  weekday_is_thursday             39644 non-null  float64
 20  weekday_is_friday               39644 non-null  float64
 21  is_weekend                      39644 non-null  float64
 22  global_subjectivity             39644 non-null  float64
 23  rate_positive_words             39644 non-null  float64
 24  rate_negative_words             39644 non-null  float64
 25  abs_title_subjectivity          39644 non-null  float64
 26  abs_title_sentiment_polarity    39644 non-null  float64
 27  shares                          39644 non-null  int64
```

3. Feature Engineering

As the dataset doesn't have missing values and the categorical features have been encoded in the dataset, we can skip the encoding and imputation in the feature engineering.

(1) Creating features: combine data_channel: lifestyle and socmed (social media)

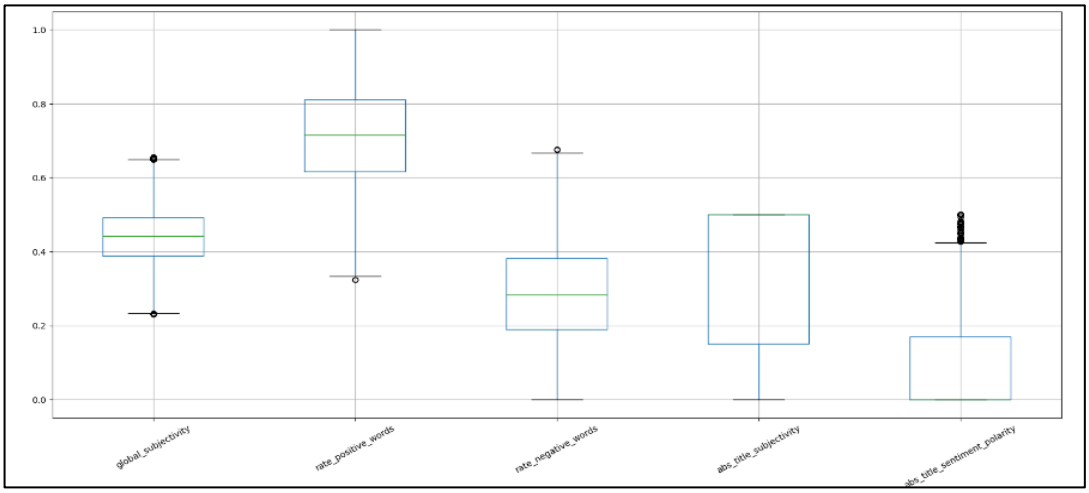
According to the data channel distribution analysis above, the number of articles on lifestyle and social media channels is much lower than the others. We will combine lifestyle and social media channels. Here is the new distribution of data_channel.



(2) Handle outliers

According to the boxplot above, there are a lot of outliers, which would impact our modeling accuracy. Therefore, we remove the outliers based on the result of the boxplot, setting the minimum= Q1-1.5*IQR, and maximum= Q3+1.5*IQR. After outliers removal, we get the new dataset with fewer outliers.

For the number of shares, we set Q1= shares.quantile(0.1), Q3=shares.quantile(0.9) and filter by maximum= Q3+3*IQR



(3) Scaling

'time delta': there is a long time span in 'time delta'. We select online news published more than 60 days and less than 550 days to relieve the impact on shares with a long duration.

	timedelta
count	20990.000000
mean	367.705669
std	220.671033
min	8.000000
25%	168.000000
50%	359.000000
75%	570.000000
max	731.000000

Time delta after scaling

	timedelta
count	13451.000000
mean	289.150546
std	146.813094
min	60.000000
25%	161.000000
50%	274.000000
75%	421.000000
max	550.000000

4. Feature Selection

In this section, we use multicollinearity, backward feature selection, forward feature selection, and lasso regression model to select proper features for our linear regression model. So far, there are 26 features as our predictors.

Multicollinearity

For multicollinearity, we use VIF to detect it. Generally, a VIF above 5 indicates high multicollinearity. There are only some data channel relative features having a VIF above 5, which means multicollinearity exists among the data channel and other features. We can remove some of them in the further selection as needed.

9	data_channel_is_bus	5.656902
10	data_channel_is_tech	5.822947
11	data_channel_is_combo	3.453411
12	data_channel_is_world	7.599553

Backward feature selection

After backward feature selection, we have 11 features for modeling training.

```
['n_tokens_title', 'num_hrefs', 'num_videos', 'data_channel_is_entertainment', 'data_channel_is_bus', 'data_channel_is_tech', 'data_channel_is_combo', 'data_channel_is_world', 'self_reference_avg_sharess', 'weekday_is_tuesday', 'global_subjectivity']
```

Forward feature selection

After forward feature selection, we have 11 features for modeling training.

```
['self_reference_avg_sharess', 'num_videos', 'global_subjectivity', 'data_channel_is_entertainment', 'data_channel_is_world', 'data_channel_is_tech', 'data_channel_is_bus', 'data_channel_is_combo', 'n_tokens_title', 'num_hrefs', 'is_weekend']
```

Lasso regression models

After the lasso regression model, we have 12 features for modeling training.

```
['n_tokens_title',
 'num_hrefs',
 'num_videos',
 'num_keywords',
 'data_channel_is_entertainment',
 'data_channel_is_combo',
 'data_channel_is_world',
 'kw_avg_max',
 'self_reference_avg_sharess',
 'is_weekend',
 'global_subjectivity',
 'abs_title_sentiment_polarity']
```

Summary of Milestone 3: Recommend the list of features.

To sum up, considering the result of backward elimination, forward selection, lasso regression model, and the feature engineering above. Finally, we determine the predictors with 22 features for the regression model. There are 11 numeric features and 11 encoding categorical features including the data channel and weekday information. We will use these 22 features to create a linear regression model to predict the number of shares.

```
 0   timedelta                      13451 non-null  float64
 1   n_tokens_title                 13451 non-null  float64
 2   n_unique_tokens                13451 non-null  float64
 3   num_hrefs                      13451 non-null  float64
 4   num_videos                     13451 non-null  float64
 5   data_channel_is_entertainment  13451 non-null  float64
 6   data_channel_is_bus            13451 non-null  float64
 7   data_channel_is_tech           13451 non-null  float64
 8   data_channel_is_combo          13451 non-null  float64
 9   data_channel_is_world          13451 non-null  float64
 10  kw_avg_max                     13451 non-null  float64
 11  self_reference_avg_sharess     13451 non-null  float64
 12  weekday_is_monday              13451 non-null  float64
 13  weekday_is_tuesday             13451 non-null  float64
 14  weekday_is_wednesday           13451 non-null  float64
 15  weekday_is_thursday            13451 non-null  float64
 16  weekday_is_friday              13451 non-null  float64
 17  is_weekend                     13451 non-null  float64
 18  global_subjectivity            13451 non-null  float64
 19  rate_positive_words            13451 non-null  float64
 20  rate_negative_words            13451 non-null  float64
 21  abs_title_sentiment_polarity   13451 non-null  float64
 22  shares                         13451 non-null  int64
```

4. Modeling

4-1 Introduction

In this section, we focus on creating various machine-learning models to predict and classify the popularity of online news. First, we utilize clustering algorithms such as K-means and DBSCAN to produce data labels. Then, we train classification models, such as KNN, Random Forest, Decision Tree, AdaBoost, and XGBoost, to analyze the online news and determine its popularity.

4-2 Model Implementation

(1) Clustering

Modeling preparation

Before we employ the clustering model, we standardize and normalize the data. Besides, we leverage PCA on the standardized and normalized data. We select 18 standardized PCA components and 15 normalized PCA components, which cover 95% of the data information, for further clustering model training.

```
#Standardization
std = StandardScaler()
data_std=pd.DataFrame(std.fit_transform(data_m3),columns=all_columns)
data_std.head()
```

	timedelta	n_tokens_title	n_unique_tokens	num_hrefs	num_videos	da
0	1.717244	0.849040	2.248670	-1.339207	-0.687128	
1	1.717244	1.307385	2.337711	-0.202540	2.504066	
2	1.717244	-1.901028	1.334952	-1.339207	-0.687128	
3	1.717244	0.390696	-0.547951	0.365793	-0.687128	
4	1.717244	0.390696	-1.083852	0.744683	-0.687128	

5 rows × 23 columns

```
#Normalization
minmax=MinMaxScaler()
data_minmax=pd.DataFrame(minmax.fit_transform(data_m3),columns=all_columns)
data_minmax.tail()
```

	timedelta	n_tokens_title	n_unique_tokens	num_hrefs	num_videos	data
2136	0.0	0.6250	0.160280	0.346154	0.5	
2137	0.0	0.6250	0.164871	0.423077	0.5	
2138	0.0	0.5000	0.152990	0.461538	0.0	
2139	0.0	0.4375	0.167759	0.000000	0.0	
2140	0.0	0.5000	0.370329	0.307692	0.5	

5 rows × 23 columns

```
pca = PCA()
pca.fit(data_std)
print(pca.explained_variance_ratio_)
print(np.cumsum(pca.explained_variance_ratio_))
```

```
[1.14356146e-01 7.26098851e-02 6.37175514e-02 5.72417930e-02
 5.57551298e-02 5.46897081e-02 5.29447400e-02 5.26115359e-02
 5.07815653e-02 4.88852195e-02 4.78945406e-02 4.41748475e-02
 4.38937852e-02 4.06252377e-02 4.01131923e-02 3.62420470e-02
 3.57767173e-02 3.41633222e-02 2.80015102e-02 2.29160504e-02
 2.60547553e-03 2.21676775e-27 4.22302695e-33]
[0.11435615 0.18696603 0.25068358 0.30792538 0.36368051 0.41837021
 0.47131495 0.52392649 0.57470805 0.62359327 0.67148781 0.71566266
 0.75955645 0.80018169 0.84029488 0.87653692 0.91231364 0.94647696
 0.97447847 0.99739452 1.         1.         1.        ]
```

```
pca = PCA()
pca.fit(data_minmax)
print(pca.explained_variance_ratio_)
print(np.cumsum(pca.explained_variance_ratio_))
```

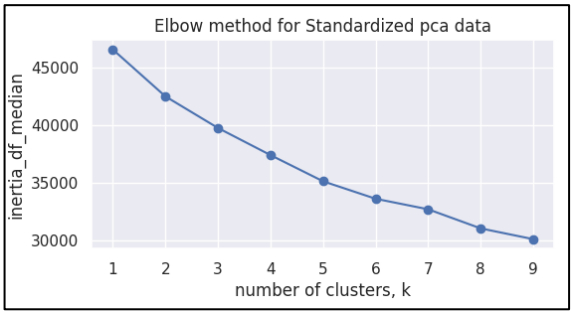
```
[1.56781197e-01 9.42913676e-02 8.76642213e-02 8.45350546e-02
 8.01803733e-02 7.73643838e-02 7.43887069e-02 6.43045447e-02
 4.58962671e-02 4.33285128e-02 3.87183192e-02 3.56717105e-02
 2.77635197e-02 2.14036925e-02 1.93795058e-02 1.55181955e-02
 1.44500232e-02 8.59657018e-03 5.85462943e-03 3.53719166e-03
 3.72013053e-04 1.06126075e-27 1.41821258e-32]
[0.1567812  0.25107256 0.33873679 0.42327184 0.50345221 0.5808166
 0.6552053  0.71950985 0.76540612 0.80873463 0.84745295 0.88312466
 0.91088818 0.93229187 0.95167138 0.96718957 0.9816396  0.99023617
 0.9960908  0.99962799 1.         1.         1.        ]
```

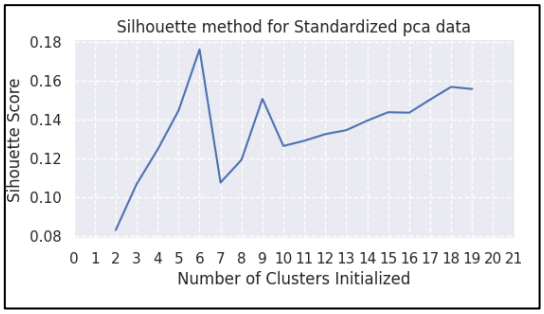
K-means

For K-Means clustering, we leverage the elbow method and silhouette method to select the optimal k cluster.

- K-Means with Standardized PCA components

According to the plots, we select n_cluster=4. Among the 4 clusters, cluster_1 and cluster_3 have higher shares, indicating the more popular newsgroups.



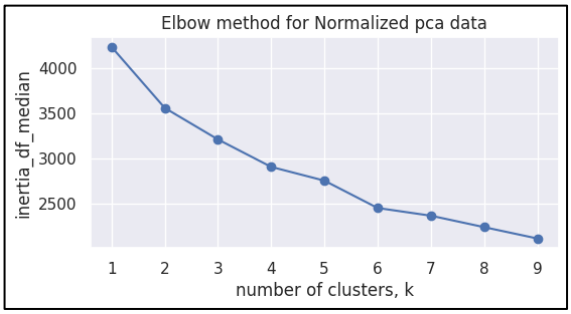


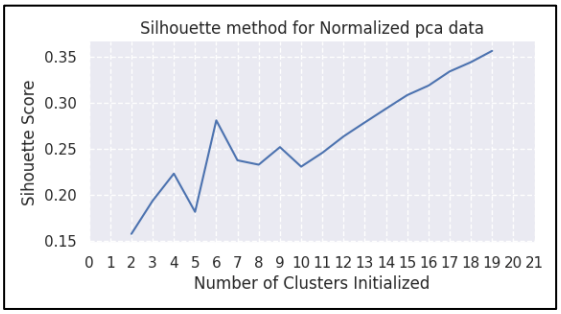

```
km_std_cluster['Kmeans_Clustering'].value_counts()

0    850
3    804
2    357
1    103
Name: Kmeans_Clustering, dtype: int64
```

```
km_df.groupby(['Kmeans_Clustering']).mean()['shares']

Kmeans_Clustering
0    1505.505882
1    2102.242718
2    1771.941176
3    2013.552239
Name: shares, dtype: float64
```

- K-Means with Normalized PCA components

According to the plots, we select n_cluster=4. Among the 4 clusters, cluster_1 and cluster_2 have higher shares, indicating the more popular newsgroups.






```
km_minmax_cluster['Kmeans_Clustering'].value_counts()

0    863
2    558
3    353
1    340
Name: Kmeans_Clustering, dtype: int64
```

```
km_df2.groupby(['Kmeans_Clustering']).mean().shares

Kmeans_Clustering
0    1523.741599
1    2025.120588
2    2000.317204
3    1778.988669
Name: shares, dtype: float64
```

These two KMeans clusterings have similar clustering results. However, both of the 4 clusters have a similar number of shares, which cannot distinguish the popularity of news. Therefore, we would perform further modeling with this clustering result.

DBSCAN

With DBSCAN clustering, we still use normalized and standardized PCA components to build the model. To get a more effective model, we develop a 5-NN

model first to estimate the eps, which is the maximum distance between two samples.

Compared the result below, clustering the standardized PCA components with DBSACN of eps=4 and min_sample=6 performs better than others. We obtain 4 clusters. With DBSCAN, we can see a more accurate model filtering out outliers in the dataset. Therefore, we get 4 different clusters with clear gaps.

- DBSCAN with Standardized PCA components

```
eps 4
\min samples 6
clusters present: [-1  0  1  2]
clusters sizes: [  45 1972   14   83]
Silhouette Score: 0.1547352972415128

eps 4
\min samples 7
clusters present: [-1  0  1  2]
clusters sizes: [  49 1969   13   83]
Silhouette Score: 0.15443617239350013

eps 4
\min samples 8
clusters present: [-1  0  1  2]
clusters sizes: [  51 1969   12   82]
Silhouette Score: 0.15195040378142124

eps 5
\min samples 6
clusters present: [-1  0]
clusters sizes: [   7 2107]
Silhouette Score: 0.2931497140921229
```

```
db_df.groupby(['DBSCAN_Clustering']).mean().shares

DBSCAN_Clustering
-1     8743.622222
 0     1609.835700
 1     2192.857143
 2     1794.349398
Name: shares, dtype: float64
```

- DBSCAN with Normalized PCA components

```
eps 0.6
\min samples 7
clusters present: [-1  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22
 23 24 25 26 27 28]
clusters sizes: [349 120  46   8  41  49 140 157  67  58  60  76 143  79  51  56 130  47
  63  45  63 135  39   6  32  28  20  16   9   8]
Silhouette Score: 0.3371243184630252

eps 0.6
\min samples 8
clusters present: [-1  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22
 23 24 25 26 27]
clusters sizes: [384 116  46  41 139 155  64  55  60  75 142  79  50  56 130  47  63  44
  63 131  38  48  30  28  17   8  15   8   9]
Silhouette Score: 0.3379395738169876

eps 0.7
\min samples 6
clusters present: [-1  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22
 23 24 25 26 27 28 29 30 31 32 33 34]
clusters sizes: [ 99  58 133  47  10  47  16 144  28  10  35  33 164  76  13  62  67  14
  76 149  82  64  12  63 134  72  55  49  64 142  47  12  21  19  13  11]
Silhouette Score: 0.319848310976158
```

Clustering Conclusion:

According to the results above, DBSCAN with standardized PCA components performs better than KMeans with standardized PCA components, as there is a clear gap among different clusters, which could help us explore the popularity of online news. Therefore, we will utilize the data with the DBSCAN clustering labels to perform classification.

(2) Classification

Modeling preparation:

As we want to classify data by the popularity of the online news, we remove the attribute 'shares' from the dataset and add the DBSCAN clustering labels. Moreover, the observations with label 1,2,3 have similar numbers of shares, so we merge these 3 labels' observations into label '0', indicating normal news. For the items with the label '-1', we set the label as '1', indicating the popular news.

```
db_df.groupby(['DBSCAN_Clustering']).mean().shares
```

```
DBSCAN_Clustering
-1    8743.622222
 0    1609.835700
 1    2192.857143
 2    1794.349398
Name: shares, dtype: float64
```

```
data_db_std.DBSCAN_Clustering.value_counts()
```

```
0    2069
1      45
Name: DBSCAN_Clustering, dtype: int64
```

```
target=pd.concat([target0,target1])
target['DBSCAN_Clustering'].value_counts()
```

```
0    1057
1    1057
Name: DBSCAN_Clustering, dtype: int64
```

Obviously, we have an imbalanced dataset. To gain a more accurate classification model, we perform resampling on the dataset. We oversample the group_1 and undersample the group_0. Finally, we get a balanced dataset with 1057 observations in both groups.

```
train_X, test_X, train_y, test_y = train_test_split(data_db_std[predictors], data_db_std[outcome],test_size=0.4,random_state=1)
```

Then we split the data into 60% training data and 40% test data. Besides, we also create a function to display evaluation metrics.

```
def evaluate(model,y_pred):
  model.fit(train_X,train_y)
  y_pred=model.predict(test_X)
  conf_matrix=confusion_matrix(test_y,y_pred)
  acc_score=accuracy_score(test_y,y_pred)
  print("confussion matrix")
  print(conf_matrix)
  val=str(model)
  print(f"Accuracy of {val}:",acc_score)
  print(classification_report(test_y,y_pred))
```

KNN

We perform the KNN classifier with n_neighbour=4, and we get an accuracy = 0.976

```
knn = KNeighborsClassifier(n_neighbors= 4)

knn.fit(train_X,train_y)
#knn.score(test_X,test_y)
```

```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=4)
```

```
y_pred_knn = knn.predict(test_X)
```

```
evaluate(knn,y_pred_knn)
```

```
confussion matrix
[[403  20]
 [  0 423]]
Accuracy of KNeighborsClassifier(n_neighbors=4): 0.9763593380614657
              precision    recall  f1-score   support

           0       1.00      0.95      0.98       423
           1       0.95      1.00      0.98       423

    accuracy                           0.98       846
   macro avg       0.98      0.98      0.98       846
weighted avg       0.98      0.98      0.98       846
```

Random Forest

We develop the random forest classifier with n_estimators=5 and criterion = ‘entropy’, and we get an accuracy =0.985

```
rf = RandomForestClassifier(n_estimators=5, criterion='entropy',random_state=2)
rf.fit(train_X,train_y)
y_pred_rf = rf.predict(test_X)
rf_conf_matrix = confusion_matrix(test_y, y_pred_rf)
rf_acc_score = accuracy_score(test_y, y_pred_rf)
print("confussion matrix")
print(rf_conf_matrix)
print("Accuracy of Random Forest:",rf_acc_score)
print(classification_report(test_y,y_pred_rf))
```

```
confussion matrix
[[411  12]
 [  0 423]]
Accuracy of Random Forest: 0.9858156028368794
              precision    recall  f1-score   support

           0       1.00      0.97      0.99       423
           1       0.97      1.00      0.99       423

    accuracy                           0.99       846
   macro avg       0.99      0.99      0.99       846
weighted avg       0.99      0.99      0.99       846
```

Decision Tree

We develop the Decision Tree classifier with max_depth =6 and criterion = entropy’, and we get an accuracy =0.855

```
dt = DecisionTreeClassifier(criterion = 'entropy',random_state=0,max_depth = 6)
dt.fit(train_X, train_y)
y_pred_dt = dt.predict(test_X)
dt_conf_matrix = confusion_matrix(test_y, y_pred_dt)
dt_acc_score = accuracy_score(test_y, y_pred_dt)
print("confussion matrix")
print(dt_conf_matrix)
print("Accuracy of DecisionTreeClassifier:",dt_acc_score)
print(classification_report(test_y,y_pred_dt))
```

```
confussion matrix
[[325  98]
 [ 25 398]]
Accuracy of DecisionTreeClassifier: 0.8546099290780141
              precision    recall  f1-score   support

           0       0.93      0.77      0.84       423
           1       0.80      0.94      0.87       423

    accuracy                           0.85       846
   macro avg       0.87      0.85      0.85       846
weighted avg       0.87      0.85      0.85       846
```

AdaBoost

We develop the AdaBoost classifier with n_estimators = 5, and we get an accuracy =0.732

```
ada= AdaBoostClassifier(n_estimators=5)
ada.fit(train_X,train_y)
y_pred_ada = ada.predict(test_X)
ada_conf_matrix = confusion_matrix(test_y, y_pred_ada)
ada_acc_score = accuracy_score(test_y, y_pred_ada)
print("confussion matrix")
print(ada_conf_matrix)
print("Accuracy of AdaBoost:",ada_acc_score)
print(classification_report(test_y,y_pred_ada))
```

```
confussion matrix
[[348  75]
 [152 271]]
Accuracy of AdaBoost: 0.7316784869976359
              precision    recall  f1-score   support

           0       0.70      0.82      0.75       423
           1       0.78      0.64      0.70       423

    accuracy                           0.73       846
   macro avg       0.74      0.73      0.73       846
weighted avg       0.74      0.73      0.73       846
```

XGBoost

We develop the AdaBoost classifier with n_estimators = 10,max_depth=15 and learning_rate=0.01, and we get an accuracy =0.945

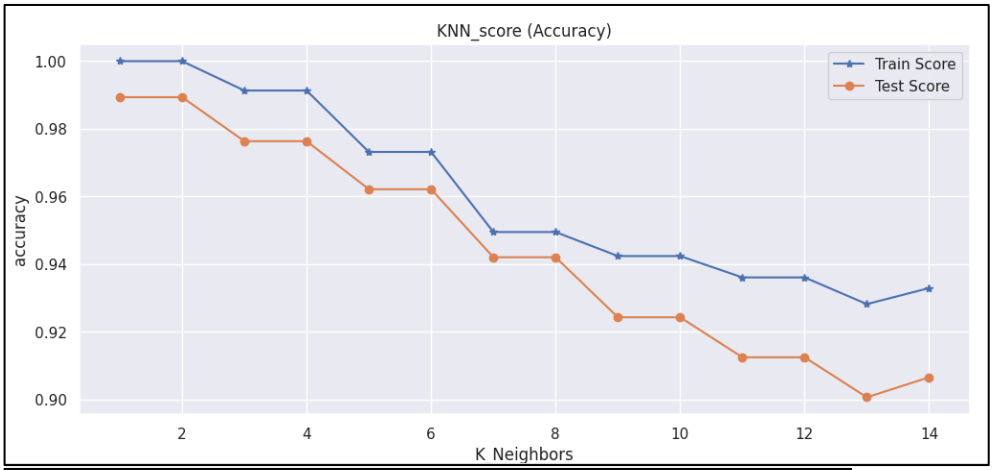
```
xgb = XGBClassifier(learning_rate=0.01, n_estimators=10, max_depth=15)
xgb.fit(train_X, train_y)
y_pred_xgb = xgb.predict(test_X)
xgb_conf_matrix = confusion_matrix(test_y, y_pred_xgb)
xgb_acc_score = accuracy_score(test_y, y_pred_xgb)
print("confussion matrix")
print(xgb_conf_matrix)
print("Accuracy of XGBoost:",xgb_acc_score)
print(classification_report(test_y,y_pred_xgb))
```

```
confussion matrix
[[377  46]
 [  0 423]]
Accuracy of XGBoost: 0.9456264775413712
              precision    recall  f1-score   support

           0       1.00      0.89      0.94       423
           1       0.90      1.00      0.95       423

    accuracy                           0.95       846
   macro avg       0.95      0.95      0.95       846
weighted avg       0.95      0.95      0.95       846
```

4-3 Cross-validation and Hyperparameter Tuning

KNN

To tune the hyperparameter of the KNN model, we utilized GridsearchCV to find the KNN model with the best performance. We get the KNN model with parameters, which are 'metric': 'Minkowski', 'n_neighbors': 1, 'weights': 'uniform'. Finally, we get a KNN model with an accuracy of 0.984. Then we evaluate the modeling with a test set and obtain a 98.9% accuracy.




```
#In case of classifier like knn the parameter to be tuned is n_neighbors
param_grid = { 'n_neighbors':np.arange(1,15),
               'weights' : ['uniform','distance'],
               'metric' : ['minkowski','euclidean','manhattan']}
knn = KNeighborsClassifier()
knn_cv= GridSearchCV(knn,param_grid,cv=5,n_jobs = -1,verbose=2)
knn_cv.fit(train_X,train_y)

print("Best Score:" + str(knn_cv.best_score_))
print("Best Parameters: " + str(knn_cv.best_params_))
```

```
Fitting 5 folds for each of 84 candidates, totalling 420 fits
Best Score:0.9842301826896144
Best Parameters: {'metric': 'minkowski', 'n_neighbors': 1, 'weights': 'uniform'}
```

```
evaluate(best_knn,y_pred_knn_best)
```

```
confussion matrix
[[414   9]
 [  0 423]]
Accuracy of KNeighborsClassifier(n_neighbors=1): 0.9893617021276596
              precision    recall  f1-score   support

           0       1.00      0.98      0.99       423
           1       0.98      1.00      0.99       423

    accuracy                           0.99       846
   macro avg       0.99      0.99      0.99       846
weighted avg       0.99      0.99      0.99       846
```

Random Forest

We utilized RandomizedSearchCV and GridSearchCV to perform the hyperparameter tuning for the random forest model. Compared the results, the randomized search cross-validation performs better than the grid search one, which obtains a 100% accuracy with the test dataset.

RandomizedSearchCV

We set a parameter grid for the randomized search cross-validation and get the best estimator with a score 0.998. Then we evaluate the modeling with a test set and obtain a 100% accuracy.

```
# Number of trees in random forest
n_estimators = [int(x) for x in range(5,20,5)]
# Number of features to consider at every split
max_features = ['auto', 'sqrt']
# Maximum number of levels in tree
max_depth = [int(x) for x in range(5,20,5)]
max_depth.append(None)
# Minimum number of samples required to split a node
min_samples_split = [2, 5, 10]
# Minimum number of samples required at each leaf node
min_samples_leaf = [1, 2, 4]
# Method of selecting samples for training each tree
bootstrap = [True, False]
```

```
print('Best Score:'+ str(rf_random.best_score_))
print('Best parameters:'+str(rf_random.best_params_))
```

```
Best Score:0.9984239558707643
Best parameters:{'n_estimators': 10, 'min_samples_split': 5, 'min_samples_leaf': 1, 'max_features':
```

```
evaluate(best_random_rf,y_pred_rf_best)
```

```
confussion matrix
[[423   0]
 [  0 423]]
Accuracy of RandomForestClassifier(bootstrap=False, max_depth=15, min_samples_split=5,
                       n_estimators=10): 1.0
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       423
           1       1.00      1.00      1.00       423

    accuracy                           1.00       846
   macro avg       1.00      1.00      1.00       846
weighted avg       1.00      1.00      1.00       846
```

GridSearchCV

We set a parameter grid for the grid search cross-validation and get the best estimator with a score 0.98. Then we evaluate the modeling with a test set and obtain a 99% accuracy.

```
param_grid = {
    'bootstrap': [True],
    'max_depth': [10,15,20],
    'max_features': [2, 3],
    'min_samples_leaf': [3, 4, 5,6],
    'min_samples_split': [3,4,5,6],
    'n_estimators': [5, 10, 15]
}
```

```
print('Best Score:'+ str(rf_grid.best_score_))
print('Best parameters:'+str(rf_grid.best_params_))
```

```
Best Score:0.9881759343290047
Best parameters:{'bootstrap': True, 'max_depth': 10, 'max_features': 3, 'min_samples_leaf': 3,
```

```
evaluate(best_grid_rf,y_pred_rf_best_grid)

confussion matrix
[[418   5]
 [  0 423]]
Accuracy of RandomForestClassifier(max_depth=10, max_features=3, min_samples_leaf=3,
                       min_samples_split=3, n_estimators=15): 0.9940898345153665
              precision    recall  f1-score   support

           0       1.00      0.99      0.99       423
           1       0.99      1.00      0.99       423

    accuracy                           0.99       846
   macro avg       0.99      0.99      0.99       846
weighted avg       0.99      0.99      0.99       846
```

Decision Tree

We utilized K-fold cross-validation to perform the hyperparameter tuning for the decision tree. We get the best-performed model with max_depth=15 with a 0.96 accuracy. Then we evaluate the modeling with a test set and obtain a 97.6% accuracy.

```
max_depth = range(5,30,5)
kf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

for val in max_depth:
    score = cross_val_score(DecisionTreeClassifier(max_depth= val, r
    print(f'Average score({val}): {"{:.3f}".format(score.mean())}')

Average score(5): 0.836
Average score(10): 0.947
Average score(15): 0.960
Average score(20): 0.959
Average score(25): 0.959
```

```
evaluate(dt_best,y_pred_dt_best)

confussion matrix
[[403  20]
 [  0 423]]
Accuracy of DecisionTreeClassifier(criterion='entropy', max_depth=15, random_state=0): 0.9763593380614657
              precision    recall  f1-score   support

           0       1.00      0.95      0.98       423
           1       0.95      1.00      0.98       423

    accuracy                           0.98       846
   macro avg       0.98      0.98      0.98       846
weighted avg       0.98      0.98      0.98       846
```

AdaBoost

We utilized GridSearchCV to perform the hyperparameter tuning for the AdaBoost model. We set a parameter grid for the grid search cross-validation and get the best estimator with a score 0.955. Then we evaluate the modeling with a test set and obtain a 99% accuracy.

```
param_grid = {
    'n_estimators': range(10,200,20),
    'learning_rate':[0.5,0.8,0.9,0.95,1]
}
```

```
print('Best hyper-parameters found:')
print(ada_grid.best_params_)
print('Best score:')
print(ada_grid.best_score_)
```

```
Best hyper-parameters found:
{'learning_rate': 1, 'n_estimators': 130}
Best score:
0.9550509973520965
```

```
ada_best=ada_grid.best_estimator_
y_pred_ada_best=ada_best.predict(test_X)
evaluate(ada_best,y_pred_ada_best)
```

```
confussion matrix
[[392  31]
 [  0 423]]
Accuracy of AdaBoostClassifier(learning_rate=1, n_estimators=130): 0.9633569739952719
              precision    recall  f1-score   support

           0       1.00      0.93      0.96       423
           1       0.93      1.00      0.96       423

    accuracy                           0.96       846
   macro avg       0.97      0.96      0.96       846
weighted avg       0.97      0.96      0.96       846
```

XGBoost

We utilized GridSearchCV to perform the hyperparameter tuning for the XGBoost model. We set a parameter grid for the grid search cross-validation and get the best estimator with a score 0.97. Then we evaluate the modeling with a test set and obtain a 99.5% accuracy.

```
param_grid = {
    'n_estimators': range(0,20,5),
    'learning_rate':[0.01,0.03,0.1,0.3,0.8],
    'max_depth':[2,5,8,13,18]
}
```

```
print('Best hyper-parameters found:')
print(xgb_grid.best_params_)
print('Best score:')
print(xgb_grid.best_score_)
```

```
Best hyper-parameters found:
{'learning_rate': 0.8, 'max_depth': 13, 'n_estimators': 15}
Best score:
0.9771380233717634
```

```
xgb_best=xgb_grid.best_estimator_
y_pred_xgb_best=xgb_best.predict(test_X)
evaluate(xgb_best,y_pred_xgb_best)
```

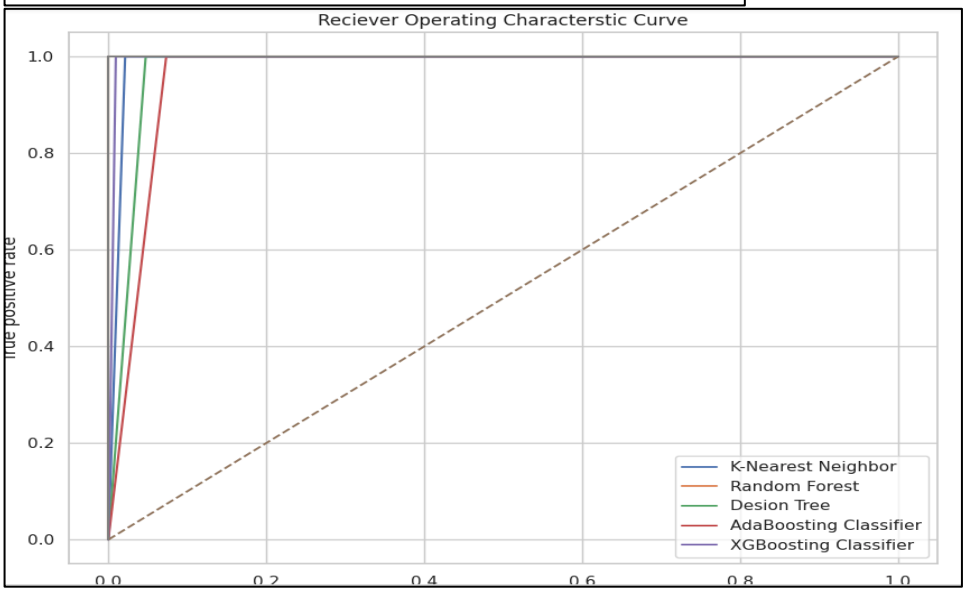
```
confussion matrix
[[419   4]
 [  0 423]]
Accuracy of XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, gpu_id=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=0.8, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=13, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              n_estimators=15, n_jobs=None, num_parallel_tree=None,
              predictor=None, random_state=None, ...): 0.9952718676122931
              precision    recall  f1-score   support

           0       1.00      0.99      1.00       423
           1       0.99      1.00      1.00       423

    accuracy                           1.00       846
   macro avg       1.00      1.00      1.00       846
weighted avg       1.00      1.00      1.00       846
```

5. Model Evaluation

5-1 Summary of all models' performance

According to the table and ROC curves below, the Random Forest model performs best among the classification models, which gets 100% accuracy and recall with the test dataset.

	Accuracy	Recall_label_0
K-Nearest Neighbour	98.936170	97.872340
Random Forest	100.000000	100.000000
Decision Tree	97.635934	95.271868
AdaBoosting	96.335697	92.671395
XGBoosting	99.527187	99.054374



5-2 Recommendations

Based on our modeling results, it appears that the DBSCAN model is a suitable method for clustering online news into distinct groups, which can effectively reflect their popularity. Indeed, one of the key advantages of the DBSCAN model is its ability to identify and filter out outlier data points, which can be particularly useful in identifying online news articles that exhibit exceptional levels of popularity. By effectively clustering the remaining data points, the DBSCAN model can help to reveal patterns and trends in online news that may otherwise be obscured or difficult to discern.

Additionally, our findings suggest that the Random Forest model performs exceptionally well in accurately predicting the extent of popularity based on known attributes prior to publication. While accuracy is an important metric for evaluating the performance of classification models, the recall score of label_0, which represents normal online news, is also crucial when predicting the popularity of online news. This is because identifying normal news that may not generate significant attention can be just as valuable as identifying popular news, especially for editors and publishers who want to optimize the content they produce and promote. Therefore, it is essential to consider both accuracy and recall scores when evaluating classification models for predicting the popularity of online news.

Conclusion

Based on our analysis, we can summarize the conclusions for the two main problems mentioned in the project proposal as follows:

1. Predicting online news popularity

To predict the popularity of online news using features known before publication, we propose a two-step approach. Firstly, we employ the DBSCAN clustering model to group similar news articles based on their features. Secondly, we use the Random Forest classification model to predict the extent of popularity of the news articles in each cluster.
By utilizing this approach, we can allocate resources more effectively and prioritize the creation and promotion of high-performing content. Moreover, individuals and news organizations can avoid publishing uninteresting news, as well as controversial or sensitive content that could harm their reputation.

2. Finding the factors of popular news

After conducting exploratory data analysis (EDA), we were able to identify the factors that are associated with popular online news.

- Titles that are between 8 and 12 words in length tend to be more popular than those that are too short or too long.
- News articles with fewer than 700 words are more attractive to readers.
- Articles with 1-6 references or 0-3 images or 1 video tend to be more popular.
- The presence of repeated keywords is positively associated with popularity.
- Entertainment-focused articles are generally more popular than those in the tech category.
- News articles published on Wednesdays tend to be more popular, while those published on Thursdays are less likely to generate interest.
- Articles with a neutral tone and subjectivity tend to be more popular.

Overall, this approach offers a powerful tool for optimizing the production and promotion of online news content, helping to ensure that it reaches the widest possible audience and maximizes its impact.