



# Northeastern University

Team: -

Sr no.	Name	Email ID	Contribution	Signature
1	Ashish Mhatre	Mhatre.as@northeastern.edu	25%	
2	Harshad Jadhav	Jadhav.h@northeastern.edu	25%	
3	Myron Aruj	Aruj.m@northeastern.edu	25%	
4	Ashwin Kadam	Kadam.as@northeastern.edu	25%	

## IE7374 – Project Report

### Classification: Online News Popularity

**Index: -**

<b>Sr no.</b>	<b>Content</b>	<b>Page no.</b>
<b>1</b>	<b>Problem statement</b>	<b>3</b>
<b>2</b>	<b>Project goal</b>	<b>3</b>
<b>3</b>	<b>Data Description</b>	<b>4-5</b>
<b>4</b>	<b>Data Sources</b>	<b>5</b>
<b>5</b>	<b>Data Exploration</b>	<b>6-11</b>
<b>6</b>	<b>Feature Selection – Stats Test</b>	<b>11-24</b>
<b>7</b>	<b>Principle Component Analysis</b>	<b>24-26</b>
<b>8</b>	<b>Logistic regression</b>	<b>26-28</b>
<b>9</b>	<b>Naive Bayes</b>	<b>28-30</b>
<b>10</b>	<b>SVM</b>	<b>30-31</b>
<b>12</b>	<b>K-means Clustering</b>	<b>31-33</b>
<b>13</b>	<b>Neural Networks</b>	<b>33-35</b>
<b>14</b>	<b>Impact of Project Outcomes</b>	<b>36</b>

## IE7374 – Project Report

### Classification: Online News Popularity

#### **Problem Statement: -**

Internet has played a vital role in spread of information around the globe. With the development and introduction of various new social media platforms the access to information has become very easy. Most of the people have developed a habitual behavior of absorbing the content or information through digital platforms for example learning through YouTube, Sharing thoughts and opinion through Twitter, Facebook, and Instagram. Also study has shown that people choose to read news through online news platforms rather than conventional way. Online news platforms provide an appealing and easy to digest news content. These digital news platforms hold important data such as news engagement, the ratio of likes and dislikes of news, and the number of shares. With the help of these data and various other contributing features, we can identify or classify the popularity and unpopularity of news. Predicting such popularity is valuable for authors, content providers, advertisers and even politicians. Predicting the popularity of online news is becoming a recent research trend.

#### **Project Goal: -**

In this project we will be using various machine learning algorithms and techniques to solve the above defined problems, we will be using classification algorithms to classify whether a news article will become popular or not. Apart from this we will also explore the data and find out the most contributing features to the news popularity. Finally, we will demonstrate several models and evaluate the best models based on performance metrics.

## IE7374 – Project Report

### Classification: Online News Popularity

#### Data Description: -

The original data contains 61 features including URL, rate of positive words, title sentiment polarity and so on. And 39797 samples are provided. All the features are numerical except URL. Data collection period January 7, 2013, to January 7, 2015.

1	url	URL of the article (non-predictive)
2	timedelta	Days between the article publication and the dataset acquisition (non-predictive)
3	n_tokens_title	Number of words in the title
4	n_tokens_content	Number of words in the content
5	n_unique_tokens	Rate of unique words in the content
6	n_non_stop_words	Rate of non-stop words in the content
7	n_non_stop_unique_tokens	Rate of unique non-stop words in the content
8	num_hrefs	Number of links
9	num_self_hrefs	Number of links to other articles published by Mashable
10	num_imgs	Number of images
11	num_videos	Number of videos
12	average_token_length	Average length of the words in the content
13	num_keywords	Number of keywords in the metadata
14	data_channel_is_lifestyle	Is data channel 'Lifestyle'?
15	data_channel_is_entertainment	Is data channel 'Entertainment'?
16	data_channel_is_bus	Is data channel 'Business'?
17	data_channel_is_socmed	Is data channel 'Social Media'?
18	data_channel_is_tech	Is data channel 'Tech'?
19	data_channel_is_world	Is data channel 'World'?
20	kw_min_min	Worst keyword (min. shares)
21	kw_max_min	Worst keyword (max. shares)
22	kw_avg_min	Worst keyword (avg. shares)
23	kw_min_max	Best keyword (min. shares)
24	kw_max_max	Best keyword (max. shares)
25	kw_avg_max	Best keyword (avg. shares)
26	kw_min_avg	Avg. keyword (min. shares)
27	kw_max_avg	Avg. keyword (max. shares)
28	kw_avg_avg	Avg. keyword (avg. shares)
29	self_reference_min_shares	Min. shares of referenced articles in Mashable
30	self_reference_max_shares	Max. shares of referenced articles in Mashable
31	self_reference_avg_shares	Avg. shares of referenced articles in Mashable
32	weekday_is_monday	Was the article published on a Monday?
33	weekday_is_tuesday	Was the article published on a Tuesday?
34	weekday_is_wednesday	Was the article published on a Wednesday?
35	weekday_is_thursday	Was the article published on a Thursday?
36	weekday_is_friday	Was the article published on a Friday?
37	weekday_is_saturday	Was the article published on a Saturday?
38	weekday_is_sunday	Was the article published on a Sunday?
39	is_weekend	Was the article published on the weekend?
40	LDA_00	Closeness to LDA topic 0

## IE7374 – Project Report

### Classification: Online News Popularity

41	LDA_01	Closeness to LDA topic 1
42	LDA_02	Closeness to LDA topic 2
43	LDA_03	Closeness to LDA topic 3
44	LDA_04	Closeness to LDA topic 4
45	global_subjectivity	Text subjectivity
46	global_sentiment_polarity	Text sentiment polarity
47	global_rate_positive_words	Rate of positive words in the content
48	global_rate_negative_words	Rate of negative words in the content
49	rate_positive_words	Rate of positive words among non-neutral tokens
50	rate_negative_words	Rate of negative words among non-neutral tokens
51	avg_positive_polarity	Avg. polarity of positive words
52	min_positive_polarity	Min. polarity of positive words
53	max_positive_polarity	Max. polarity of positive words
54	avg_negative_polarity	Avg. polarity of negative words
55	min_negative_polarity	Min. polarity of negative words
56	max_negative_polarity	Max. polarity of negative words
57	title_subjectivity	Title subjectivity
58	title_sentiment_polarity	Title polarity
59	abs_title_subjectivity	Absolute subjectivity level
60	abs_title_sentiment_polarity	Absolute polarity level
61	shares	Number of shares (target)

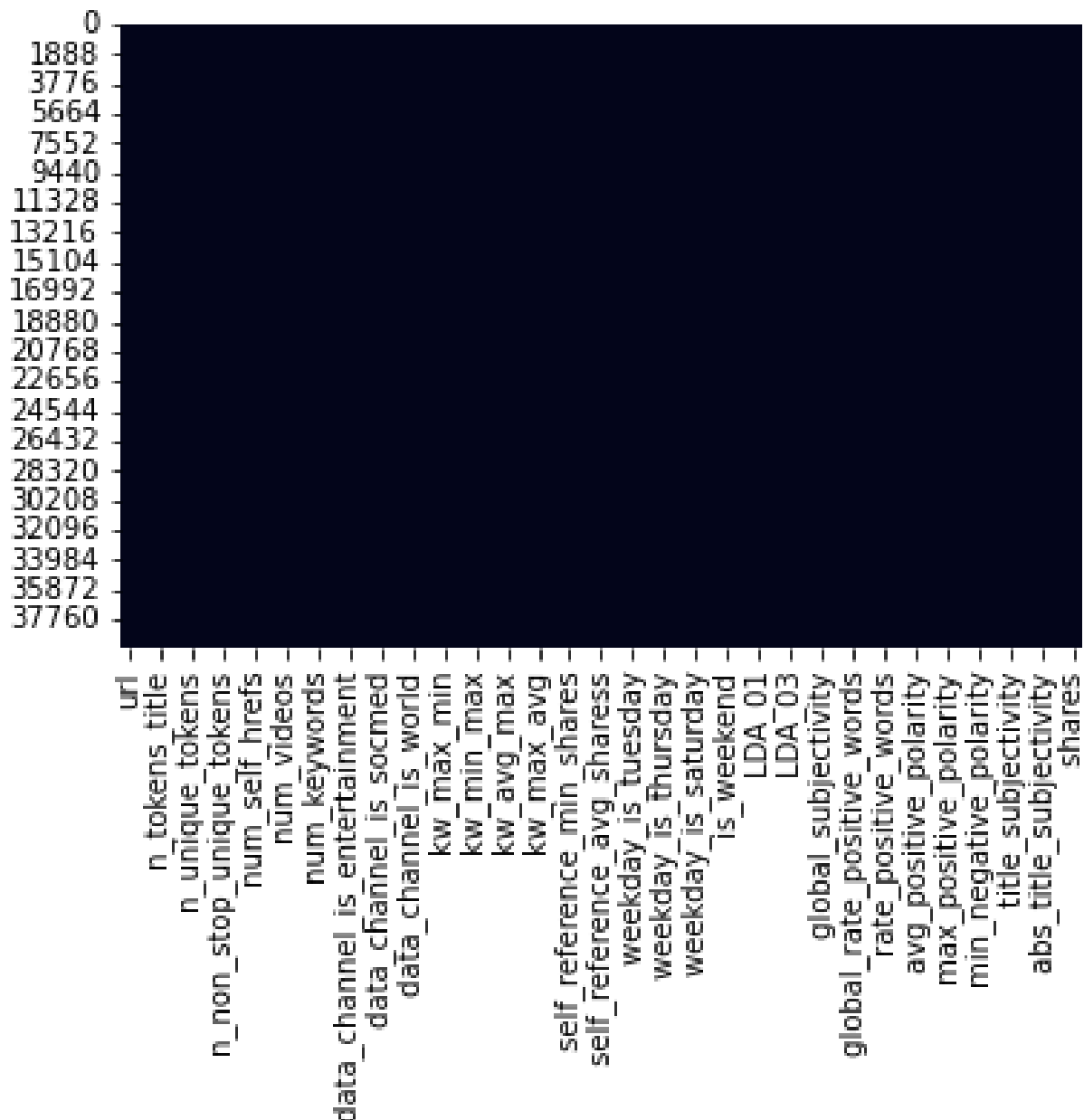
### Data Sources:

We will be taking the data source from UCI ML repository. Link:  
<https://archive.ics.uci.edu/ml/datasets/Online+News+Popularity>

## Classification: Online News Popularity

**Data Exploration:**

In data exploration, we are charting out histograms of all the features, to check the distribution of data. Also, we are plotting the boxplots of all the features, to check the outliers in the data. The data has been checked for null values and we haven't found any NULL values in the data. The following figure represents a heatmap where NULL values are highlighted as we can't see any highlighted cell, we confirm that there is no presence of NULL values in the dataset.



# IE7374 – Project Report

## Classification: Online News Popularity

Figure 1:  $n\_tokens\_title$

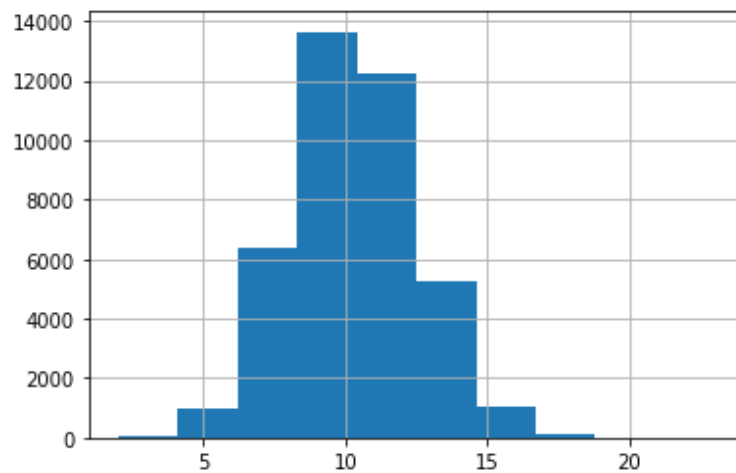


Figure 2:  $n\_unique\_tokens$

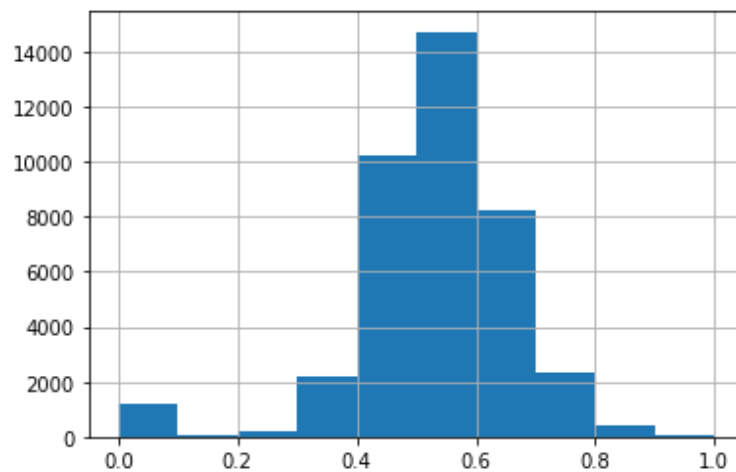
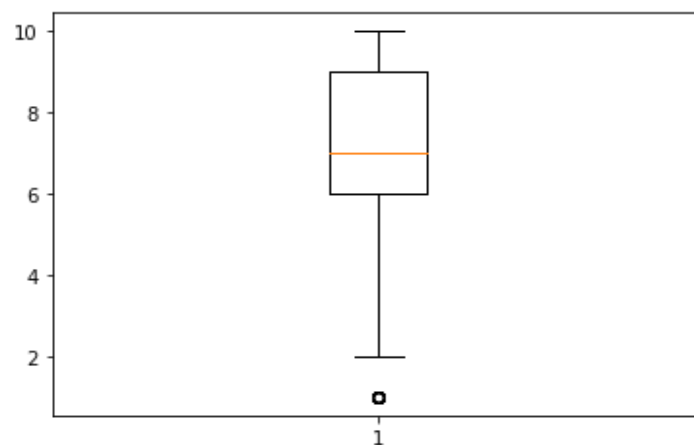


Figure 3:  $num\_keywords$



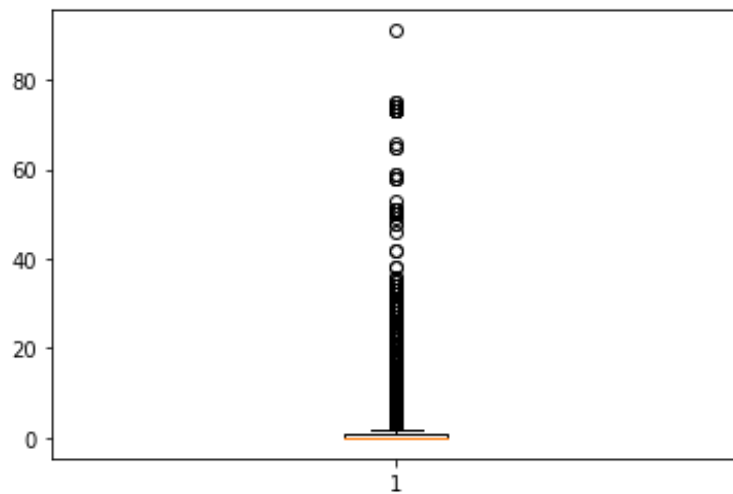
# IE7374 – Project Report

## Classification: Online News Popularity

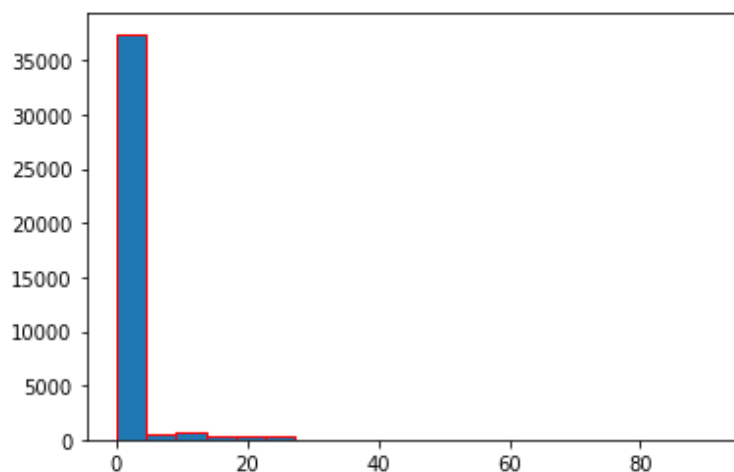
We can see, for the fig 1 and fig 2 we have distribution close to a normal distribution which is desirable for our analysis. also in fig 3, we can see that feature “num\_keyword” have a single outlier which suggests that the distribution of data is uniform, and we have less data points which lie at an abnormal distance from other data points.

In this analysis we will keep the features having a distribution closer to normal distribution and with minimum number of outliers.

*Figure 4: num\_videos*



*Figure 5: num\_videos*

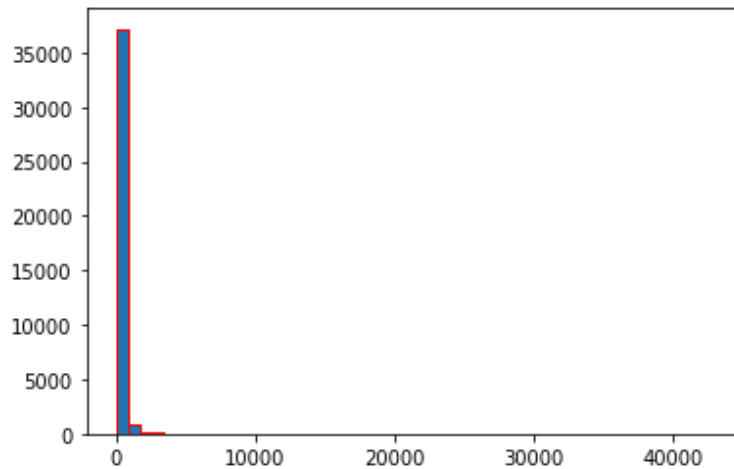




## IE7374 – Project Report

### Classification: Online News Popularity

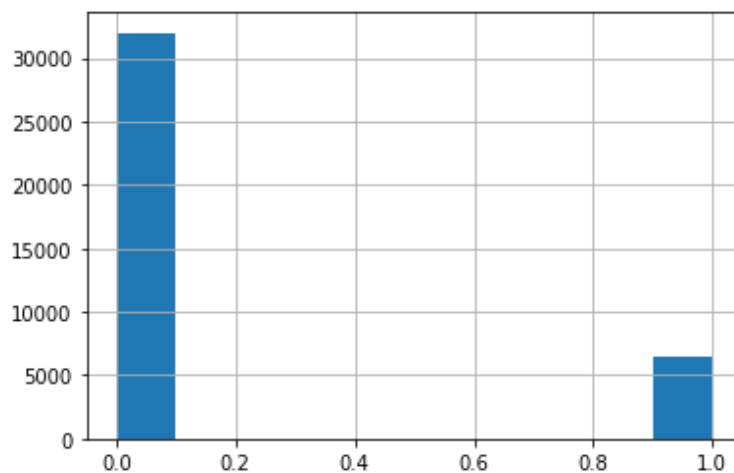
Figure 6: kw\_avg\_min



In second set of figures, we can see that for fig 5 feature – “num\_videos”, we have highly left skewed data that is 95% of data is falling in the range 0-5, which gives us very low variance. Also, fig 4 box plot for the same feature confirms the same and we see that there many outliers as well in this feature.

For the fig 6, we can see that this feature is highly left skewed with near 0 variance. Based on these facts , we decide to drop these features from our analysis, as these are highly left or right skewed data with very low variance or having large number of outliers.

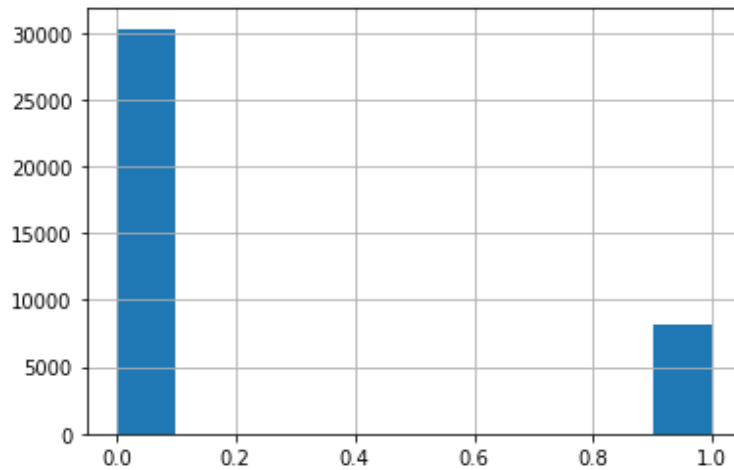
Figure 7: weekday\_is\_monday



## IE7374 – Project Report

### Classification: Online News Popularity

Figure 8: data\_channel\_is\_world



For the third set of figures, we can say that we have the distribution of the data between 0 and 1 which suggest that feature has only two discrete values (0,1).

Similarly, below are the list of discrete binary features:

data_channel_is_lifestyle	weekday_is_tuesday
data_channel_is_entertainment	weekday_is_wednesday
data_channel_is_bus	weekday_is_thursday
data_channel_is_socmed	weekday_is_friday
data_channel_is_tech	weekday_is_saturday
data_channel_is_world	weekday_is_sunday
weekday_is_monday	is_weekend

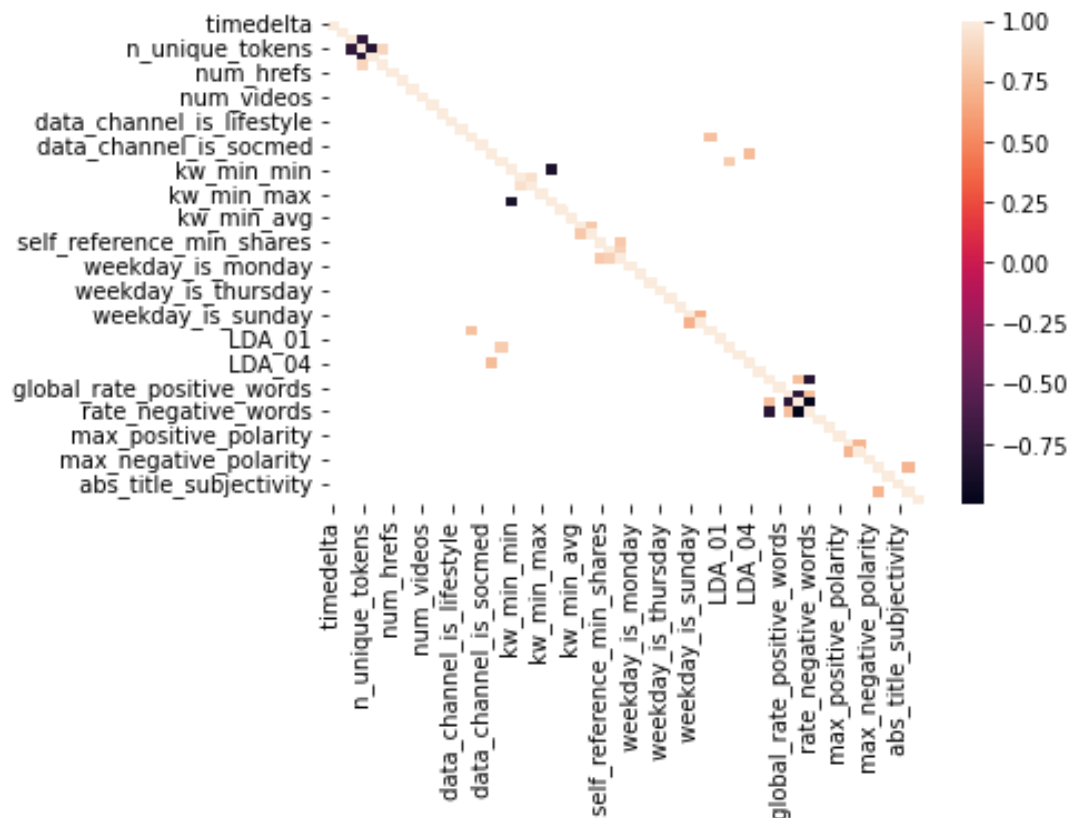
These all featured will be exempted from normalization of data and outlier treatment.

## IE7374 – Project Report

### Classification: Online News Popularity

#### Co-relation plot:

Checking for correlation between the feature as if have two or more independent variables that are very highly correlated. we could run into the multicollinearity conundrum.



So in the above fig, we have calculated and plotted the covariance matrix, highlighted features represent the correlation coefficient which is greater than 0.7. We will be dropping all the highlighted features as these are highly correlated and redundant.

'url'	'self_reference_min_shares'
'timedelta'	'self_reference_avg_shares'
'n_non_stop_words'	'LDA_02'
'n_non_stop_unique_tokens'	'LDA_00'
'num_hrefs'	'LDA_04'
'num_videos'	'abs_title_sentiment_polarity'
'kw_min_min'	'is_weekend'
'kw_max_min'	'min_negative_polarity'
'kw_avg_min'	'n_unique_tokens'
'kw_min_max'	'rate_negative_words'
'kw_max_max'	'rate_positive_words'
'kw_avg_avg'	'data_channel_is_bus'
'self_reference_max_shares'	

## IE7374 – Project Report

### Classification: Online News Popularity

#### Feature selection based on exploratory data analysis:

Based on the study of distribution and outliers in the feature in combination with the correlation matrix we will be dropping the following set of features from our analysis.

##### Drop Features:

'url'	'self_reference_min_shares'
'timedelta'	'self_reference_avg_shares'
'n_non_stop_words'	'LDA_02'
'n_non_stop_unique_tokens'	'LDA_00'
'num_hrefs'	'LDA_04'
'num_videos'	'abs_title_sentiment_polarity'
'kw_min_min'	'is_weekend'
'kw_max_min'	'min_negative_polarity'
'kw_avg_min'	'n_unique_tokens'
'kw_min_max'	'rate_negative_words'
'kw_max_max'	'rate_positive_words'
'kw_avg_avg'	'data_channel_is_bus'
'self_reference_max_shares'	

Total number of features dropped after EDA:- 25

Features remaining :- 36

Performing outlier treatment and normalization on remaining continuous features of the data. Alongside raw data with 36 features we will be bifurcating dataset into 3 additional categories for further analysis.

1. Data with Normalization and Outlier Treatment
2. Data with Normalization
3. Data with Outlier Treatment

#### Converting target variable to dichotomous variable with High and Low Share

Convert all values above 2000 to 1 and rest to 0 in Dependent (Target) Feature “Shares”

## IE7374 – Project Report

### Classification: Online News Popularity

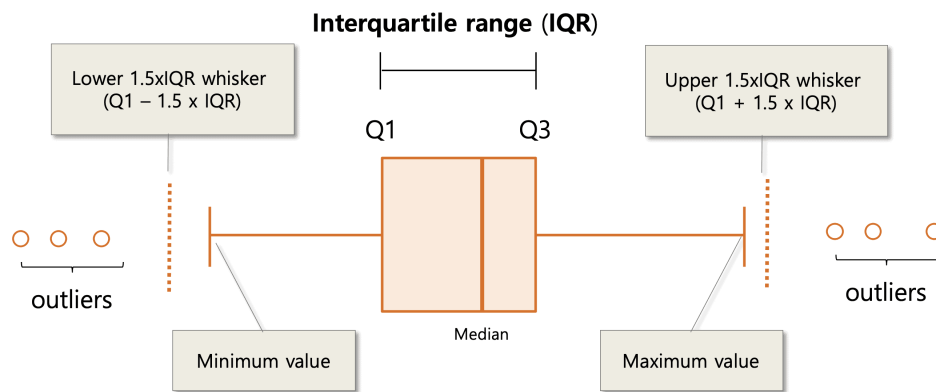
#### Outlier Treatment:

Outlier Trimming:

The interquartile range (IQR) is the difference between the 75th percentile (Q3) and the 25th percentile (Q1) in a dataset. It measures the spread of the middle 50% of values.

You could define an observation to be an outlier if it is 1.5 times the interquartile range greater than the third quartile (Q3) or 1.5 times the interquartile range less than the first quartile (Q1).

**Outliers = Observations  $> Q3 + 1.5 \times IQR$  or  $< Q1 - 1.5 \times IQR$**



```
[ ] 1 for i in Num_cols:
2     #print('Quantile Range of Columns {}'.format(i))
3     percentile25 = new_df[i].quantile(0.25)
4     percentile75 = new_df[i].quantile(0.75)
5     #print('25th Quantile is ', percentile25)
6     #print('75th Quantile is ', percentile75)
7     iqr = percentile75 - percentile25
8     upper_limit = percentile75 + 1.5 * iqr
9     lower_limit = percentile25 - 1.5 * iqr
10    new_df.drop(new_df[ (new_df[i] > upper_limit) | (new_df[i] < lower_limit) ].index, inplace=True)
```

We can't drop these many rows moving to approach two (Capping)

```
[ ] 1 print(new_df.shape)
2
3 print('number of rows dropped or trimmed ', len(df) - len(new_df))
4
5 #We can't drop these many rows moving to approach two (Capping)

(9288, 45)
number of rows dropped or trimmed 30355
```

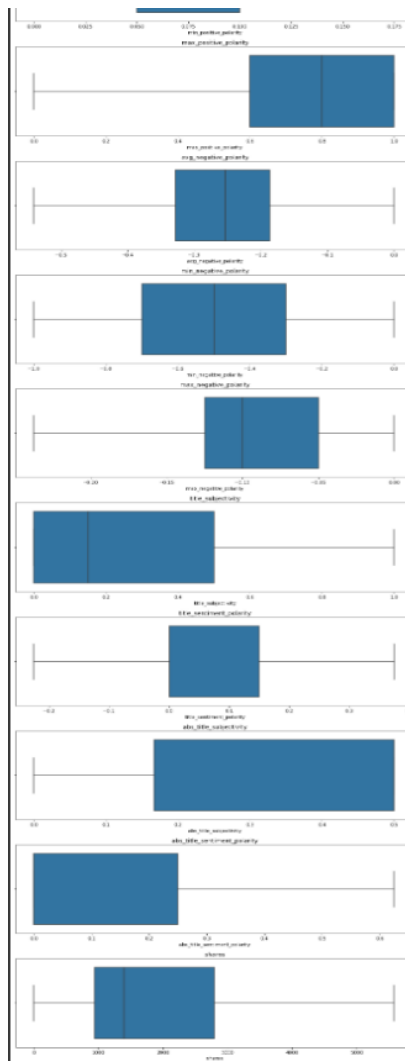
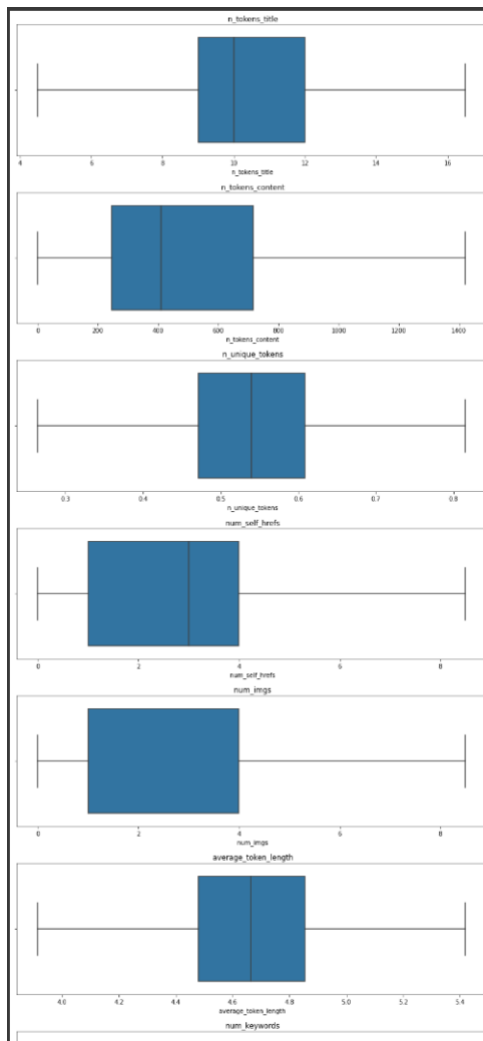
From the above code we can see that using IQR trimming we trim around 30,355 rows. Since we can't drop these many rows, we will move towards second approach which is capping

# IE7374 – Project Report

## Classification: Online News Popularity

### Outlier Capping:

Capping in a sense is similar to trimming the dataset, but the difference here is, while trimming we used IQR or z-score and trimmed the data based on some IQR or z-score value. Here instead of trimming or removing the values from the dataset, we convert the outliers and bring them in the limit or range of our data.



```
[ ] 1 for i in Num_cols:
2     #print('Quantile Range of Columns {}'.format(i))
3     percentile25 = df_cap[i].quantile(0.25)
4     percentile75 = df_cap[i].quantile(0.75)
5     #print('25th Quantile is ', percentile25)
6     #print('75th Quantile is ', percentile75)
7     iqr = percentile75 - percentile25
8     upper_limit = percentile75 + 1.5 * iqr
9     lower_limit = percentile25 - 1.5 * iqr
10    df_cap[i] = np.where( df_cap[i] > upper_limit, upper_limit, np.where( df_cap[i] < lower_limit, lower_limit, df_cap[i] ) )
```

# IE7374 – Project Report

## Classification: Online News Popularity

### Normalization:

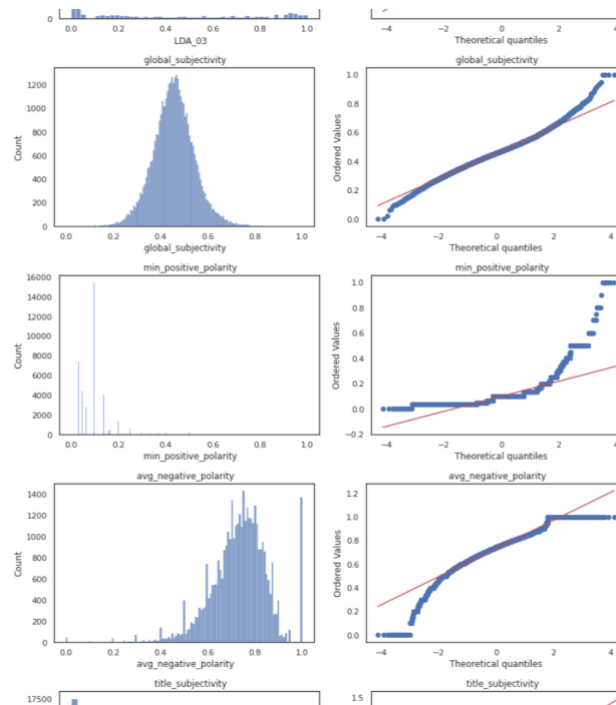
Normalization is generally required when we are dealing with attributes on a different scale, as it may lead to a dilution in effectiveness of an important equally important attribute (on lower scale) because of other attribute having values on larger scale. The goal of normalization is to make every datapoint have the same scale, so each feature is equally important.

Min-max normalization:- For every feature, the minimum value of that feature gets transformed into a 0, the maximum value gets transformed into a 1, and every other value gets transformed into a decimal between 0 and 1.

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

Performing Min Max Normalization

```
[ ] 1 for column in Num_cols:
    2     df_norm[column] = (df_norm[column] - df_norm[column].min()) / (df_norm[column].max() - df_norm[column].min())
```



Plotting Histogram and QQ Plot to validate the normal transformation. We can notice that some features like “Global\_subjectivity” have achieved near perfect transformation. While some features have minor improvement in their distribution.

## Feature selection based on statistical testing

1. Splitting features based on similarity, usage, and domain knowledge for performing statistical test like Chi-Square and Logistic Regression to determine significant features.

Dividing the columns as per feature use and description

```
1 word_features = [' n_tokens_title',
2                 ' n_tokens_content',
3                 ' average_token_length',
4                 ' shares']
5
6 media_features = [' num_imgs', ' shares']
7
8 temporal_features = [' weekday_is_monday',
9                     ' weekday_is_tuesday',
10                    ' weekday_is_wednesday',
11                    ' weekday_is_thursday',
12                    ' weekday_is_friday',
13                    ' weekday_is_saturday',
14                    ' weekday_is_sunday',
15                    ' shares']
16
17 channel_features = [' data_channel_is_lifestyle', ' data_channel_is_entertainment',
18                    ' data_channel_is_socmed',
19                    ' data_channel_is_tech', ' data_channel_is_world', ' shares']
20
21 keyword_features = [' kw_avg_max', ' kw_min_avg', ' kw_max_avg',
22                    ' num_keywords', ' shares']
23
24 reference_features = [' num_self_hrefs', ' shares']
25
26 topic_features = [' LDA_01', ' LDA_03', ' shares']
27
28 subjectivity_features = [' global_subjectivity', ' title_subjectivity', ' abs_title_subjectivity', ' shares']
29
30 sentiment_features = [' global_sentiment_polarity', ' global_rate_positive_words',
31                      ' global_rate_negative_words', ' avg_positive_polarity',
32                      ' min_positive_polarity', ' max_positive_polarity',
33                      ' avg_negative_polarity',
34                      ' max_negative_polarity', ' title_sentiment_polarity',
35                      ' shares']
36 target = [' shares']
```

We will perform Chi-Square test on all Categorical (Discrete Set of features) which are “temporal\_features” and “channel\_features”. For set having continuous independent features and dichotomous dependent variables we will perform a logistic regression statistical test to determine the significance of each feature in their respective sets and overall, in the model.

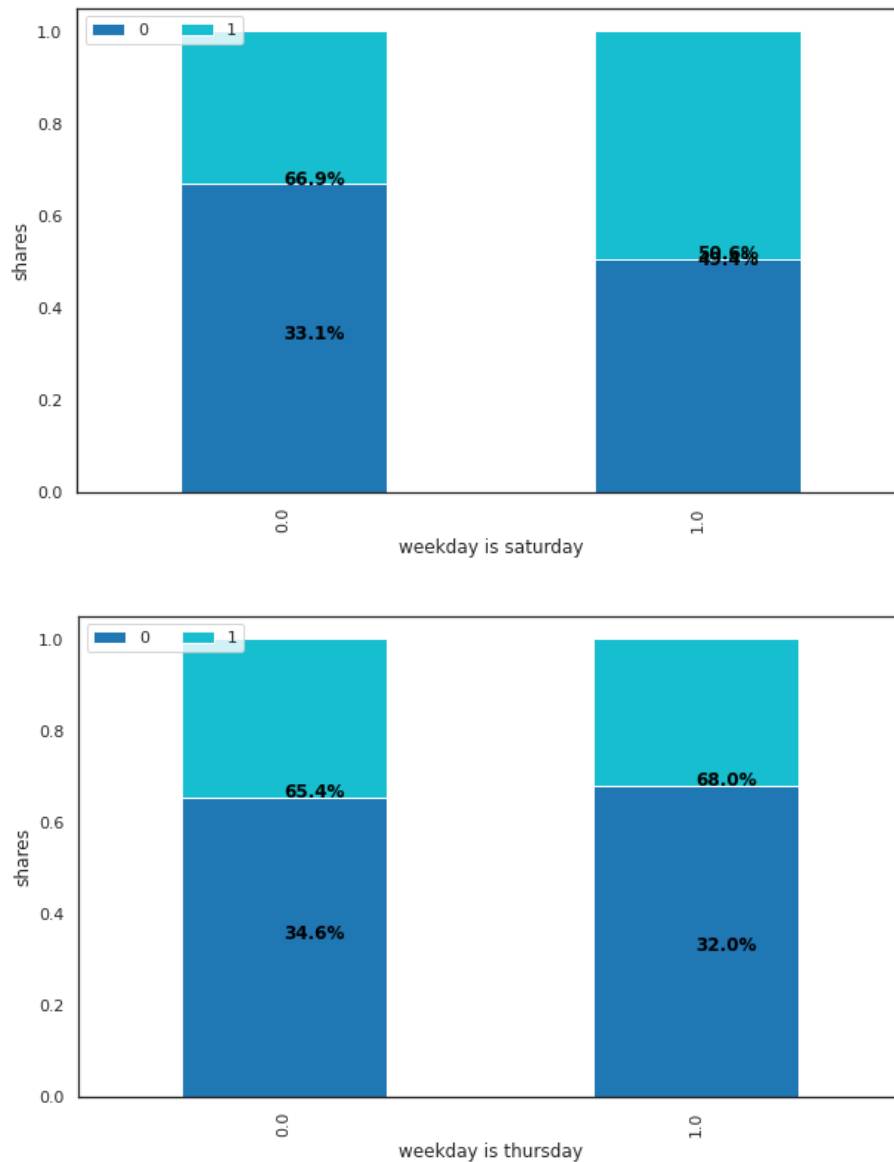


# IE7374 – Project Report

## Classification: Online News Popularity

### 2. Plotting the data to understand underlying trends

Plotting features from “temporal\_features” and “channel\_features” sets to understand the effects of an article being published on any given particular day of week or for any given topic of the article.

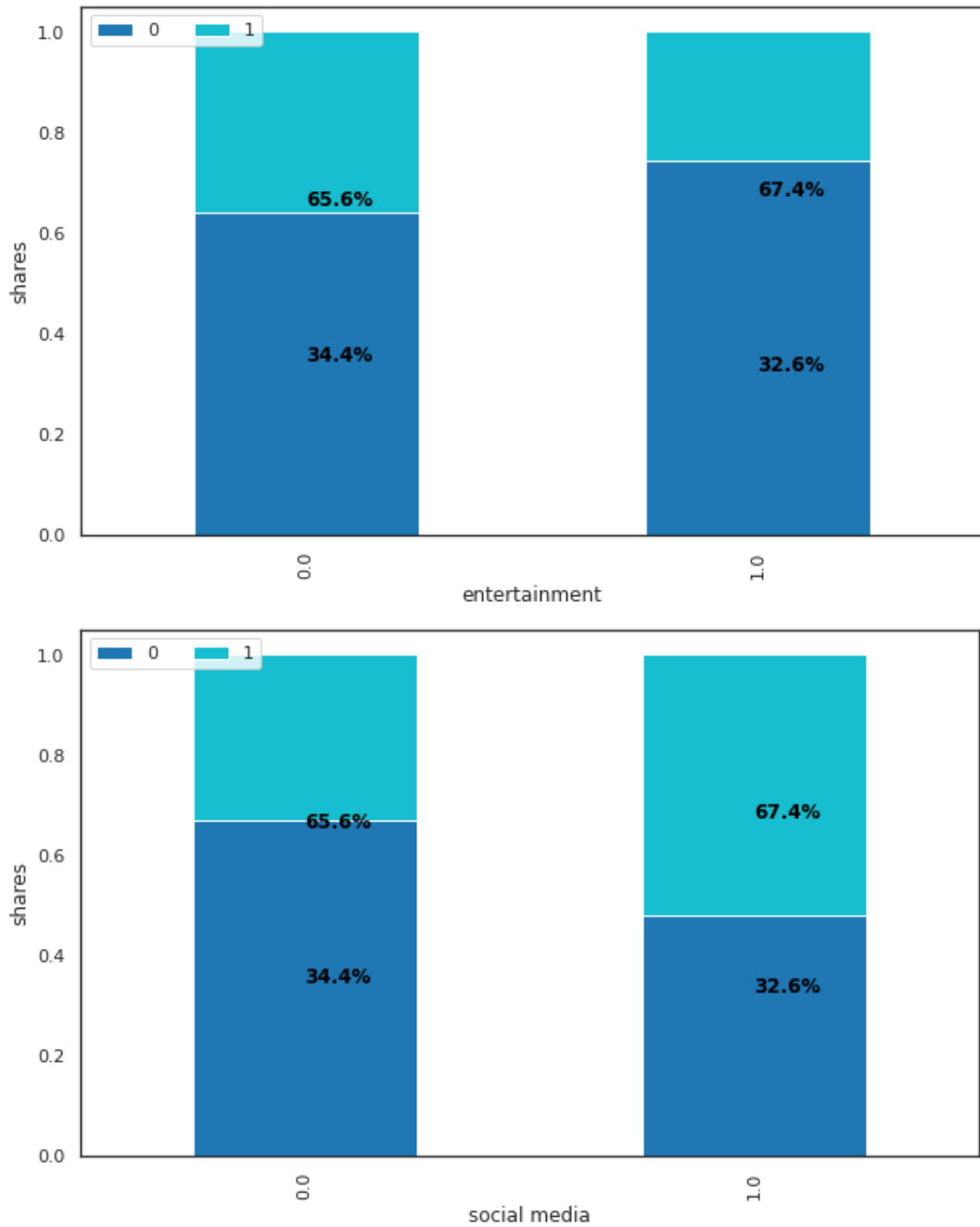


Observing the all the plots on temporal features give us a context that articles published during the weekends tend to have a higher chance of getting a high share than if the article is published on a weekday. Later we will perform Chi-Square test to understand which days are significant in having an effect on the dependent variable “Shares”.

## IE7374 – Project Report

### Classification: Online News Popularity

Plotting the channel features to understand the effect of any particular topic on the shares



From the above two figures we can see that articles which are based on the topic “Entertainment” tend to have a low chance of getting high number of shares. While if the articles are based on “Social Med” we have a high chance of getting high number of shares.

## Classification: Online News Popularity

## 3. Performing Chi – Square Test of Independence

The Chi-Square Test of Independence determines whether there is an association between categorical variables i.e., whether the variables are independent or related in this analysis we will be testing the significance of independent categorical variables like the temporal features and channel features.

$$\chi^2 = \sum_{i=1}^R \sum_{j=1}^C \frac{(o_{ij} - e_{ij})^2}{e_{ij}}$$

where

$o_{ij}$  is the observed cell count in the  $i^{\text{th}}$  row and  $j^{\text{th}}$  column of the table

$e_{ij}$  is the expected cell count in the  $i^{\text{th}}$  row and  $j^{\text{th}}$  column of the table, computed as

$$e_{ij} = \frac{\text{row } i \text{ total} * \text{col } j \text{ total}}{\text{grand total}}$$

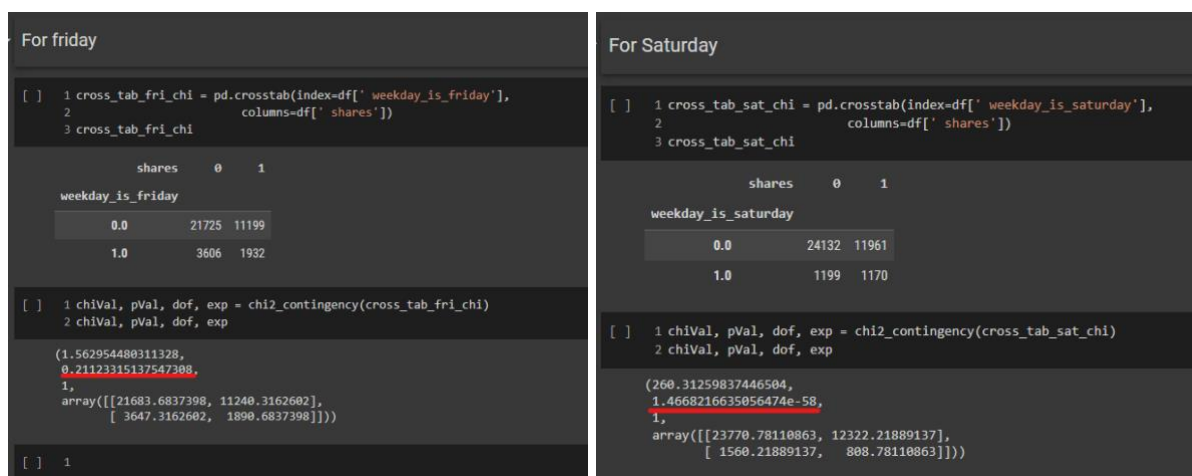
The quantity  $(o_{ij} - e_{ij})$  is sometimes referred to as the *residual* of cell  $(i, j)$ , denoted  $r_{ij}$ .

We will consider the hypothesis as follows:

H0: The “temporal\_features”/ “channel\_features” are independent of “Shares”(Target)

H1: The “temporal\_features”/ “channel\_features” are dependent of “Shares” (Target)

For this analysis we will consider a confidence interval of 95% and reject the null hypothesis if we get a ‘p value’ less 0.05.



In the above figures we can see that for fig1 we calculated the Chi-Square and p value for “Weekday\_is\_Friday” which turns out to be 1.56 and 0.21 respectively which suggests that

## IE7374 – Project Report

### Classification: Online News Popularity

we have a 21% probability that this feature is independent of the target variable under analysis. Thus, we can drop the feature based on the p value.

For Fig2 we see that we have 260 and  $1.46 \times 10^{-58}$  as our Chi-Square and p value respectively which suggest that we have high dependency between the independent feature “Weekday\_is\_Saturday” and dependent feature “Shares”. Therefore, we can conclude that this particular feature is good to have in our analysis.

Performing similar testing on channel features:

```
SocMedia
```

```
[ ] 1 cross_tab_socmed = pd.crosstab(index=df[' data_channel_is_socmed'],
2                                   columns=df[' shares'])
3 cross_tab_socmed
```

	shares	0	1
data_channel_is_socmed			
0.0		24220	11931
1.0		1111	1200

```
[ ] 1 chiVal, pVal, dof, exp = chi2_contingency(cross_tab_socmed)
2 chiVal, pVal, dof, exp
```

```
(345.0597164297639,
5.0463565110546e-77,
1,
array([[23808.97979824, 12342.02020176],
       [ 1522.02020176,   788.97979824]]))
```

```
World
```

```
[ ] 1 cross_tab_world = pd.crosstab(index=df[' data_channel_is_world'],
2                                   columns=df[' shares'])
3 cross_tab_world
```

	shares	0	1
data_channel_is_world			
0.0		18922	11372
1.0		6409	1759

```
[ ] 1 chiVal, pVal, dof, exp = chi2_contingency(cross_tab_world)
2 chiVal, pVal, dof, exp
```

```
(732.0884722529091,
3.1482932672386513e-161,
1,
array([[19951.57074515, 10342.42925485],
       [ 5379.42925485,  2788.57074515]]))
```

We can observe we have a very low p value for both the features “data\_channel\_is\_socmed” and “data\_channel\_is\_world” suggesting a high probability that both features are dependent on the target feature.

Performing Chi-Square test for channel features we get all the features as significant. We will perform logistic statistical test to understand more on the relationship between the dependent variable and the independent variables.

## IE7374 – Project Report

### Classification: Online News Popularity

#### 4. Performing Logistic Model Test

The logistic model is a statistical model that models the probability of one event taking place by having the log-odds for the event be a linear combination of one or more independent variables

We will test all 9 set of features created individually in addition to testing with all features to understand the relationship between the independent variables and dependent variable.

Hypothesis for logistic Model:

H0: Feature X is independent of Target variable y

H1: Feature X is dependent on Target variable y

For this analysis we will consider a confidence interval of 95% and reject the null hypothesis if we get a 'p value' less 0.05.

Logit Regression Results						
Dep. Variable:	shares	No. Observations:	38462			
Model:	Logit	Df Residuals:	38459			
Method:	MLE	Df Model:	2			
Date:	Sat, 06 Aug 2022	Pseudo R-squ.:	0.003330			
Time:	17:49:32	Log-Likelihood:	-24609.			
converged:	True	LL-Null:	-24691.			
Covariance Type:	nonrobust	LLR p-value:	1.938e-36			
	coef	std err	z	P> z	[0.025	0.975]
n_tokens_title	-0.0323	0.005	-6.739	0.000	-0.042	-0.023
n_tokens_content	0.0002	2.24e-05	9.257	0.000	0.000	0.000
average_token_length	-0.0946	0.011	-8.655	0.000	-0.116	-0.073

► From Word Features we have all features as our significant variable

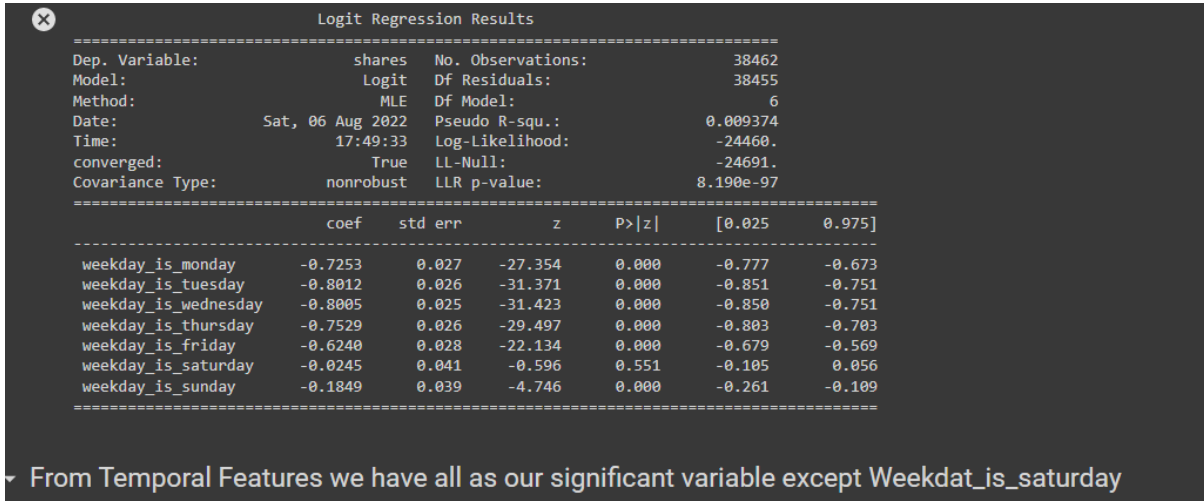
Logit Regression Results						
Dep. Variable:	shares	No. Observations:	38462			
Model:	Logit	Df Residuals:	38459			
Method:	MLE	Df Model:	2			
Date:	Sat, 06 Aug 2022	Pseudo R-squ.:	-0.01016			
Time:	17:49:34	Log-Likelihood:	-24942.			
converged:	True	LL-Null:	-24691.			
Covariance Type:	nonrobust	LLR p-value:	1.000			
	coef	std err	z	P> z	[0.025	0.975]
global_subjectivity	-0.9220	0.063	-14.744	0.000	-1.045	-0.799
title_subjectivity	0.0163	0.038	0.424	0.672	-0.059	0.092
abs_title_subjectivity	-0.5353	0.060	-8.924	0.000	-0.653	-0.418

► From Keyword Features Global\_subjectivity and abs\_title\_subjectivity are significant

## IE7374 – Project Report

### Classification: Online News Popularity

From the above figures we can see that for Word features we have all our set of features as significant features, While for Keyword features, we get “title\_subjectivity” as insignificant feature.



Logit Regression Results

Dep. Variable:	shares	No. Observations:	38462
Model:	Logit	Df Residuals:	38455
Method:	MLE	Df Model:	6
Date:	Sat, 06 Aug 2022	Pseudo R-squ.:	0.009374
Time:	17:49:33	Log-Likelihood:	-24460.
converged:	True	LL-Null:	-24691.
Covariance Type:	nonrobust	LLR p-value:	8.190e-97

	coef	std err	z	P> z	[0.025	0.975]
weekday_is_monday	-0.7253	0.027	-27.354	0.000	-0.777	-0.673
weekday_is_tuesday	-0.8012	0.026	-31.371	0.000	-0.851	-0.751
weekday_is_wednesday	-0.8005	0.025	-31.423	0.000	-0.850	-0.751
weekday_is_thursday	-0.7529	0.026	-29.497	0.000	-0.803	-0.703
weekday_is_friday	-0.6240	0.028	-22.134	0.000	-0.679	-0.569
weekday_is_saturday	-0.0245	0.041	-0.596	0.551	-0.105	0.056
weekday_is_sunday	-0.1849	0.039	-4.746	0.000	-0.261	-0.109

From Temporal Features we have all as our significant variable except Weekdat\_is\_saturday

For Temporal features we get “weekday\_is\_Saturday” as insignificant feature while all others significant but in Chi-Square test, we get it as significant this is due to the fact that we do not have control variables in Chi-Square test. While we have it in Logistic model resulting in Insignificant feature.

After performing the test on all the groups, we select the significant features from the respective groups and perform an iterative testing using all the features until we have all the features as our significant variables in the final iteration of logistic model.

## IE7374 – Project Report

### Classification: Online News Popularity

Logit Regression Results						
=====						
Dep. Variable:	shares	No. Observations:	38462			
Model:	Logit	Df Residuals:	38438			
Method:	MLE	Df Model:	23			
Date:	Sat, 06 Aug 2022	Pseudo R-squ.:	0.05711			
Time:	17:49:36	Log-Likelihood:	-23281.			
converged:	True	LL-Null:	-24691.			
Covariance Type:	nonrobust	LLR p-value:	0.000			
=====						
	coef	std err	z	P> z	[0.025	0.975]
-----						
n_tokens_content	0.0002	2.77e-05	6.055	0.000	0.000	0.000
num_self_hrefs	-0.0155	0.003	-5.060	0.000	-0.022	-0.010
num_imgs	0.0091	0.001	6.155	0.000	0.006	0.012
average_token_length	-0.3751	0.017	-22.122	0.000	-0.408	-0.342
num_keywords	0.0562	0.006	8.930	0.000	0.044	0.069
data_channel_is_entertainment	-0.6751	0.034	-20.025	0.000	-0.741	-0.609
data_channel_is_socmed	0.6139	0.047	13.061	0.000	0.522	0.706
data_channel_is_tech	0.1815	0.033	5.506	0.000	0.117	0.246
data_channel_is_world	-0.6192	0.036	-17.410	0.000	-0.689	-0.550
kw_min_avg	0.0001	1.04e-05	11.388	0.000	9.82e-05	0.000
kw_max_avg	1.881e-05	2.23e-06	8.431	0.000	1.44e-05	2.32e-05
weekday_is_monday	-0.1045	0.040	-2.633	0.008	-0.182	-0.027
weekday_is_tuesday	-0.2297	0.039	-5.889	0.000	-0.306	-0.153
weekday_is_wednesday	-0.2067	0.039	-5.305	0.000	-0.283	-0.130
weekday_is_thursday	-0.1673	0.039	-4.290	0.000	-0.244	-0.091
weekday_is_saturday	0.5027	0.052	9.711	0.000	0.401	0.604
weekday_is_sunday	0.3760	0.050	7.534	0.000	0.278	0.474
LDA_03	0.3606	0.046	7.760	0.000	0.270	0.452
global_subjectivity	0.8248	0.138	5.995	0.000	0.555	1.094
min_positive_polarity	-0.8032	0.174	-4.625	0.000	-1.144	-0.463
avg_negative_polarity	-0.2046	0.099	-2.066	0.039	-0.399	-0.010
title_subjectivity	0.1709	0.040	4.256	0.000	0.092	0.250
title_sentiment_polarity	0.1754	0.044	4.002	0.000	0.090	0.261
abs_title_subjectivity	0.2745	0.068	4.024	0.000	0.141	0.408
=====						

In the final iteration of logistic model, we can see that we have around 24 features which are significant, and we will proceed with model building using these set of 24 independent variables along with a dependent variable “Shares”.

Features	Odds ratio
global_subjectivity	2.281424435
data_channel_is_socmed	1.847623096
weekday_is_saturday	1.653178833
weekday_is_sunday	1.456447134
average_token_length	0.687220553
data_channel_is_world	0.538374965
data_channel_is_entertai	0.509105508
min_positive_polarity	0.44789341

From model co-efficient we can calculate the odds ratio to see the effect of each feature has on the target variable in table above the features highlighted in green have a greater odds of article being high shared while the lower features have a lower odds to contributing for article being high shared.

## Classification: Online News Popularity

**Principal Component Analysis:**

Principal Component Analysis, or PCA, is a dimensionality-reduction method that is often used to reduce the dimensionality of large data sets, by transforming a large set of variables into a smaller one that still contains most of the information in the large set.

Reducing the number of variables of a data set naturally comes at the expense of accuracy, but the trick in dimensionality reduction is to trade a little accuracy for simplicity. Steps involved in PCA:

1. Before applying PCA, the variables will be standardized to have a mean of 0 and a standard deviation of 1. This is important because all variables go through the origin point (where the value of all axes is 0) and share the same variance.
  2. Computing the Eigenvectors and Eigenvalues
    - A. Calculating the covariance matrix. The covariance matrix of the dataset by multiplying the matrix of features by its transpose.
- $$\text{Covariance matrix: } \Sigma = \frac{1}{n-1}((X - \bar{x})^T(X - \bar{x}))$$
- B. Eigen decomposition of the Covariance Matrix
  3. Picking Principal Components Using the Explained Variance
  4. Determining how many components
  5. Project Data onto Lower-Dimensional Linear Subspace

Below is the code snippet of PCA implemented:

```

Principle Component Analysis (PCA)

[ ] 1 class PCA:
2     def __init__(self):
3         pass
4     def standardize_data(self, data):
5         arr = data.to_numpy()
6         rows, columns = arr.shape
7         standardizedArray = np.zeros(shape=(rows, columns))
8         tempArray = np.zeros(rows)
9
10        for column in range(columns):
11            mean = np.mean(arr[:, column])
12            std = np.std(arr[:, column])
13            tempArray = np.empty(0)
14            for element in arr[:, column]:
15                tempArray = np.append(tempArray, ((element - mean) / std))
16            standardizedArray[:, column] = tempArray
17        return standardizedArray
18
19    def fit(self, data):
20        self.X = self.standardize_data(data)
21        covariance_matrix = np.cov(self.X.T)
22        eigen_values, eigen_vectors = np.linalg.eig(covariance_matrix)
23        PC = np.dot(self.X, eigen_vectors)
24        return PC
25
26    def eigen_value_decomp(self, X):
27        covariance_matrix = np.cov(self.X.T)
28        eigen_values, eigen_vectors = np.linalg.eig(covariance_matrix)
29        return eigen_values, eigen_vectors
30
31    def explained_variance(self):
32        eigen_values, eigen_vectors = self.eigen_value_decomp(self.X)
33        variance_explained = []
34        for i in eigen_values:
35            variance_explained.append((i/sum(eigen_values))*100)
36        return variance_explained
  
```

From PCA we will be selecting the components in the combination of 10, 5 and 3 for performing our analysis



## IE7374 – Project Report

### Classification: Online News Popularity

#### Model Exploration:

Once we have the intuition of the Data and some insights of our predictor variables. Next step is to explore various Models. We will implement the below models to understand the accuracy of each one of them and select the best performing one. Below we have explored various models that we have tested on the following dataset

Data sets	Features	Outlier Treatment	Normalization
Dataset 1	25		
Dataset 2	25	✓	
Dataset 3	25		✓
Dataset 4	25	✓	✓
Dataset 5	36		
Dataset 6	36	✓	
Dataset 7	36		✓
Dataset 8	36	✓	✓
Dataset 9(PCA)	10		
Dataset 10(PCA)	5		
Dataset 11(PCA)	3		

We will be testing the datasets on the following models

Models Developed
Logistic regression
Naive Bayes
SVM
K-means Clustering
Neural Networks

## Classification: Online News Popularity

**Logistic regression**

Logistic Regression is Simple Classification algorithm that make use of sigmoid function to return the output/response variable probabilities. These probabilities are then compared with threshold value and a label are assigned to data point which we want to classify.

Logistic Regression is simple and intuitive algorithm which is easy to implement. In Logistic Regression the response variable needs to be binary, and data should be free from missing values and outliers. The logistic function is a sigmoid function, which takes any real input  $t$ , and outputs a value between zero and one.

The logistic function is of the form:

$$p(x) = \frac{1}{1 + e^{-(x-\mu)/s}}$$

```
def sigmoid(self, z):
    sig = 1/ (1 + np.exp(-z))
    return sig

def costFunction(self, X, y):
    y_hat = self.sigmoid( X.dot(self.w) )
    pred = y * np.log( y_hat ) + (1 - y) * np.log( 1 - y_hat )
    cost = - pred.sum()
    return cost

def gradient(self, X, y):
    y_hat = self.sigmoid( X.dot(self.w) )
    grad = (y_hat - y).dot(X)
    return grad

def gradientDescent(self, X, y):

    errors = []
    last_error = float('inf')
    for i in tqdm( range(self.maxiter) ):
        self.w = self.w - self.learningRate * self.gradient(X,y)
        currentError = self.costFunction(X,y)
        diff = last_error - currentError
        last_error = currentError
        errors.append(currentError)
        if np.abs(diff) < self.tolerance:
            print('Model stopped learning')
            break
    self.plot_cost(errors)
    return

def predict(self,X):

    self.X_test = X
    sig = self.sigmoid(self.X_test.dot(self.w))
    return np.around(sig)
```

## IE7374 – Project Report

### Classification: Online News Popularity

#### Accuracy

Data sets	Features	Outlier Treatment	Normalization	Accuracy	Time (sec)
Dataset 1	25			65.83%	1.610
Dataset 2	25	✓		65.83%	29.08
Dataset 3	25		✓	67.03%	51.13
Dataset 4	25	✓	✓	67.00%	51.11
Dataset 5	36			-	-
Dataset 6	36	✓		-	-
Dataset 7	36		✓	-	-
Dataset 8	36	✓	✓	-	-
Dataset 9(PCA)	10			59.87%	77.80
Dataset 10(PCA)	5			58.63%	32.19
Dataset 11(PCA)	3			58.27%	0.236

## Classification: Online News Popularity

**Naïve Bayes:**

It is a classification technique based on Bayes' Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.

$$p(Y | \mathbf{x}) = \frac{p(Y) p(\mathbf{x} | Y)}{p(\mathbf{x})}$$

We have Implemented the below Naïve Bayes algorithm:

**Gaussian naive Bayes**

When dealing with continuous data, a typical assumption is that the continuous values associated with each class are distributed according to a normal (or Gaussian) distribution.

$$p(x = v | C_k) = \frac{1}{\sqrt{2\pi\sigma_k^2}} e^{-\frac{(v-\mu_k)^2}{2\sigma_k^2}}$$

We will be using the Gaussian naive Bayes model for training the continuous features of data and later we will multiply the probability of classes with the probability of Bernoulli naive Bayes which will be training on the discrete features of the dataset. We can multiply the probabilities with each other since each feature is independent.

```
class NaiveBayes:
    def fit(self, X, y):
        n_samples, n_features = X.shape
        self._classes = np.unique(y) # y = 0, y=1
        n_classes = len(self._classes) # no of classes = 2

        # calculate mean, var, and prior for each class
        self._mean = np.zeros((n_classes, n_features), dtype=np.float64)
        self._var = np.zeros((n_classes, n_features), dtype=np.float64)
        self._priors = np.zeros(n_classes, dtype=np.float64)

        for idx, c in enumerate(self._classes):
            X_c = X[y == c]
            self._mean[idx, :] = X_c.mean(axis=0)
            self._var[idx, :] = X_c.var(axis=0)
            self._priors[idx] = X_c.shape[0] / float(n_samples)

    def predict(self, X):
        y_pred = [self._predict(x) for x in X]
        return np.array(y_pred)

    def _predict(self, x):
        posteriors = []

        # calculate posterior probability for each class
        for idx, c in enumerate(self._classes):
            prior = np.log(self._priors[idx])
            posterior = np.sum(np.log(self._pdf(idx, x)))
            posterior = prior + posterior
            posteriors.append(posterior)

        # return class with highest posterior probability
        return posteriors

    def _pdf(self, class_idx, x):
        mean = self._mean[class_idx]
        var = self._var[class_idx]
        numerator = np.exp(-((x - mean) ** 2) / (2 * var))
        denominator = (np.sqrt(2 * np.pi * var)) + 0.001
        return numerator / denominator
```

## Classification: Online News Popularity

## Bernoulli naive Bayes

In the multivariate Bernoulli event model, features are independent Booleans (binary variables) describing inputs. where binary term occurrence features are used rather than term frequencies.

$$p(\mathbf{x} \mid C_k) = \prod_{i=1}^n p_{ki}^{x_i} (1 - p_{ki})^{(1-x_i)}$$

```
def prior_calculation(self):
    for outcome in np.unique(self.y_train):
        outcome_count = sum(self.y_train == outcome)
        self.class_priors[outcome] = outcome_count / self.train_size

def likelihoods_calculation(self):
    for feature in self.columns: #For particular feature
        for outcome in np.unique(self.y_train): #For particular output class
            outcome_count = sum(self.y_train == outcome) #Count of particular class
            feat_likelihood = self.X_train[feature][self.y_train == outcome].index.values.tolist().value_counts().to_dict() #Count of X=0 and X=1 for particular Y outcome
            for feat_val, count in feat_likelihood.items(): #Update the Likelihoods based on Feat_likelihood and Outcome_count
                # (X,Y) -> ((0,0),(0,1),(1,0),(1,1))
                self.likelihoods[feature][str(feat_val) + '_' + str(outcome)] = count/outcome_count

def predict(self, X):
    results = [] #Initialize result array
    X = np.array(X) #X_test
    for query in X: #For each row
        probs_outcome = [] #List
        for outcome in np.unique(self.y_train): #for both Y=1 and Y=0
            prior = self.class_priors[outcome] #take prior for the respective y outcome
            likelihood = 1 #Initialize
            for feat, feat_val in zip(self.columns, query): #Feat = Features and feat_val = Value of Feature (0 or 1)
                likelihood *= self.likelihoods[feat][str(feat_val) + '_' + str(outcome)] #check for Likelihood['Feature'] ['Vae of Feature - Value of Outcome'] Eg P(X=1|y=0)
                #Multiple all likelihoods
            posterior = np.log(likelihood * prior) #Calculate the Posterior
            probs_outcome.append(posterior) #Probs_outcome --> [0: Posterior, 1: Posterior] for a row

    #result = max(probs_outcome, key = probs_outcome.get) #Select the KEY with Maximum Value of the two
    result = probs_outcome
    results.append(result) #Append result for each row

    return np.array(results) #Return predicted array Y-hat
```

## Accuracy:

Data sets	Features	Outlier Treatment	Normalization	Accuracy	Time(sec)
Dataset 1	25			65.29%	4.43
Dataset 2	25	✓		66.41%	4.76
Dataset 3	25		✓	65.28%	4.16
Dataset 4	25	✓	✓	66.41%	4.15
Dataset 5	36			64.96%	4.52
Dataset 6	36	✓		66.45%	4.32
Dataset 7	36		✓	65.02%	4.34
Dataset 8	36	✓	✓	66.41%	4.33
Dataset 9(PCA)	10			64.69%	0.502
Dataset 10(PCA)	5			63.95%	0.383
Dataset 11(PCA)	3			64.39%	0.407

## Classification: Online News Popularity

**SVM**

A support vector machine (SVM) is a machine learning algorithm that analyzes data for classification and regression analysis. SVM is a supervised learning method that looks at data and sorts it into one of two categories. An SVM outputs a map of the sorted data with the margins between the two as far apart as possible while keeping the distance between the data point and the margin minimum. Here, we used Soft Margin SVM. A soft-margin SVM modifies the constraints from the hard-margin SVM by allowing some points to violate the margin. It introduces slack variables  $\xi_i$ , one for each training point, into the constraints:

$$\begin{aligned} y_i(\mathbf{w}^\top \mathbf{x}_i - b) &\geq 1 - \xi_i \\ \xi_i &\geq 0 \end{aligned}$$

Putting the objective and constraints together, the soft-margin SVM optimization problem is:

$$\begin{aligned} \min_{\mathbf{w}, b, \xi_i} \quad & \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n \xi_i \\ \text{s.t.} \quad & y_i(\mathbf{w}^\top \mathbf{x}_i - b) \geq 1 - \xi_i \quad \forall i \\ & \xi_i \geq 0 \quad \forall i \end{aligned}$$

```
import numpy as np

class SVM:
    def __init__(self, learning_rate=0.001, lambda_param=0.1, n_iters=10000):
        self.lr = learning_rate
        self.lambda_param = lambda_param
        self.n_iters = n_iters
        self.w = None
        self.b = None

    def fit(self, X, y):
        n_samples, n_features = X.shape

        y_ = np.where(y <= 0, -1, 1)

        self.w = np.zeros(n_features)
        self.b = 0

        for _ in range(self.n_iters):
            for idx, x_i in enumerate(X):
                condition = y_[idx] * (np.dot(x_i, self.w) - self.b) >= 1
                if condition:
                    self.w -= self.lr * (2 * self.lambda_param * self.w)
                else:
                    self.w -= self.lr * (
                        2 * self.lambda_param * self.w - np.dot(x_i, y_[idx])
                    )
                    self.b -= self.lr * y_[idx]

    def predict(self, X):
        approx = np.dot(X, self.w) - self.b
        return np.sign(approx)
```

## IE7374 – Project Report

### Classification: Online News Popularity

#### Accuracy:

Data sets	Features	Outlier Treatment	Normalization	Accuracy	Time(sec)
Dataset 1	25			63.65%	49.55
Dataset 2	25	✓		65.91%	46.64
Dataset 3	25		✓	65.83%	42.06
Dataset 4	25	✓	✓	65.83%	43.48
Dataset 5	36			-	-
Dataset 6	36	✓		-	-
Dataset 7	36		✓	-	-
Dataset 8	36	✓	✓	-	-
Dataset 9(PCA)	10			65.39%	53.09
Dataset 10(PCA)	5			65.12%	54.38
Dataset 11(PCA)	3			65.12%	48.99

## Classification: Online News Popularity

**K- Means Clustering**

The K-means clustering algorithm computes centroids and repeats until the optimal centroid is found. It is presumptively known how many clusters there are. It assigns data points to a cluster such that the sum of the squared distance between the data points and the cluster's centroid (arithmetic mean of all the data points that belong to that cluster) is at the minimum. The less variation we have within clusters, the more homogeneous (similar) the data points are within the same cluster. K-means implements the Expectation-Maximization strategy to solve the problem.

1. First, we need to provide the number of clusters, K, that need to be generated by this algorithm.
2. Next, choose K data points at random which will be the centroid for the clusters.
3. Iterate the steps below until we find the ideal centroid, which is the assigning of data points to clusters that do not vary.
  - a) The sum of squared distances between data points and centroids would be calculated first.
  - b) At this point, we need to allocate each data point to the cluster that is closest to the others (centroid).
  - c) Finally, compute the centroids for the clusters by averaging all the cluster's data points.

```
def initialize_random_centroids(self, X):
    centroids = np.zeros((self.K, self.num_features))
    for k in range(self.K):
        centroid = X[np.random.choice(range(self.num_examples))]
        centroids[k] = centroid
    return centroids

def create_clusters(self, X, centroids):
    # Will contain a list of the points that are associated with that specific cluster
    clusters = [[] for _ in range(self.K)]
    # Loop through each point and check which is the closest cluster
    for point_idx, point in enumerate(X):
        closest_centroid = np.argmin(
            np.sqrt(np.sum((point - centroids) ** 2, axis=1))
        )
        clusters[closest_centroid].append(point_idx)
    return clusters

def calculate_new_centroids(self, clusters, X):
    centroids = np.zeros((self.K, self.num_features))
    for idx, cluster in enumerate(clusters):
        new_centroid = np.mean(X[cluster], axis=0)
        centroids[idx] = new_centroid
    return centroids

def predict_cluster(self, clusters, X):
    y_pred = np.zeros(self.num_examples)
    for cluster_idx, cluster in enumerate(clusters):
        for sample_idx in cluster:
            y_pred[sample_idx] = cluster_idx

    return y_pred

def fit(self, X):
    centroids = self.initialize_random_centroids(X)
    for it in range(self.max_iterations):
        clusters = self.create_clusters(X, centroids)
        previous_centroids = centroids
        centroids = self.calculate_new_centroids(clusters, X)
        diff = centroids - previous_centroids
        if not diff.any():
            print("Termination criterion satisfied")
            break
    y_pred = self.predict_cluster(clusters, X)

    return y_pred
```



## IE7374 – Project Report

### Classification: Online News Popularity

#### Accuracy:

Data sets	Features	Outlier Treatment	Normalization	Accuracy	Time(sec)
Dataset 1	25			65.82%	9.56
Dataset 2	25	✓		64.26%	9.64
Dataset 3	25		✓	47.63%	9.37
Dataset 4	25	✓	✓	53.76%	7.44
Dataset 5	36			56.21%	13.76
Dataset 6	36	✓		44.54%	15.08
Dataset 7	36		✓	52.13%	7.79
Dataset 8	36	✓	✓	43.37%	10.97
Dataset 9(PCA)	10			47.26%	8.93
Dataset 10(PCA)	5			54.00%	16.76
Dataset 11(PCA)	3			54.01%	16.26

## IE7374 – Project Report

### Classification: Online News Popularity

#### Neural Networks:

Artificial neural networks (ANNs) are comprised of a node layer, containing an input layer, one or more hidden layers, and an output layer. Each node, or artificial neuron, connects to another and has an associated weight and threshold. If the output of any individual node is above the specified threshold value, that node is activated, sending data to the next layer of the network. Otherwise, no data is passed along to the next layer of the network.

We will be building our NN model using Keras:

Model in keras is defined as a sequence of layers and we can add layers until we achieve the desired model. When creating the first layer in keras we need to specify input layer the correct number of input features by passing the number of features in “input\_shape” which in our analysis is 35 features (35,)

While the optimum exact number of layers and number of nodes are found through a process of trial-and-error experimentation we will initially proceed with 5 dense connected layers with an activation function as ReLU and Sigmoid Function for the output layer as we have a binary outcome variable.

```
1 model = Sequential()
2 model.add(Dense(32, input_shape=(35,), activation='relu'))
3 model.add(Dense(16, activation='relu'))
4 model.add(Dense(8, activation='relu'))
5 model.add(Dense(4, activation='relu'))
6 model.add(Dense(2, activation='relu'))
7 model.add(Dense(1, activation='sigmoid'))
```

Model: "sequential\_7"

Layer (type)	Output Shape	Param #
dense_40 (Dense)	(None, 32)	1152
dense_41 (Dense)	(None, 16)	528
dense_42 (Dense)	(None, 8)	136
dense_43 (Dense)	(None, 4)	36
dense_44 (Dense)	(None, 2)	10
dense_45 (Dense)	(None, 1)	3

=====  
Total params: 1,865

Trainable params: 1,865

Non-trainable params: 0  
=====

## IE7374 – Project Report

### Classification: Online News Popularity

#### Accuracy:

<b>Data sets</b>	<b>Features</b>	<b>Accuracy</b>
Dataset 1	25	65.83%
Dataset 2	36	65.84%
Dataset 3(PCA)	10	65.84%
Dataset 3(PCA)	5	65.84%
Dataset 3(PCA)	3	65.84%

## IE7374 – Project Report

### Classification: Online News Popularity

#### Project Outcome:

With the implementation of Multiple models we realised that Logistic Regression model gave maximum accuracy which is 67.03%.

With respect to computation timing of each model we see that naïve bayes took less time to run(4.15 sec) and accuracy of 66.41%, which is close to Logistic Regression model.

Model behave differently when we subjected to different number of features and in presence of outlier. Below tables gives a summary of it.

Model	Outlier	Normalization	Accuracy	Time(sec)
Logistic Regression		✓	67.03%	51.13
Naïve Bayes	✓	✓	66.41%	4.15
SVM		✓	65.83%	42.06
K-Means			65.82%	9.56
Neural Networks			65.84%	-

With respect to target variable, that is number of shares. We found out that the articles that we published on weekends are more likely to be shared. Also, by doing deep diving into other features we found out that articles that are related to entertainment and positive sentiment are more likely to share.