ONLINE NEWS POPULARITY ANALYSIS

Report by:
Gunjan Lalwani
Rishabh Jain
Ann Sajee

INDEX

1. OVERVIEW:

With the help of Internet, the online news can be instantly spread around the world. Most of peoples now have the habit of reading and sharing news online, for instance, using social media like Twitter and Facebook. Typically, the news popularity can be indicated by the number of reads, likes or shares. For the online news stake holders such as content providers or advertisers, it's very valuable if the popularity of the news articles can be accurately predicted prior to the publication. Thus, it is interesting and meaningful to use the machine learning techniques to predict the popularity of online news articles. We have dataset with 39,643 articles from the Mashable website. In this project, based on the dataset including 39,643 news articles from website Mashable, we will try to find the best classification learning algorithm to accurately predict if a news article will become popular or not prior to publication.

2. PROBLEM STATEMENT:

In this project, we have used machine learning techniques to solve a binary classification problem, which is to predict if an online news article will become popular or not prior to publication. The popularity is characterized by the number of shares. If the number of shares is higher than a pre-defined threshold, the article is labeled as popular, otherwise it is labeled as unpopular. Thus, the problem is to utilize a list of article's features and find the best machine learning model to accurately classify the target label (popular/unpopular) of the articles to be published. As the problem can be formulated as a binary classification problem, we have implemented and compared three classification learning algorithms including Logistic Regression, RF, and Adaboost. The best model is selected based on the metrics introduced in next part.

3. METRICS:

As a classification task, we have calculated the following three evaluation metrics: accuracy, F1-score and AUC. For all three metrics, the higher value of the metric means the better performance of model.

(a) Accuracy: Accuracy is direct indication of the proportion of correct classification.
It considers both true positives and true negatives with equal weight and it can be computed as accuracy = (true positives + true negatives)/ dataset size

Although the measure of accuracy might be naive when the data class distribution is highly skewed, but it is still an intuitive indication of model's performance.

(b) F1-score: F1-score is an unweighted measure for accuracy by taking harmonic mean of precision and recall, which can be computed as
F1 = (2 *precision*recall)/ (precision + recall)
It is a robust measurement since it is independent of data class distribution.

(c) AUC: The AUC is the area under the ROC (Receiver Operating Characteristics) curve, which is a plot of the True Positive Rate versus the False Positive Rate. AUC value is a good measure of classifier's discrimination power and it is a more robust measure for model performance.

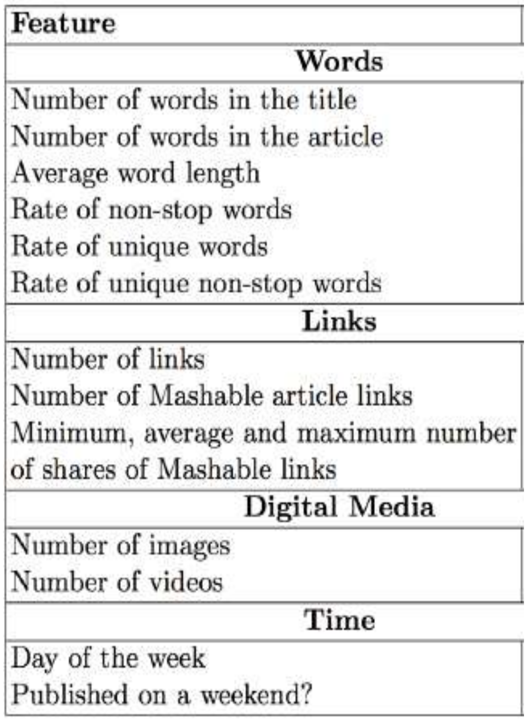
4. ANALYSIS

4.1 DATA EXPLORATION AND VISUALIZATION

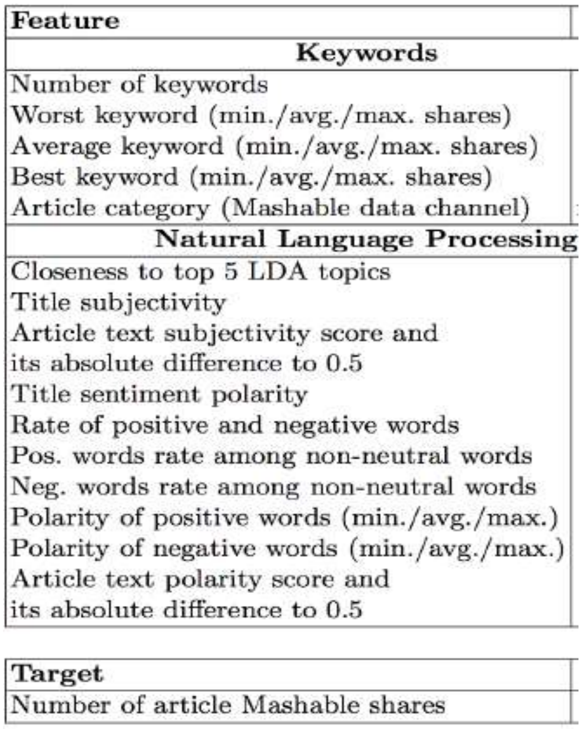
4.1.1 USING MATPLOTLIB, SEABORN, PLOTLY

The dataset is consisted of 39,643 news articles from an online news website called Mashable collected over 2 years from Jan. 2013 to Jan. 2015. It is downloaded from UCI Machine Learning Repository as https://archive.ics.uci.edu/ml/datasets/Online+News+Popularity and this dataset is generously denoted by the author. For each instance of the dataset, it has 61 attributes which includes 1 target attribute (number of shares), 2 non-predictive features (URL of the article and Days between the article publication and the dataset acquisition) and 58 predictive features as shown in Fig. below.

The dataset has already been initially preprocessed. For examples, the categorical features like the published day of the week and article category have been transformed by one-hot encoding scheme, and the skewed feature like number of words in the article has been log- transformed.



Feature
Words
Number of words in the title
Number of words in the article
Average word length
Rate of non-stop words
Rate of unique words
Rate of unique non-stop words
Links
Number of links
Number of Mashable article links
Minimum, average and maximum number of shares of Mashable links
Digital Media
Number of images
Number of videos
Time
Day of the week
Published on a weekend?



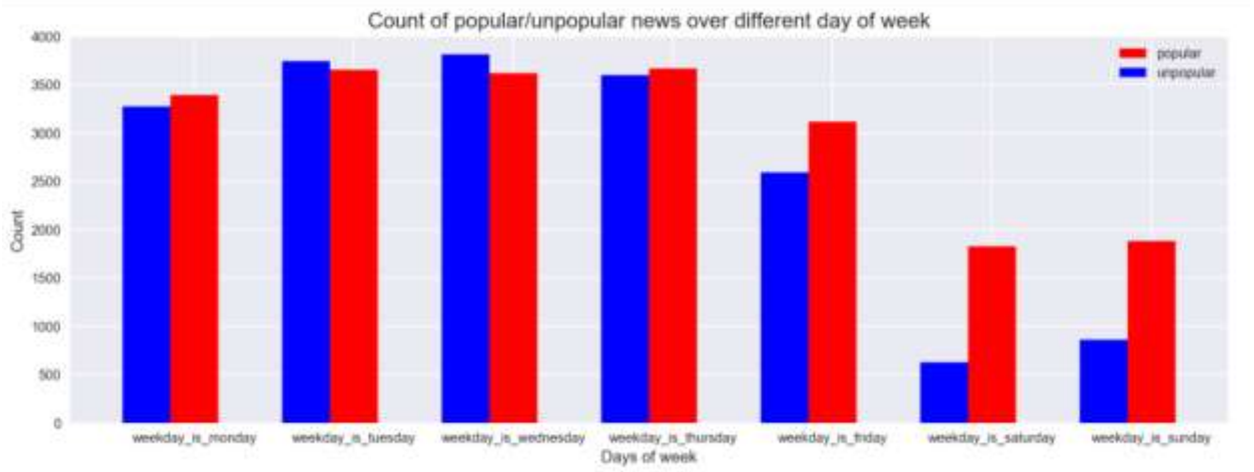
Feature
Keywords
Number of keywords
Worst keyword (min./avg./max. shares)
Average keyword (min./avg./max. shares)
Best keyword (min./avg./max. shares)
Article category (Mashable data channel)
Natural Language Processing
Closeness to top 5 LDA topics
Title subjectivity
Article text subjectivity score and its absolute difference to 0.5
Title sentiment polarity
Rate of positive and negative words
Pos. words rate among non-neutral words
Neg. words rate among non-neutral words
Polarity of positive words (min./avg./max.)
Polarity of negative words (min./avg./max.)
Article text polarity score and its absolute difference to 0.5

Target
Number of article Mashable shares

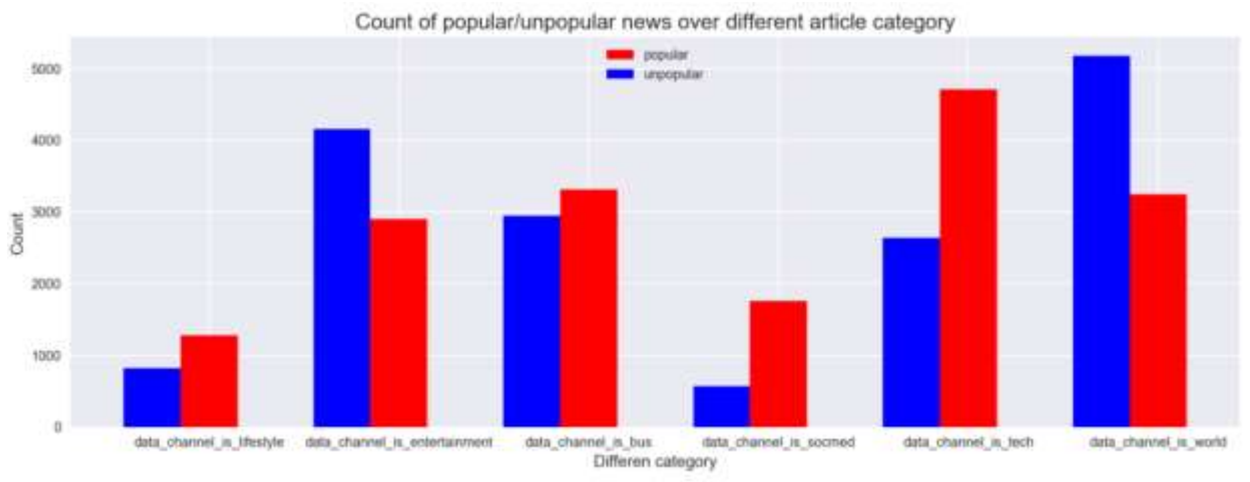
Firstly, we determined the appropriate threshold for number of shares to discriminate the news to be popular or unpopular. So, we show the statistics of the target attribute "shares" in below fig., and we find the median of target attribute is 1,400, thus it is reasonable if we take 1,400 as a threshold. Then we used this threshold to convert the continuous number target attribute into a Boolean label.

```
count      39644.000000
mean        3395.380184
std        11626.950749
min            1.000000
25%          946.000000
50%         1400.000000
75%         2800.000000
max       843300.000000
Name:  shares, dtype: float64
```

By observing various features, I think there are several relevant features like day of the week and article category. In Fig. 3, the count of popular/unpopular news over different day of the week is plotted. We can clearly find that the articles published over the weekends has larger potential to be popular. It makes sense because it is very likely that people will spend more time online browsing the news over the weekends. In Fig below, the count of popular/unpopular news over different article category is plotted. We can observer that in category of technology ("data channel is tech") and social media ("data channel is socmed"),the proportion of popular news is much larger the unpopular ones, and in category of world ("data channel is world") and entertainment ("data channel is entertainment"), the proportion of unpopular news is larger the popular ones. This might reflect that the readers of Mashable prefer the channel of technology and social media much over the channel of world and entertainment.



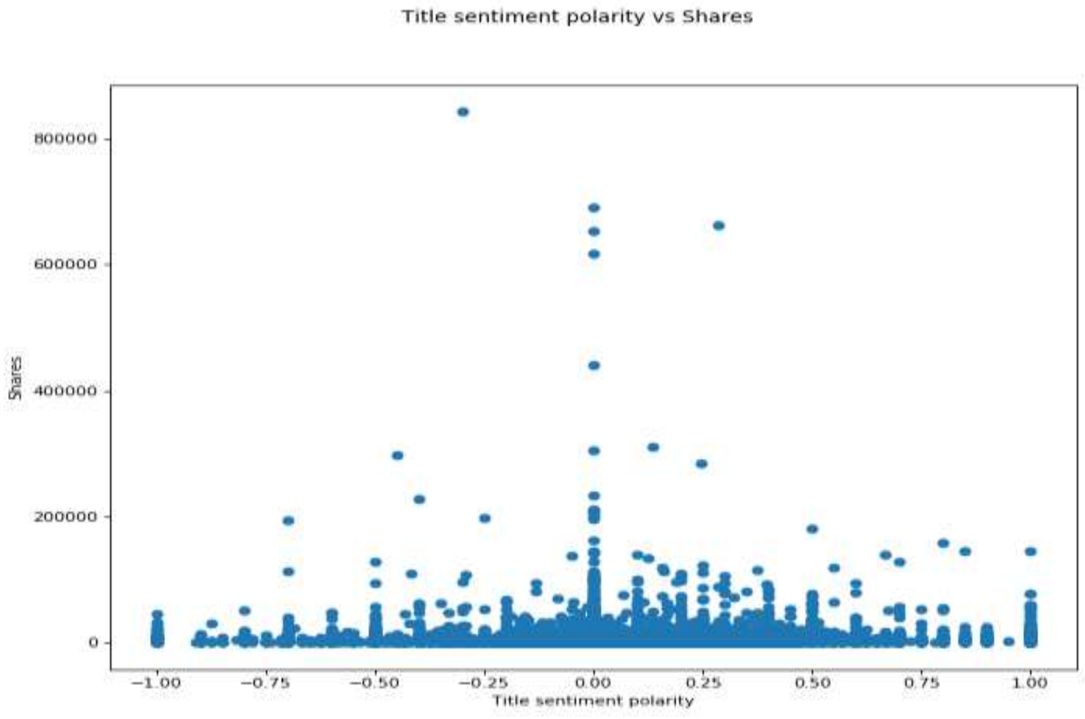
Days of the week vs Shares for popular and unpopular articles



Category for popular and unpopular articles

Title Sentiment Polarity:

How does the title of an article affect its shares? What sentiments should be expressed in title?



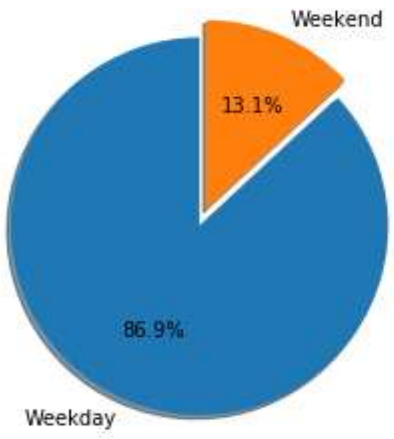
Title sentiment Polarity vs Shares

RESULT:

Mostly the articles have titles which are not too positive or negative. It lies within the range of -0.5 to 0.5. However highest concentration can be seen in the 0 axis i.e. high no. of articles is neutral in nature.

Weekday vs Weekend:

When are most articles published? Is it more on weekdays or weekends?



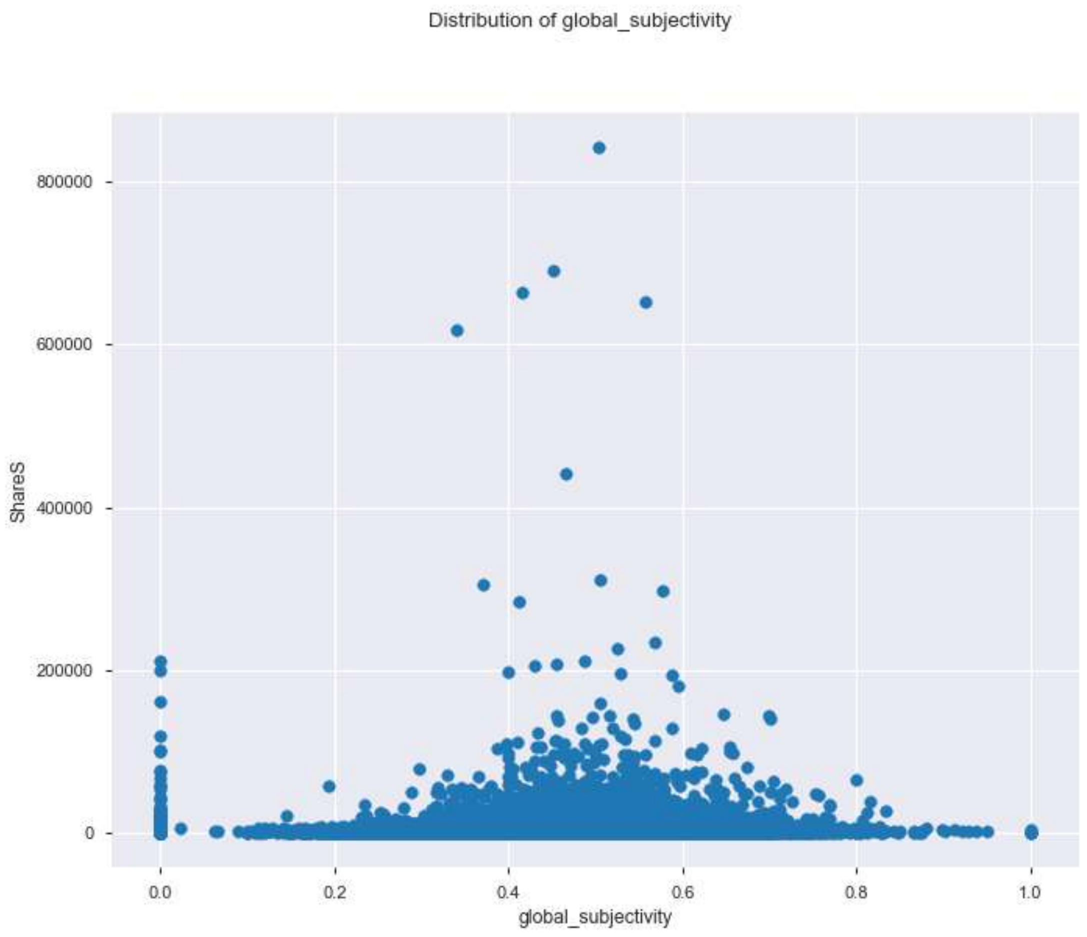
Weekends vs Weekday Pie chart

RESULT:

The above graph shows that most of the articles are published in Weekdays. However, in order to get more popularity on must publish the article on weekends.

Global subjectivity:

Understanding the global subjectivity of an article is extremely important when it is being read by millions of people across the globe. Will it appeal to masses?



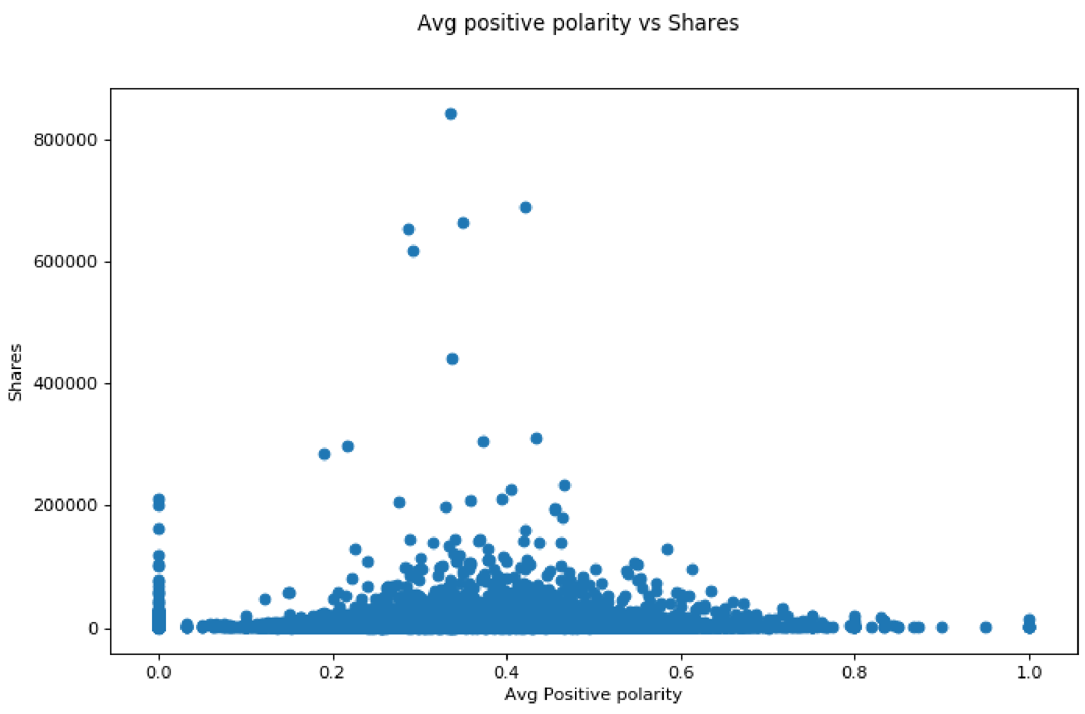
Distribution of Global subjectivity

RESULT:

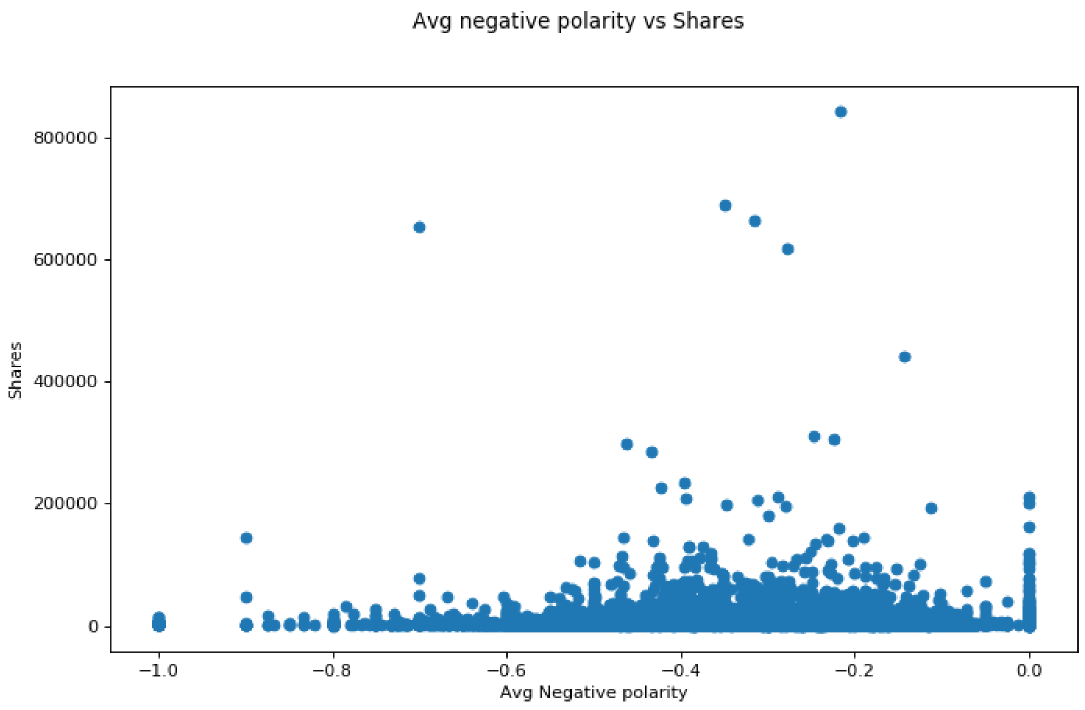
Maximum of global subjectivity lies from 0.2 to 0.8. A significant outlier lies at global subjectivity of 0.503 with share of 957. Hence, we conclude that most of the articles with medium global subjectivity are maximum shares.

Average Polarity:

How is the content of the article? Is it highly positive or negative?



Average Positive Polarity



Average Negative polarity

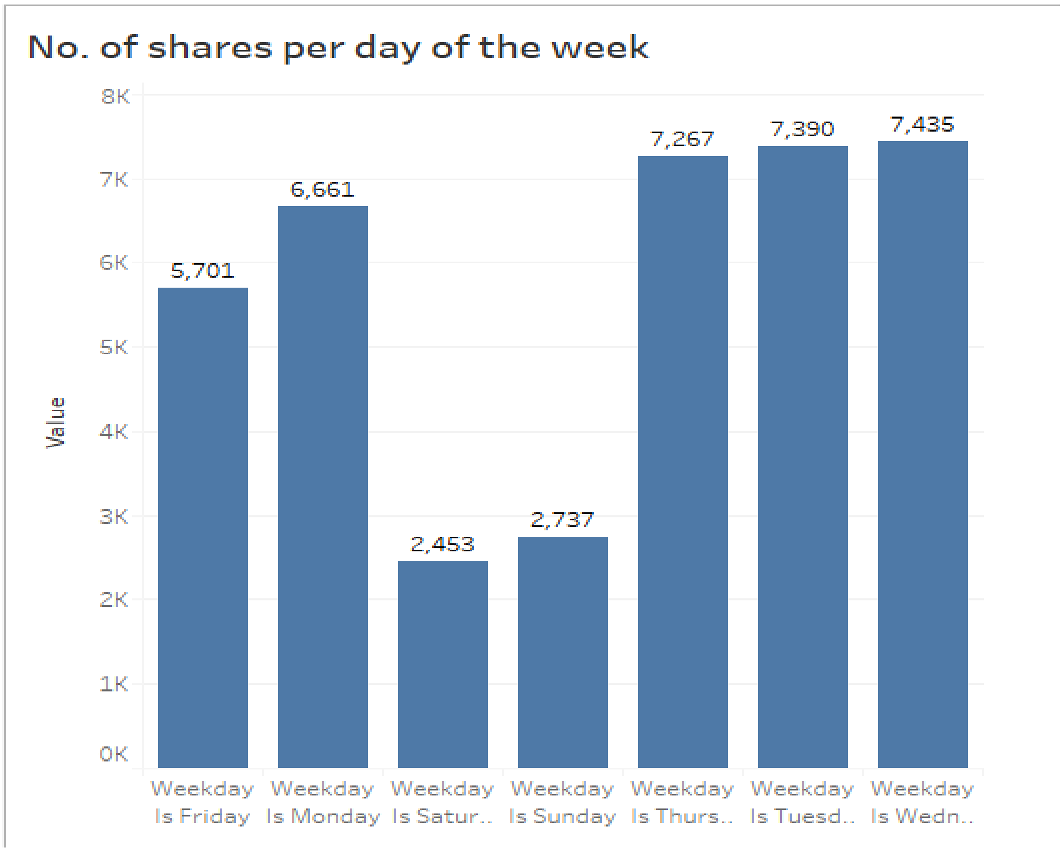
RESULT:

Most of the shared articles are slightly to medium positive or mostly neutral in nature.

4.1.2 USING TABLEAU

Tableau is a great data visualizing tool using which one can explore data and get useful insights from it. We have visualized the dataset in Tableau as well and found useful insights from the dataset.

1. **Number of shares per day of the week:**
   On which days are most of the articles shared?



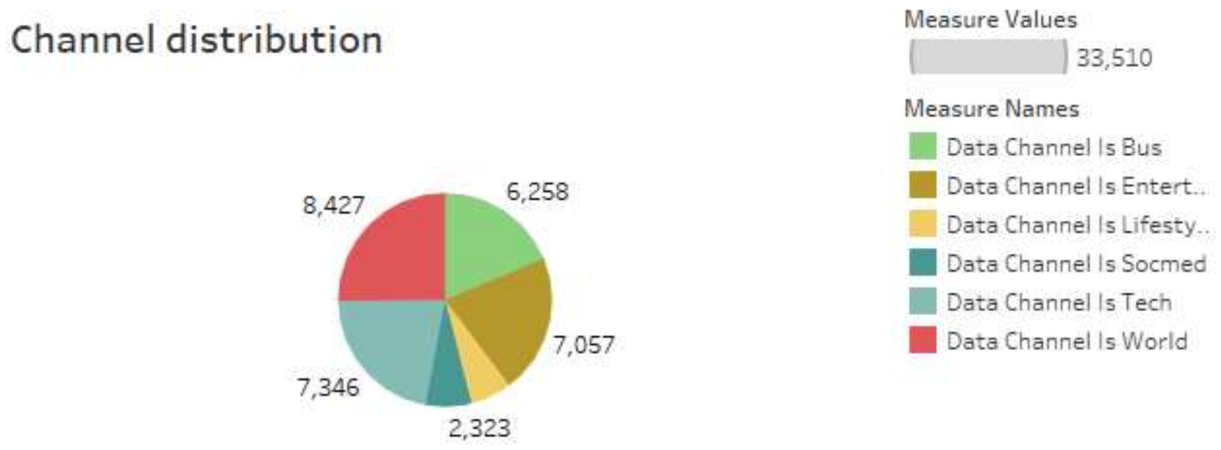
Number of shares per day of the week

RESULT:

Most of the articles are shared on Weekdays rather than weekends. However, after our analysis, we have found that publishing an article on weekends will result in more shares as more people read online news during weekends.

2. **Channel distribution:**

How is the channel distribution amongst shares? Which channel is most popular?



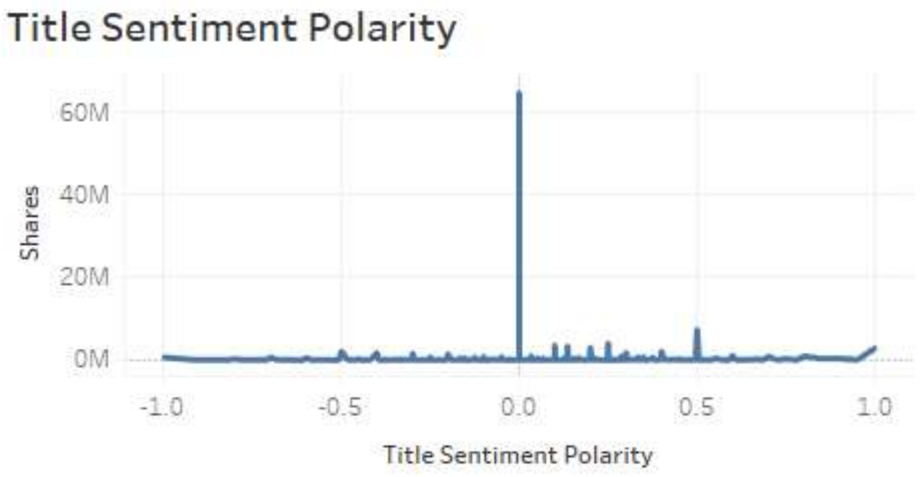
Channel distribution

RESULT:

We can conclude that most of the articles published are related to world news

3. **Title sentiment polarity:**

   How is the polarity of title? Is it positive, negative or neutral?

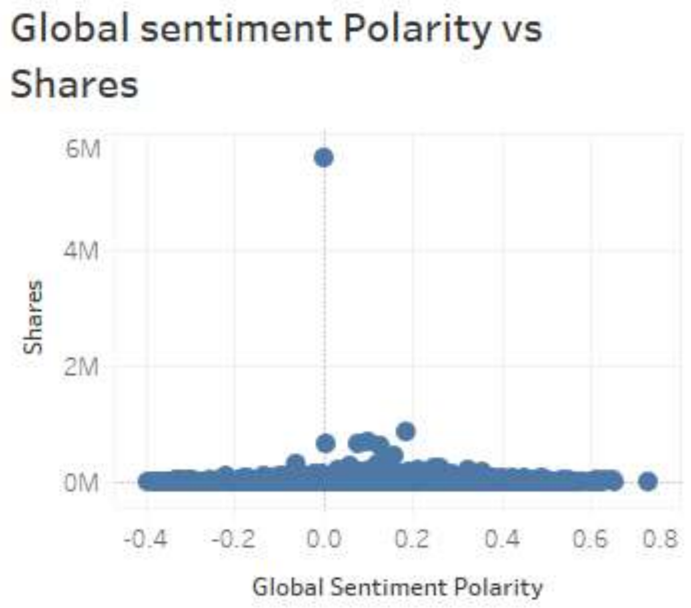


RESULT:

Most of the articles have their title as neutral and rest positive in nature.

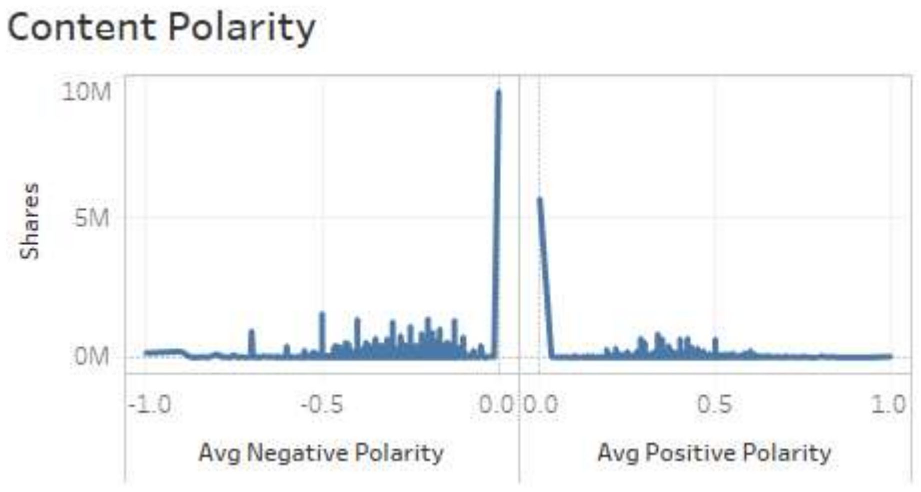
4. **Global sentiment Polarity:**

   Is the article appealing to the masses across the globe?



RESULT:
Most of the articles lie in the range of -0.2 to 0.3. That means it slightly negative to neutral to slightly positive.

5. **Content polarity:**
How is the content of the article? Is it positive, negative or neutral?



RESULT:
Most of the articles have their content neutral.

4.2 ALGORITHMS AND TECHNIQUES

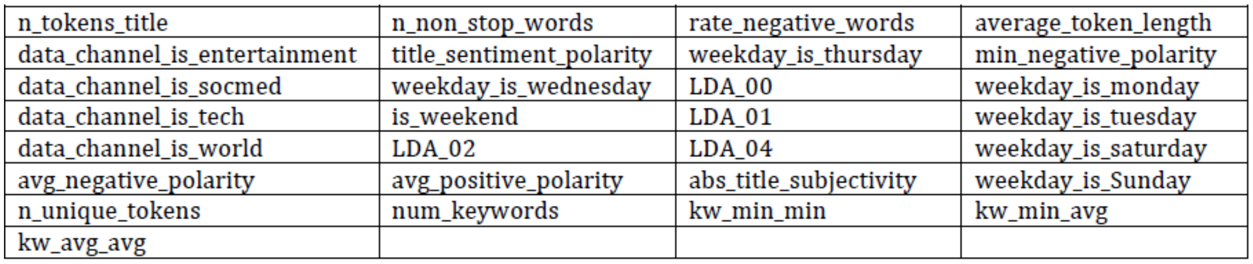
Since we formulate this problem as a binary classification problem.
In this project, three classification learning algorithms including Logistic Regression, RF, and Adaboost, Decision Tree, Support Vector Machine, etc are implemented and compared based on the evaluation metric such as accuracy, F1-score and Area Under ROC Curve (AUC). All the learning algorithm are implemented by sklearn toolbox.

5 METHODOLOGY

5.1 DATA PREPROCESSING

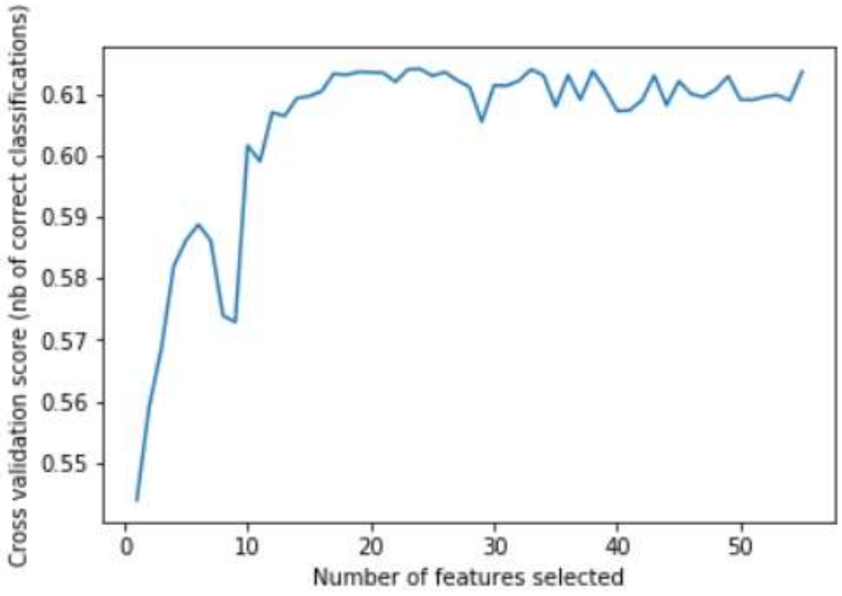
As mentioned above, some data preprocessing works have been done by the data's
donator. The categorical features like the published day of the week and article category
have been transformed by one-hot encoding scheme, and the skewed feature like number of
words in the article has been log-transformed. Based on this, I further preprocess the dataset
by normalizing the numerical feature to the interval [0; 1] such that each feature is treated
equally when applying supervised learning. I also select the median of target attribute as
The threshold to convert the continuous target attribute to Boolean label.
Since there are 58 features in the dataset, it is reasonable to conduct a feature selection to
reduce the data noise and increase the algorithm running speed. One effective way is using
recursive feature elimination with cross validation (RFECV) to automatically select the most
Significant features for certain classifier. Sklearn provides a function called REFCV() that
can help us.
Firstly, we run RFECV with a logistic regression estimator. The cross validation score versus the number of feature selected. From the figure, we can find there is drop of score when number of features is 29. Thus RFECV algorithm selects 29 most relevant features from 58 original features. The selected 29 features are listed below figure.
Interestingly, the day of week and article category features are included in these 29 features
as we discussed and visualized in above section.

Logistic Regression:



n_tokens_title	n_non_stop_words	rate_negative_words	average_token_length
data_channel_is_entertainment	title_sentiment_polarity	weekday_is_thursday	min_negative_polarity
data_channel_is_socmed	weekday_is_wednesday	LDA_00	weekday_is_monday
data_channel_is_tech	is_weekend	LDA_01	weekday_is_tuesday
data_channel_is_world	LDA_02	LDA_04	weekday_is_saturday
avg_negative_polarity	avg_positive_polarity	abs_title_subjectivity	weekday_is_Sunday
n_unique_tokens	num_keywords	kw_min_min	kw_min_avg
kw_avg_avg			

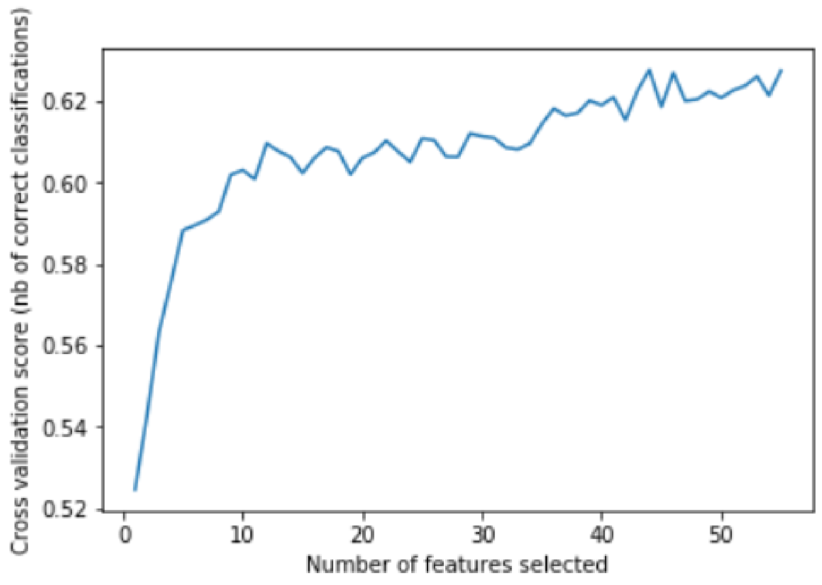
Figure: 29 features selected using RFECV with logistic regression estimator.



```
24
['n_tokens_title' 'n_tokens_content' 'num_hrefs' 'num_self_hrefs'
 'num_imgs' 'num_videos' 'average_token_length' 'num_keywords'
 'data_channel_is_entertainment' 'data_channel_is_socmed'
 'data_channel_is_tech' 'data_channel_is_world' 'kw_min_min' 'kw_max_min'
 'kw_max_max' 'kw_min_avg' 'kw_max_avg' 'kw_avg_avg'
 'self_reference_min_shares' 'self_reference_avg_sharess'
 'weekday_is_saturday' 'is_weekend' 'LDA_02' 'LDA_04']
```

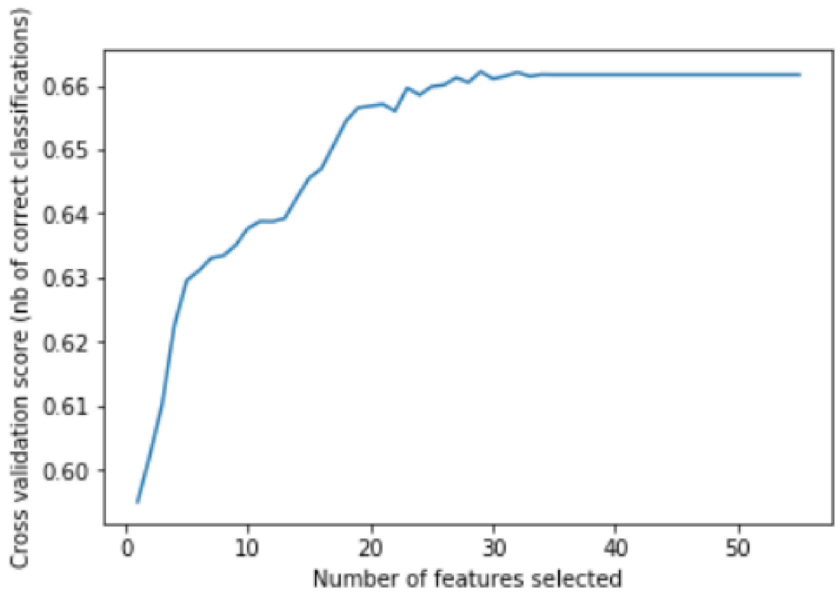
Random Forest:

Next, we run RFECV with a RF estimator. The cross-validation score versus the number of feature selected is shown in below fig. We find that RFECV selects 56 features for RF, which
is almost the full features in dataset. The only two features RFECV excludes for RF is ['n nonstop words', ' data channel is lifestyle'].



```
44
['n_tokens_title' 'n_tokens_content' 'n_unique_tokens' 'num_hrefs'
 'num_self_hrefs' 'num_imgs' 'num_videos' 'average_token_length'
 'num_keywords' 'data_channel_is_entertainment' 'data_channel_is_tech'
 'kw_min_min' 'kw_max_min' 'kw_min_max' 'kw_max_max' 'kw_avg_max'
 'kw_min_avg' 'kw_max_avg' 'kw_avg_avg' 'self_reference_min_shares'
 'self_reference_max_shares' 'self_reference_avg_sharess' 'is_weekend'
 'LDA_00' 'LDA_01' 'LDA_02' 'LDA_03' 'LDA_04' 'global_subjectivity'
 'global_sentiment_polarity' 'global_rate_positive_words'
 'global_rate_negative_words' 'rate_positive_words' 'rate_negative_words'
 'avg_positive_polarity' 'min_positive_polarity' 'max_positive_polarity'
 'avg_negative_polarity' 'min_negative_polarity' 'max_negative_polarity'
 'title_subjectivity' 'title_sentiment_polarity' 'abs_title_subjectivity'
 'abs_title_sentiment_polarity']
```

Adaptive Boosting:

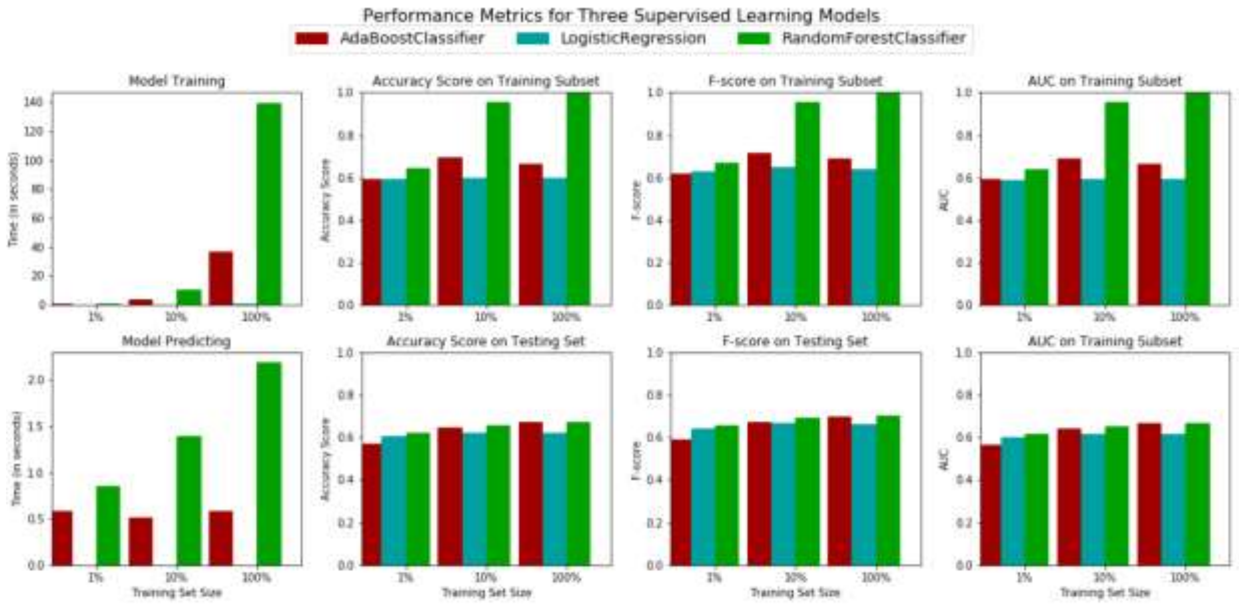


```
29
['n_unique_tokens' 'num_hrefs' 'num_self_hrefs' 'num_imgs' 'num_videos'
 'num_keywords' 'data_channel_is_entertainment' 'data_channel_is_socmed'
 'data_channel_is_tech' 'data_channel_is_world' 'kw_max_min' 'kw_min_max'
 'kw_max_max' 'kw_avg_max' 'kw_min_avg' 'kw_max_avg' 'kw_avg_avg'
 'self_reference_min_shares' 'self_reference_avg_sharess'
 'weekday_is_friday' 'is_weekend' 'LDA_00' 'LDA_01' 'global_subjectivity'
 'global_rate_positive_words' 'rate_negative_words'
 'avg_positive_polarity' 'min_positive_polarity' 'title_subjectivity']
```

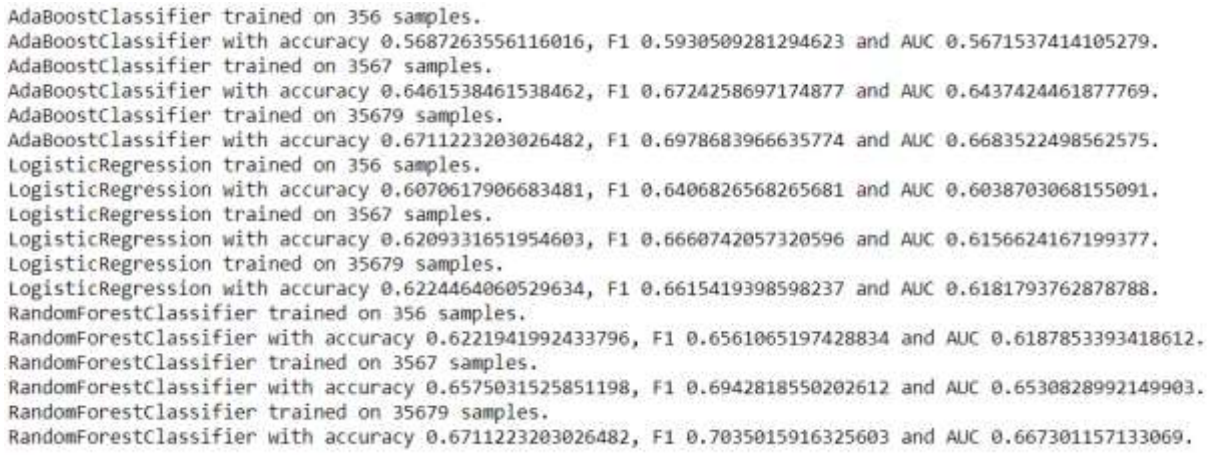
5.2 IMPLEMENTATION

5.2.1 MANUALLY CREATED CLUSTERS

Before algorithm implementation, for each algorithm, I also randomly split dataset with its
own selected features into training set (90%) and testing set (10%). The logistic regression,
RF and Adaboost are implemented by the sklearn function LogisticRegression(), Random-
ForestClassifier() and AdaBoostClassifier(), respectively. At this stage, I have used the default setting for model hyperparameters.



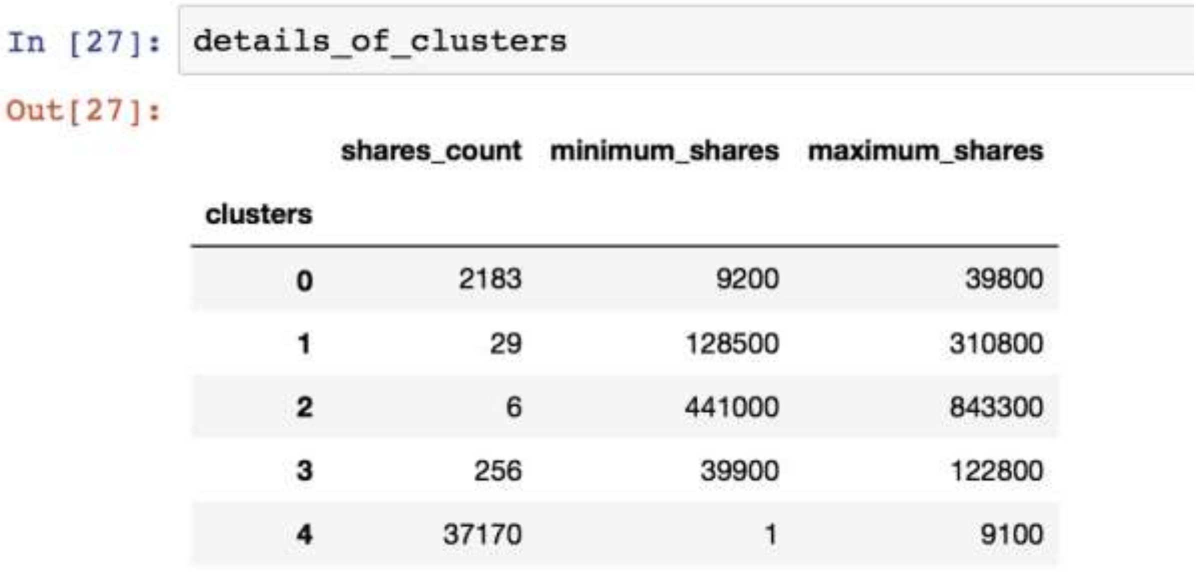
Evaluation Metrics:



```
AdaBoostClassifier trained on 356 samples.
AdaBoostClassifier with accuracy 0.5687263556116016, F1 0.5930509281294623 and AUC 0.5671537414105279.
AdaBoostClassifier trained on 3567 samples.
AdaBoostClassifier with accuracy 0.6461538461538462, F1 0.6724258697174877 and AUC 0.6437424461877769.
AdaBoostClassifier trained on 35679 samples.
AdaBoostClassifier with accuracy 0.6711223203026482, F1 0.6978683966635774 and AUC 0.6683522498562575.
LogisticRegression trained on 356 samples.
LogisticRegression with accuracy 0.6070617906683481, F1 0.6406826568265681 and AUC 0.6038703068155091.
LogisticRegression trained on 3567 samples.
LogisticRegression with accuracy 0.6209331651954603, F1 0.6660742057320596 and AUC 0.6156624167199377.
LogisticRegression trained on 35679 samples.
LogisticRegression with accuracy 0.6224464060529634, F1 0.6615419398598237 and AUC 0.6181793762878788.
RandomForestClassifier trained on 356 samples.
RandomForestClassifier with accuracy 0.6221941992433796, F1 0.6561065197428834 and AUC 0.6187853393418612.
RandomForestClassifier trained on 3567 samples.
RandomForestClassifier with accuracy 0.6575031525851198, F1 0.6942818550202612 and AUC 0.6530828992149903.
RandomForestClassifier trained on 35679 samples.
RandomForestClassifier with accuracy 0.6711223203026482, F1 0.7035015916325603 and AUC 0.667301157133069.
```

5.2.2 CLUSTERS CREATED USING K-MEANS

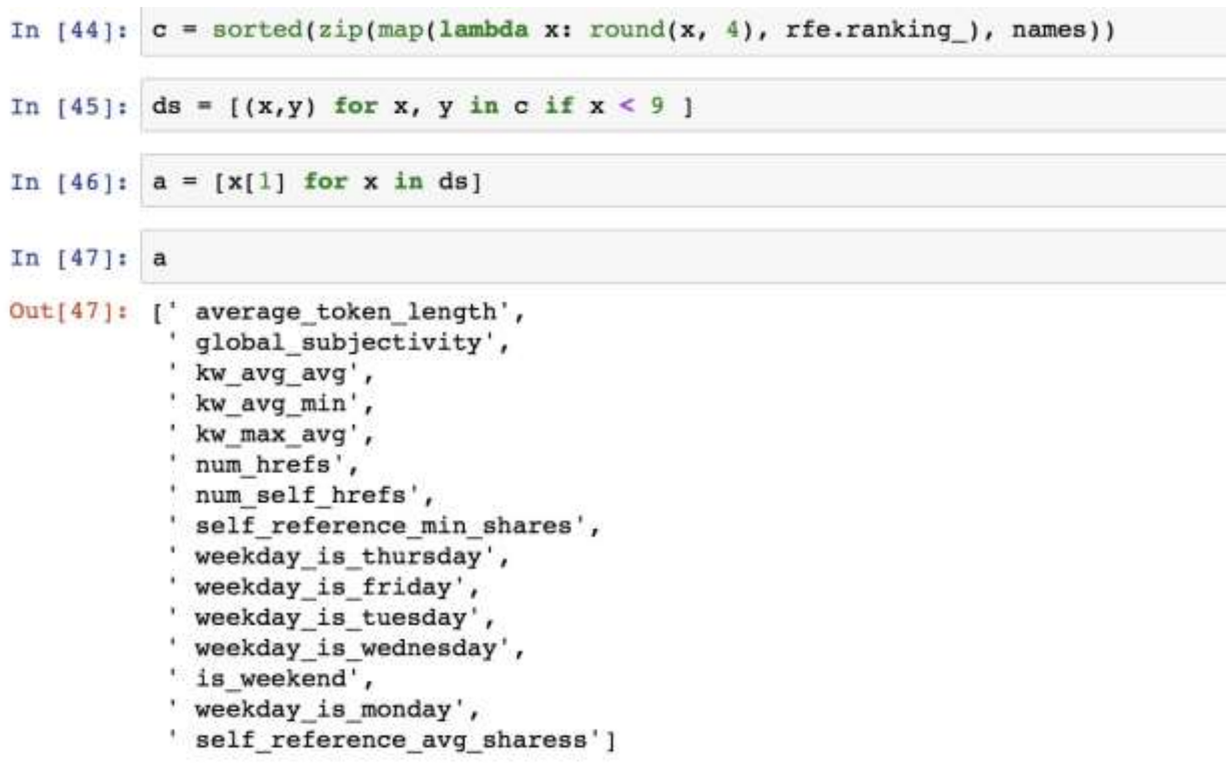
Cluster Description-
We got 5 clusters with the help of K-Means
Obscure- Obscure Cluster contains of minimum shares
Mediocre- Mediocre consist of shares higher than obscure but lower than Popular
Popular- It consists of shares that are popular.
Super Popular- They are less popular than viral
Viral- Viral cluster shows shares of online news which were viral



```
In [27]: details_of_clusters
Out[27]:
```

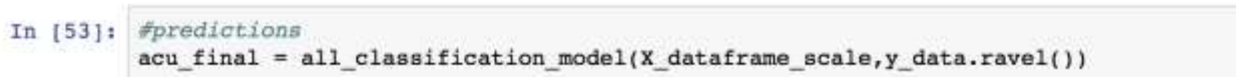
clusters	shares_count	minimum_shares	maximum_shares
0	2183	9200	39800
1	29	128500	310800
2	6	441000	843300
3	256	39900	122800
4	37170	1	9100

Best Feature Selected Using RFE

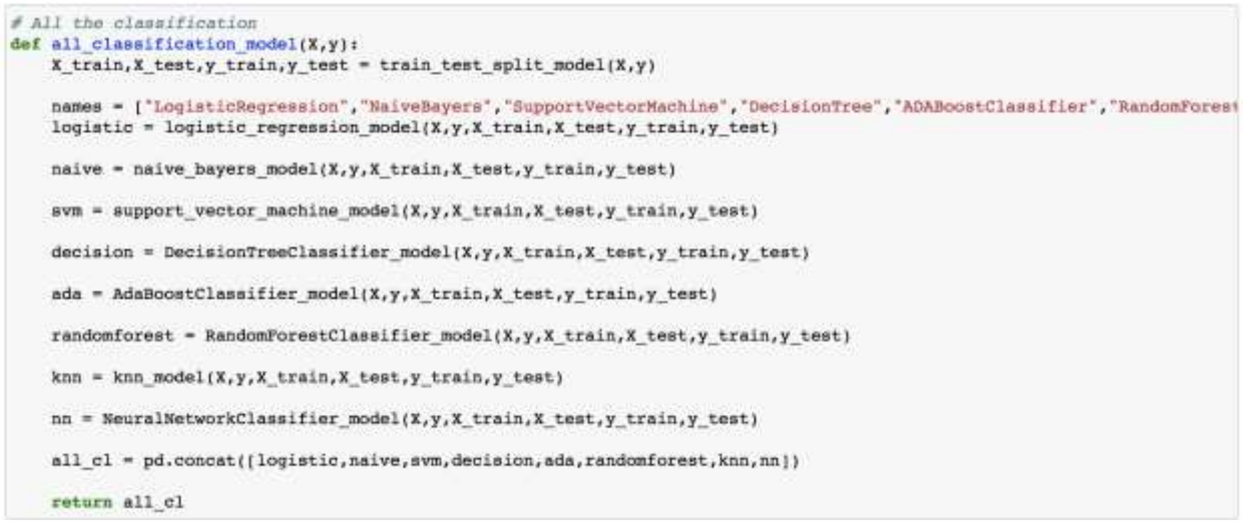


```
In [44]: c = sorted(zip(map(lambda x: round(x, 4), rfe.ranking_), names))

In [45]: ds = [(x,y) for x, y in c if x < 9 ]

In [46]: a = [x[1] for x in ds]

In [47]: a

Out[47]: [' average_token_length',
          ' global_subjectivity',
          ' kw_avg_avg',
          ' kw_avg_min',
          ' kw_max_avg',
          ' num_hrefs',
          ' num_self_hrefs',
          ' self_reference_min_shares',
          ' weekday_is_thursday',
          ' weekday_is_friday',
          ' weekday_is_tuesday',
          ' weekday_is_wednesday',
          ' is_weekend',
          ' weekday_is_monday',
          ' self_reference_avg_sharess']
```

Model Training & Testing-
We have used a single function for training and testing the models which calls each model recursively. The Precision, Recall, F1-Score, Accuracy are calculated with the help of classification report



```
In [53]: #predictions
         acu_final = all_classification_model(X_dataframe_scale,y_data.ravel())
```

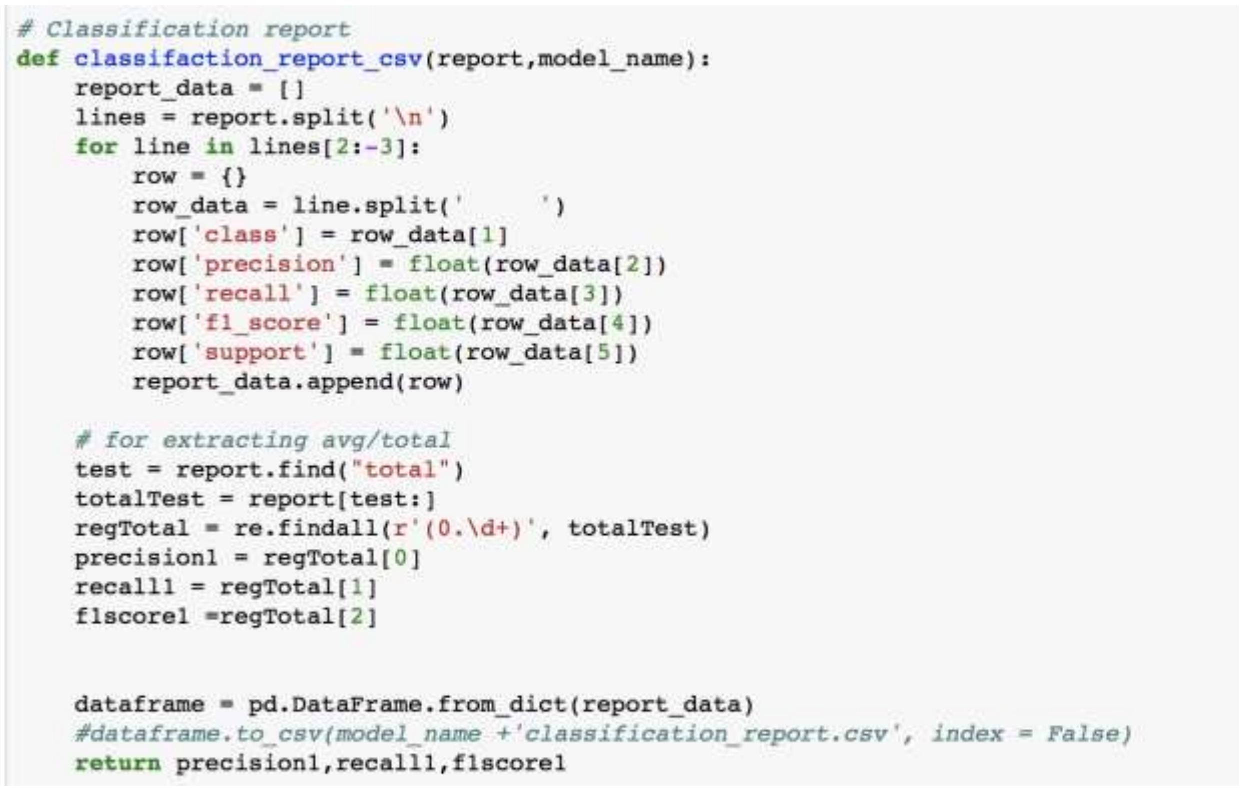


```
# All the classification
def all_classification_model(X,y):
    X_train,X_test,y_train,y_test = train_test_split_model(X,y)

    names = ["LogisticRegression","NaiveBayers","SupportVectorMachine","DecisionTree","ADABoostClassifier","RandomForest
    logistic = logistic_regression_model(X,y,X_train,X_test,y_train,y_test)

    naive = naive_bayers_model(X,y,X_train,X_test,y_train,y_test)

    svm = support_vector_machine_model(X,y,X_train,X_test,y_train,y_test)

    decision = DecisionTreeClassifier_model(X,y,X_train,X_test,y_train,y_test)

    ada = AdaBoostClassifier_model(X,y,X_train,X_test,y_train,y_test)

    randomforest = RandomForestClassifier_model(X,y,X_train,X_test,y_train,y_test)

    knn = knn_model(X,y,X_train,X_test,y_train,y_test)

    nn = NeuralNetworkClassifier_model(X,y,X_train,X_test,y_train,y_test)

    all_cl = pd.concat([logistic,naive,svm,decision,ada,randomforest,knn,nn])

    return all_cl
```

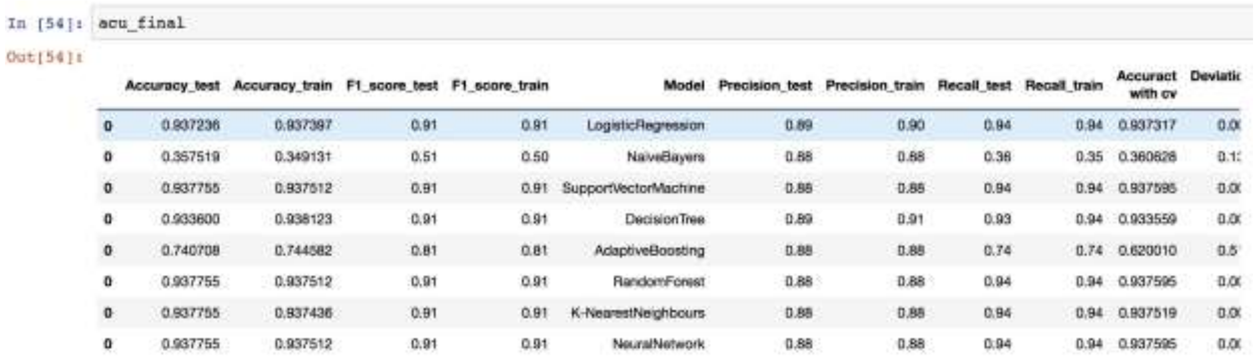
Accuracy: Accuracy = TP+TN/TP+FP+FN+TN

Precision: The question that this Precision metric answer is of all passengers that labeled as survived, how many actually survived?
Precision = TP/TP+FP
Recall: The question recall answer is: Of all the passengers that truly survived, how many did we label?
Recall = TP/TP+FN

F1 score: Accuracy works best if false positives and false negatives have similar cost. If the cost of false positives and false negatives are very different
F1 Score = 2*(Recall * Precision) / (Recall + Precision)



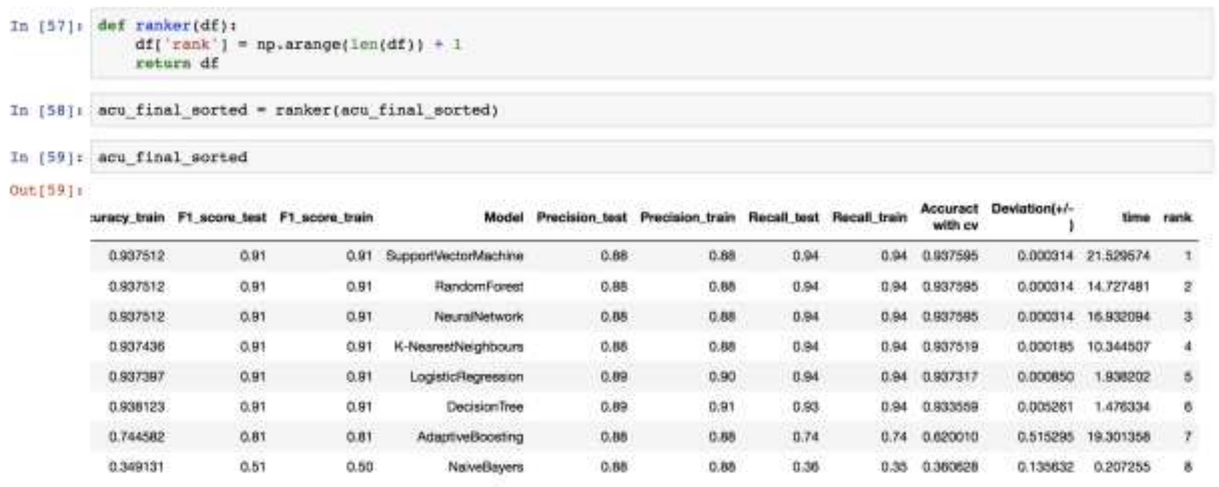
```
# Classification report
def classifaction_report_csv(report,model_name):
    report_data = []
    lines = report.split('\n')
    for line in lines[2:-3]:
        row = {}
        row_data = line.split('      ')
        row['class'] = row_data[1]
        row['precision'] = float(row_data[2])
        row['recall'] = float(row_data[3])
        row['f1_score'] = float(row_data[4])
        row['support'] = float(row_data[5])
        report_data.append(row)

    # for extracting avg/total
    test = report.find("total")
    totalTest = report[test:]
    regTotal = re.findall(r'(0.\d+)', totalTest)
    precision1 = regTotal[0]
    recall1 = regTotal[1]
    f1score1 =regTotal[2]


    dataframe = pd.DataFrame.from_dict(report_data)
    #dataframe.to_csv(model_name +'classification_report.csv', index = False)
    return precision1,recall1,f1score1
```



```
In [54]: acu_final
Out[54]:
```

	Accuracy_test	Accuracy_train	F1_score_test	F1_score_train	Model	Precision_test	Precision_train	Recall_test	Recall_train	Accuract with cv	Deviatic
0	0.937236	0.937397	0.91	0.91	LogisticRegression	0.89	0.90	0.94	0.94	0.937317	0.0
0	0.357519	0.349131	0.51	0.50	NaiveBayers	0.88	0.88	0.36	0.35	0.360628	0.1
0	0.937755	0.937512	0.91	0.91	SupportVectorMachine	0.88	0.88	0.94	0.94	0.937595	0.0
0	0.933600	0.938123	0.91	0.91	DecisionTree	0.89	0.91	0.93	0.94	0.933559	0.0
0	0.740708	0.744582	0.81	0.81	AdaptiveBoosting	0.88	0.88	0.74	0.74	0.620010	0.5
0	0.937755	0.937512	0.91	0.91	RandomForest	0.88	0.88	0.94	0.94	0.937595	0.0
0	0.937755	0.937436	0.91	0.91	K-NearestNeighbours	0.88	0.88	0.94	0.94	0.937519	0.0
0	0.937755	0.937512	0.91	0.91	NeuralNetwork	0.88	0.88	0.94	0.94	0.937595	0.0

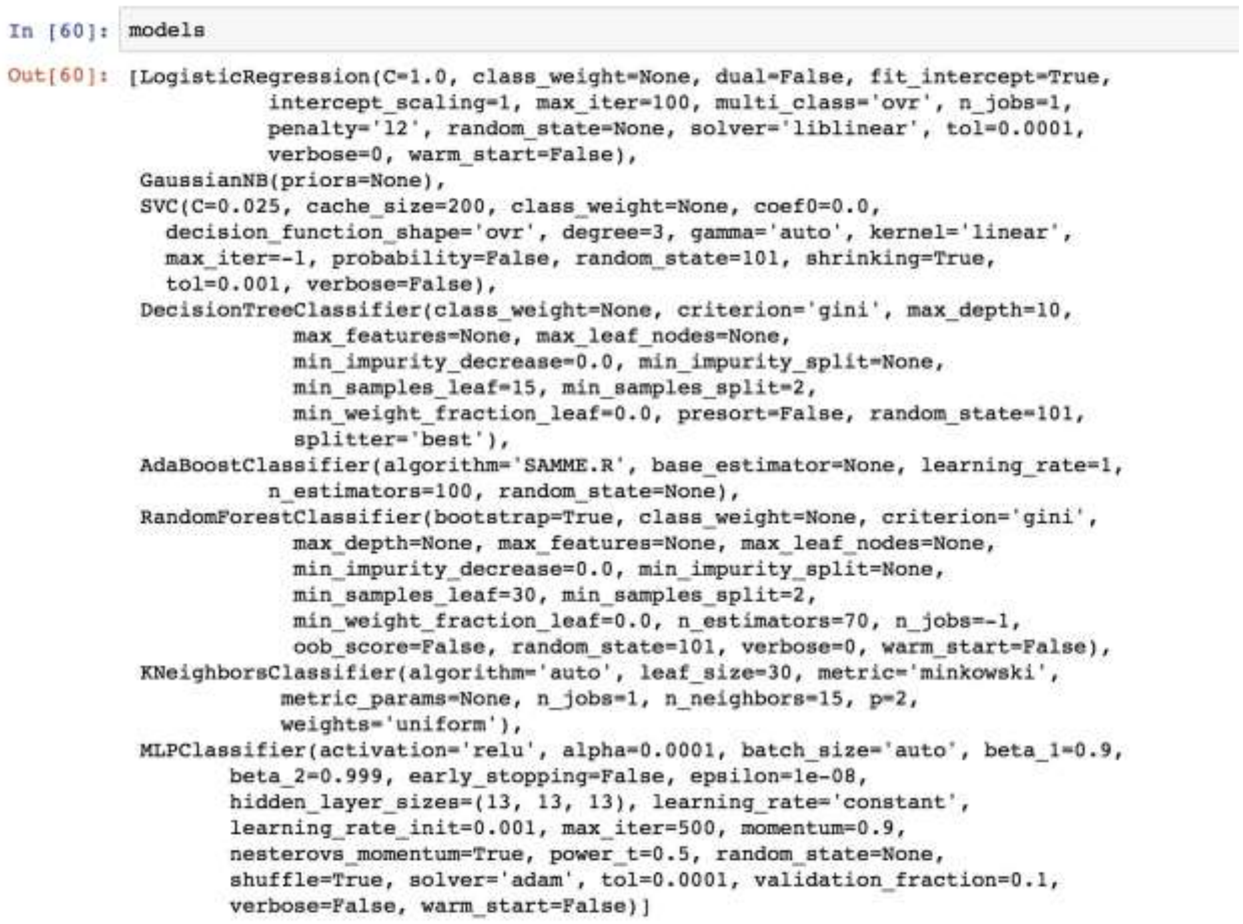
Ranking Of Models-



```
In [57]: def ranker(df):
             df['rank'] = np.arange(len(df)) + 1
             return df

In [58]: acu_final_sorted = ranker(acu_final_sorted)

In [59]: acu_final_sorted
Out[59]:
```

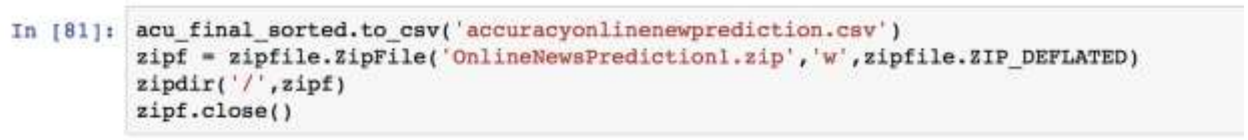
uracy_train	F1_score_test	F1_score_train	Model	Precision_test	Precision_train	Recall_test	Recall_train	Accuract with cv	Deviation(+/-)	time	rank
0.937512	0.91	0.91	SupportVectorMachine	0.88	0.88	0.94	0.94	0.937595	0.000314	21.529574	1
0.937512	0.91	0.91	RandomForest	0.88	0.88	0.94	0.94	0.937595	0.000314	14.727481	2
0.937512	0.91	0.91	NeuralNetwork	0.88	0.88	0.94	0.94	0.937595	0.000314	16.932094	3
0.937436	0.91	0.91	K-NearestNeighbours	0.88	0.88	0.94	0.94	0.937519	0.000185	10.344507	4
0.937387	0.91	0.91	LogisticRegression	0.89	0.90	0.94	0.94	0.937317	0.000850	1.938202	5
0.938123	0.91	0.91	DecisionTree	0.89	0.91	0.93	0.94	0.933559	0.005261	1.476334	6
0.744582	0.81	0.81	AdaptiveBoosting	0.88	0.88	0.74	0.74	0.620010	0.515295	19.301358	7
0.349131	0.51	0.50	NaiveBayers	0.88	0.88	0.36	0.35	0.360628	0.135632	0.207255	8

Pickled Models-



```
In [60]: models

Out[60]: [LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                    intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
                    penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
                    verbose=0, warm_start=False),
          GaussianNB(priors=None),
          SVC(C=0.025, cache_size=200, class_weight=None, coef0=0.0,
            decision_function_shape='ovr', degree=3, gamma='auto', kernel='linear',
            max_iter=-1, probability=False, random_state=101, shrinking=True,
            tol=0.001, verbose=False),
          DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=10,
                      max_features=None, max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=15, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, presort=False, random_state=101,
                      splitter='best'),
          AdaBoostClassifier(algorithm='SAMME.R', base_estimator=None, learning_rate=1,
                    n_estimators=100, random_state=None),
          RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                      max_depth=None, max_features=None, max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=30, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, n_estimators=70, n_jobs=-1,
                      oob_score=False, random_state=101, verbose=0, warm_start=False),
          KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=1, n_neighbors=15, p=2,
                     weights='uniform'),
          MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
                 beta_2=0.999, early_stopping=False, epsilon=1e-08,
                 hidden_layer_sizes=(13, 13, 13), learning_rate='constant',
                 learning_rate_init=0.001, max_iter=500, momentum=0.9,
                 nesterovs_momentum=True, power_t=0.5, random_state=None,
                 shuffle=True, solver='adam', tol=0.0001, validation_fraction=0.1,
                 verbose=False, warm_start=False)]
```

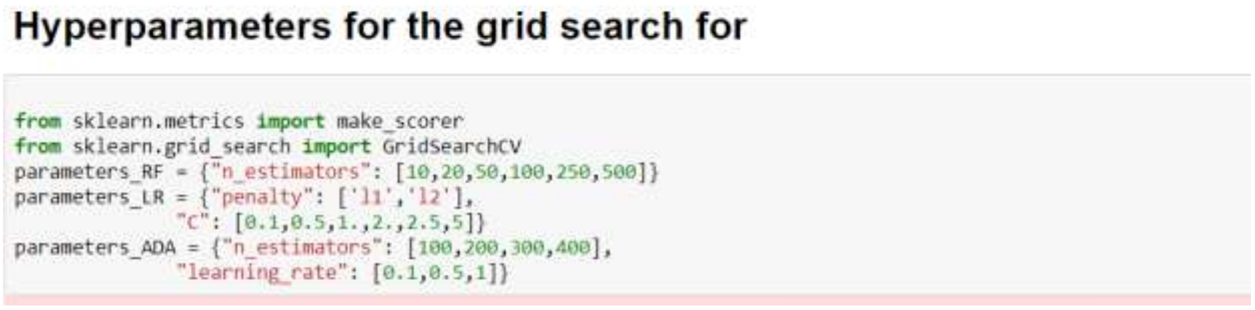
Pickled File Stored in Zip Format



```
In [81]: acu_final_sorted.to_csv('accuracyonlinenewprediction.csv')
         zipf = zipfile.ZipFile('OnlineNewsPrediction1.zip','w',zipfile.ZIP_DEFLATED)
         zipdir('/',zipf)
         zipf.close()
```

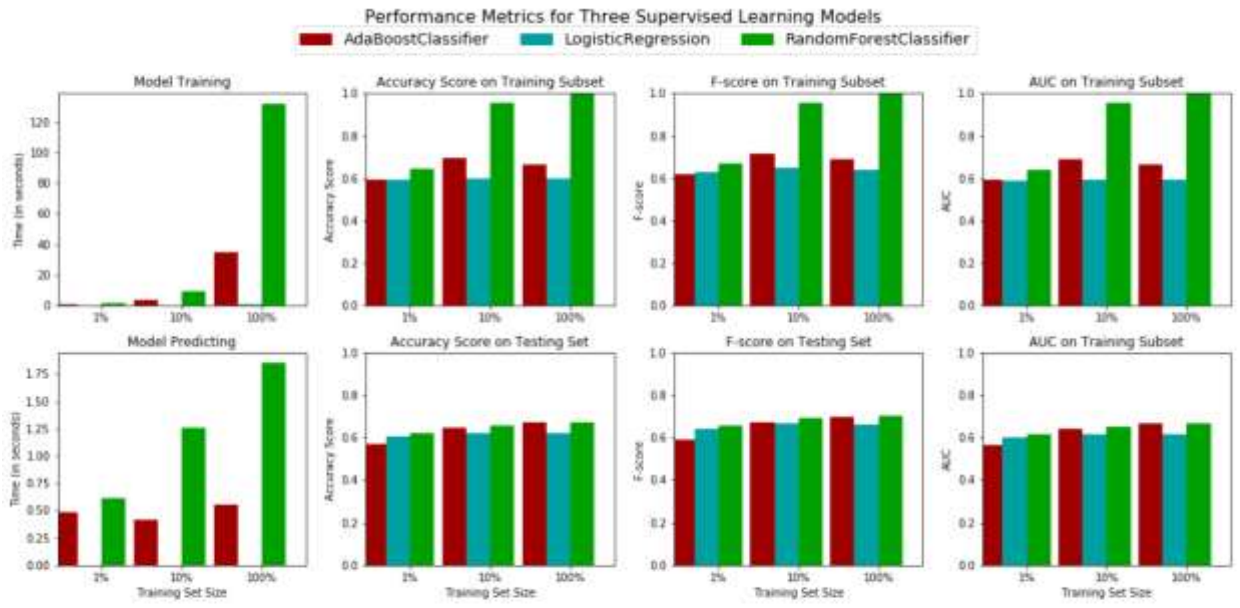
5.3 HYPER-PARAMETERS TUNING

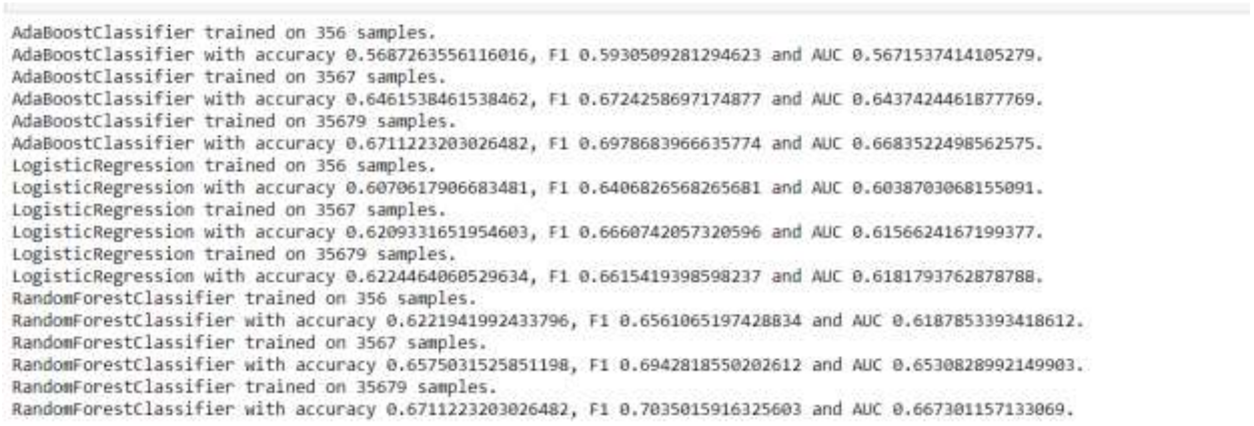
In this part, I have used the grid search method for each of the three classifiers to tune their hyperparameters. The grid search method exhaustively search through all possible combinations of model parameters, cross validate the model and then determine which set
of model parameters gives the best performance. Since the grid search can help select an
optimal model parameter, thus the learning algorithm can be optimized. In sklearn, we can
use the function GridSearchCV() to implement grid search.



Hyperparameters for the grid search for

```
from sklearn.metrics import make_scorer
from sklearn.grid_search import GridSearchCV
parameters_RF = {"n_estimators": [10,20,50,100,250,500]}
parameters_LR = {"penalty": ['l1','l2'],
                 "C": [0.1,0.5,1.,2.,2.5,5]}
parameters_ADA = {"n_estimators": [100,200,300,400],
                  "learning_rate": [0.1,0.5,1]}
```





```
AdaBoostClassifier trained on 356 samples.
AdaBoostClassifier with accuracy 0.5687263556116016, F1 0.5930509281294623 and AUC 0.5671537414105279.
AdaBoostClassifier trained on 3567 samples.
AdaBoostClassifier with accuracy 0.6461538461538462, F1 0.6724258697174877 and AUC 0.6437424461877769.
AdaBoostClassifier trained on 35679 samples.
AdaBoostClassifier with accuracy 0.6711223203026482, F1 0.6978683966635774 and AUC 0.6683522498562575.
LogisticRegression trained on 356 samples.
LogisticRegression with accuracy 0.6070617906683481, F1 0.6406826568265681 and AUC 0.6038703068155091.
LogisticRegression trained on 3567 samples.
LogisticRegression with accuracy 0.6209331651954603, F1 0.6660742057320596 and AUC 0.6156624167199377.
LogisticRegression trained on 35679 samples.
LogisticRegression with accuracy 0.6224464060529634, F1 0.6615419398598237 and AUC 0.6181793762878788.
RandomForestClassifier trained on 356 samples.
RandomForestClassifier with accuracy 0.6221941992433796, F1 0.6561065197428834 and AUC 0.6187853393418612.
RandomForestClassifier trained on 3567 samples.
RandomForestClassifier with accuracy 0.6575031525851198, F1 0.6942818550202612 and AUC 0.6530828992149903.
RandomForestClassifier trained on 35679 samples.
RandomForestClassifier with accuracy 0.6711223203026482, F1 0.7035015916325603 and AUC 0.667301157133069.
```

Compared with the metrics in fir above, we can find the metrics of logistic regression and Adaboost are just slightly improved, but the metrics of random forest are almost same after tuning by grid search.

6. SUMMARY

To summarize, the project is conducted through following steps:
(a) **Data collection:** The dataset of some 40,000 online news articles are downloaded from
UCI Machine Learning Repository, which is originally collected and donated by the author .

(b) **Data preprocessing**: Based on the initial data preprocessing in original dataset, I further preprocess the dataset by normalizing the numerical feature such that each feature is treated equally when applying supervised learning. I also select the median of the target attribute (number of shares) as an appropriate threshold to label all the data as either popular or unpopular.

(c) **Data exploration and visualization**: I explore the relevance of certain feature by visualization. And I also visualize the data distribution by PCA.

(d) **Feature selection:** To select the most relevant features among all 58 features, I use RFECV to select the most relevant features for each of the classifier.

(e) **Classifier implementation and hyperparamter tuning:** The classification algorithms including Logistic Regression, RF and Adaboost ,Support Vector Machine, Neural Networks, Decision Tree are implemented using sklearn. Then the model's hyperparameters are tuned by grid search method.

(f) **Model evaluation and validation**: The refined models are evaluated and compared using
three metrics (accuracy, F1-score, AUC).

7. CONCLUSION:

We have used **2 Approaches** for Online News Popularity-

1. **Manually Created Cluster-**
   Popular, Unpopular

2. **Creation of Cluster with the help of K-Means-**
   Obscure, Mediocre, Popular, Super Popular, Viral

With the help of manually created clusters we got the best Model as RandomForestClassifier
In K-Means Cluster we got the best Model as RandomForestClassifier:
F1-score = 0.91
Precision = 0.88
Recall as 0.94