Online News Popularity Prediction

G6

1. Problem Description:

In todays world, there is a very high reliance on online news for information among users. Some studies even show that the number could be as high as 62% of US adults relying on social media for news [1]. Social Media like Facebook, Twitter is often looked up as the primary source of finding new information. The social media has evolved into a system where the news/information part is given a high priority and has become its essential component (Eg. Trending in Facebook).

New businesses have been established in recent years centered around delivering news and information through the Internet and using social media as a platform for their promotion. This model is used for promotions, advertisements and thus, coming up with news which will be popular can help businesses reach their goals.

We viewed it as a classification problem, which when provided with details about the news will notify the user whether or not it will be popular. We used the data present in UCI repository [2]. We started with data analysis as our first step to get some insights about the data. Then we used Supervised Learning methods like Logistic Regression, SVM, Naive Bayes and Random Forest Algorithm to come up with the proposed prediction system. Random Forest was the best performing algorithm with accuracy of 67.18%. We also studied implications of unbalanced dataset.

2. Data Exploration:

For this project, we considered Online News Popularity [2] dataset present in UC Irvine Machine Learning Repository. The data set was gathered from articles published by Mashable.com, a well-known digital media website. The data consists of only some statistics about the news articles and not the content. However, the first attribute consists of the URL of the actual article and it can be accessed publicly to retrieve the article content as well for further analysis.

Overall, it has 39644 records, each having 61 attributes. Out of which, the last attribute is `Number of Shares` which is our target variable.

Following is the summary of our data analysis using visualizations and other functions like looking at the data summary, checking for missing values, outliers and correlations.

(a)**Response Variable:** In the data, response variable is continuous and has values in the range of 1 to 843300. As we are planning to perform classification on this dataset to identify popular/non-popular articles. We converted the response variable from continuous to binary using the threshold of 1400 (Median value), so that we get a balanced dataset in 1:1 ratio of each output class.

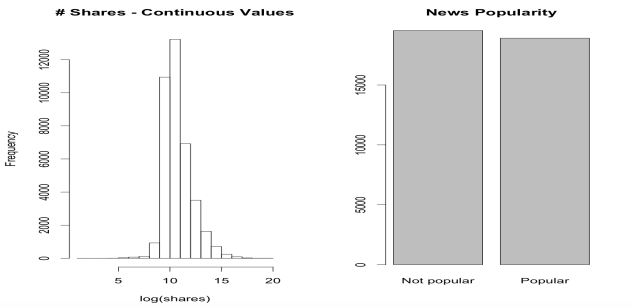

Figure 1: Response Variable Distribution

(b)**Correlations:** Since, the data had many predictors, we first tried to check the Information Value (IV) in each of the predictor w.r.t. response value (Figure 2). IV was computed as:

$IV = \sum(\%non-events - \%events) * WOE$

where, WOE (Weight of Evidence) = $ln(\frac{\%non-events}{\%events})$

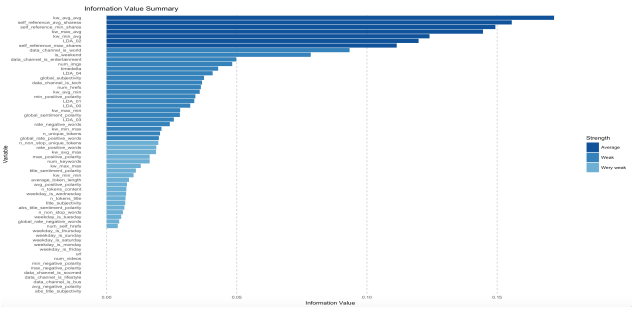

Figure 2: Information Value Summary

It is evident from the figure above that none of the predictors strongly correlate to the response variable. We also cross-checked these results by computing correlations using `cor()` function in R. Maximum correlation that we got is 0.1627 for `kw_avg_avg` (Average keyword for average shares) with shares. Dataset also contains binary predictors.

(c)**Missing Values:** In this dataset, **missing values** are encoded as 0 instead of `NA`. We found three such predictors containing missing values, i.e., `Average length of the words in the content`, sum of `Rate of positive words among non-neutral tokens` and `Rate of negative words among non-neutral tokens` where the value could not be 0, we removed those observations.

(d)**Outliers:** In order to find and clean the outliers, we created histogram and boxplot for each predictor separately as the scale was different for each of them.

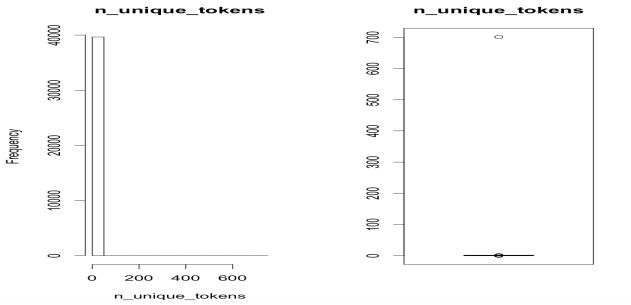

Figure 3: Outlier in Rate of unique words in the content

In the entire dataset, there was one noticeable observation for which `Rate of unique words in the content` had a value of 701 while other observations had values between 0 and 1. For the same observation, values in two other predictors (`Rate of non-stop words in the content` and `Rate of unique non-stop words in the content`) was also an outlier. So, we removed that observation. To study the impact of removing outliers, we trained a logistic regression model on all predictors with and without outliers. The performance of model without outliers was better than with outliers.

(e)**Multi-collinearity and Reduntant Predictors:** By looking at the data and headers, we observed some of the predictors were multi-collinear. For example, we have 7 binary columns for days of week where we can work with just 6 columns instead to avoid multi-collinearity. Apart from this, we also have another binary indicator for checking whether the publication day was a weekend or not. We further analyzed the effect of each day on the number of shares by visualizing the data (See Figure 4 and Figure 1) and also trying out logistic regression on All Predictors v/s Retaining 6 weekday columns v/s Retaining Is_weekend indicator only. Performance of model with Is_weekend was almost equal to model with 6 weekday variables. Also, from the Figure 4, it is evident that there is not much impact in number of shares on any particular weekday. So, for building all of the models, we retained only `Is_weekend` predictor and removed other 7 weekday predictors.

We even suspected presence of multi-collinearity in channels information, however, there were cases where all six columns had values zeroes, indicating a seventh channel and hence absence of collinearity. Hence, we were left with 52 predictors and 1 response variable.

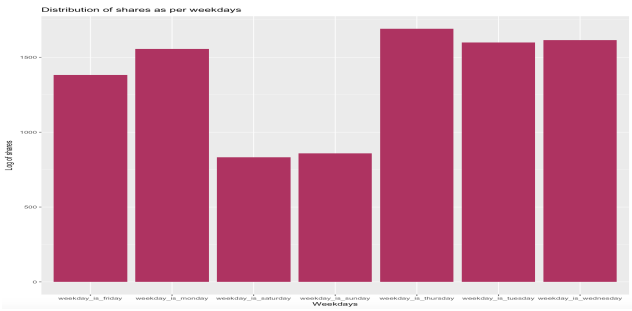

Figure 4: Distribution of shares as per weekdays

Also, as observed in Figure 1, the IV for `URL` predictor is 0 and it is of String data type. Since, all it's values are from same domain *(www.mashable.com)*, no useful information

could be obtained from it. Hence, we removed that predictor.

(f)**Data Sampling:** Since, this is a data-rich situation, we have split the dataset into 3 subsets of 50% training, 25% validation and 25% test set. We used training set for model building, validation for model selection and test set is used only once at the very end, to evaluate the performance of the final selected model.

3. **Methods:**

We have used several methods for classifying popular and non-popular news articles mainly to study the impact of different algorithms on our dataset and find out which one performs better. For each algorithm, we built models using remaining 52 predictors (after data cleanup) and also using top 30 predictors as per the Information Value w.r.t. response variable (Figure 1). We compared these models on the basis of prediction `Accuracy` and `AUC`. This will help us understand these different algorithms in a better way, as we have studied the theoretical aspect and this clarified its practical use.

(a)**Logistic Regression:**

We used Logistic Regression model to classify the binary output estimating the popularity of news. Logistic Regression hypothesis function is defined by sigmoid function

$$g(t) = \frac{exp(t)}{(1 + exp(t))}$$

where `t` is of the form of hypothesis function of linear regression.

Expectation value of output for logistic regression is given as

$$E\{Y\} = \frac{exp(\beta_0 + \beta_1 X_i)}{(1 + exp(\beta_0 + \beta_1 X_i))}.$$

We first trained a logistic model using all 52 and top 30 predictors to find the optimal value of β. We got the accuracy of 65.54% and 64.47% and AUC of 0.7049 and 0.6960 respectively. We also computed Variable Importance for these two models and it is shown in the following figure (Only for 52 predictors). Please refer Appendix if variable importance plots for other models are needed.

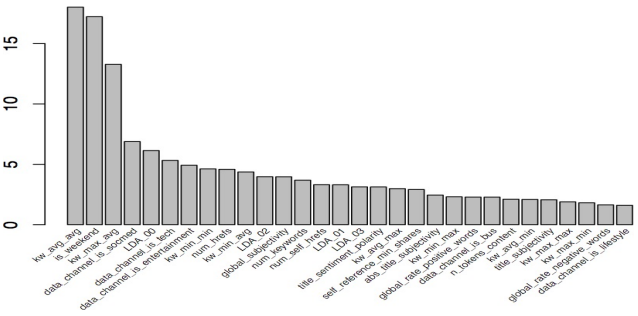

Figure 5: Variable Importance - Logistic Regression (All 52 predictors)

Then, to enhance the accuracy of model and for variable selection we implemented the lasso regularization model.

Lasso Regularization is defined to minimize the expression

$$\sum_{i=1}^{n}(Y_i - X_i\beta_i)^2 + \lambda \sum_{j=1}^{p} |\beta_i|$$

where λ value controls the extent of regularization. Lasso helps in shrinking the weights of parameters in hypothesis function and even shrinking it to zero for less important variables. We implemented the Lasso with value of lambda as sequence of values from $10e^{-2}$ to $10e^{10}$. Then, we implemented the cross validation for lasso to find the minimum lambda value.

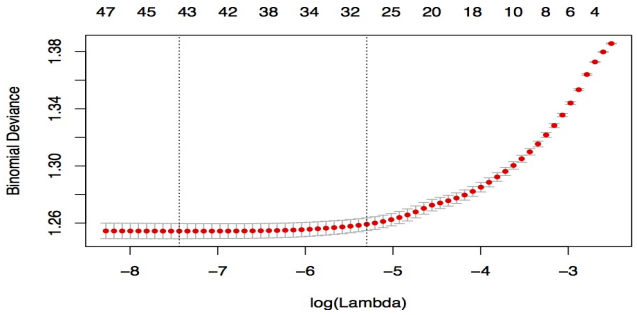

Figure 6: Lasso - Cross Validation plot

This model also performed variable selection and used 43 predictors. We got the accuracy of 65.48% and AUC of 0.7050 using the lasso regularization model.

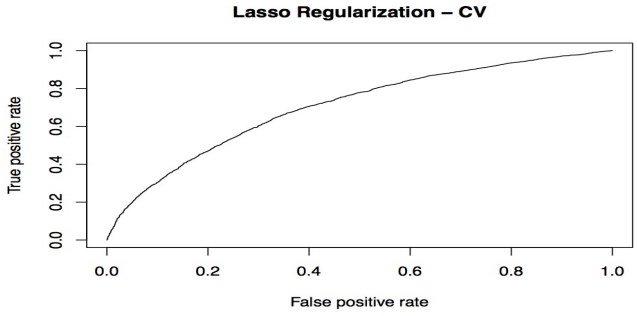

Figure 7: ROC Curve for Lasso (best model)

We even performed cross-validation on Logistic Regression for all 52 predictors which gave an accuracy of 65.54% and AUC of 0.7049.

(b)**Support Vector Machines (SVM):**
Given that we have non-linear higher dimensional data set under consideration, we implemented the Support Vector Machines with radial kernel. SVM with linear kernel doesnt perform well due to high bias. Also, as we had very large number of predictors and comparatively much lesser predictors, radial kernel became the suitable choice. [3]. For higher degree model, we implemented the SVM with polynomial of higher degrees starting from 3 to 8 but they didnt fit well on our data set.
Taking all the predictors and default value of gamma = 0.016 and cost(C) = 1, we got an AUC of 0.696 and accuracy of 64.1% on validation data. For comparison of kernels, we then tried with polynomial kernel with different degrees. We got best results with degree 3 in which case we got an accuracy of 64% and AUC of 0.69. As we increase the degree, the model seems to overfit the training data and performs poorly on validation data. Then, we opted to look for degree 1 as given by linear kernel. With linear kernel we got an accuracy of 60% and AUC of 0.69.
After hyper-parameter optimization for SVM, we got the best result with Radial kernel as expected for our dataset. After selecting the best kernel and parameters value for Radial kernel, we have implemented the SVM with all 52 and top 30 predictors. With 30 predictors, we got accuracy of 65.39% and AUC of 0.7026. With 52 predictors, we got better results with accuracy of 65.27% and AUC of 0.7100. We even tried finding the optimal parameters for Radial kernel by using cross-validation which gave us an accuracy of 64.55% and AUC of 0.6445.

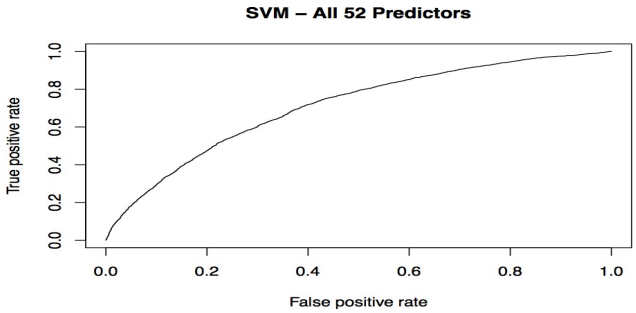

Figure 8: ROC Curve for best SVM Model

(c)**Random Forest Classifier:**
Random Forest Classifier is one of the simplest ensemble classifier to classify based on the output of several decision trees. It overcomes the problem of higher variance without increasing the bias as the output is based on several decision trees models.
We started with 1 decision based tree for all 52 predictors, for which we got the accuracy of 56.05% and AUC of 0.5603. As expected, we got poor performance of decision tree on the validation set due to overfitting. Random Forest Classifiers most significant parameter is number of trees, increasing the trees in forest increases the accuracy of classifier as it decreases the variance of model.
We increased the number of trees, starting with 500 to have number of decision trees suitable for our dataset. We implemented the RFC with 52 number of predictors sampled for splitting at each node and number of trees as 500, 800, 1200, 1500

Kernal	**Formula**	**Features**
Radial	$exp(-\gamma * \|x_1 - x_2\|^2)$	$\gamma = 0.016$
Polynomial	$(\gamma * x_1' * x_2 + const)^{degree}$	$degree = 3$
Linear	$\gamma * x_1' * x_2 + const$	

Table 1: SVM Models

and 1800. There is not much increase in accuracy beyond 1200 trees. So we stopped at 1800 trees and getting the best accuracy of 66.8% and AUC of 0.7286. We also converted the channel data into a categorical variable, giving 7 categories to train one of the models. This model performed almost equally as with the best model giving an accuracy of 66.62% and AUC of 0.7271.

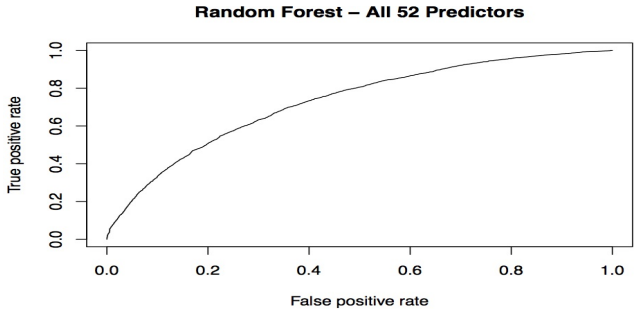

Figure 9: ROC Curve for best RF Model

Apart from finding ROC, we computed variable importance for RF model based on Mean Decrease in Accuracy and also Mean Decrease in Gini Value.

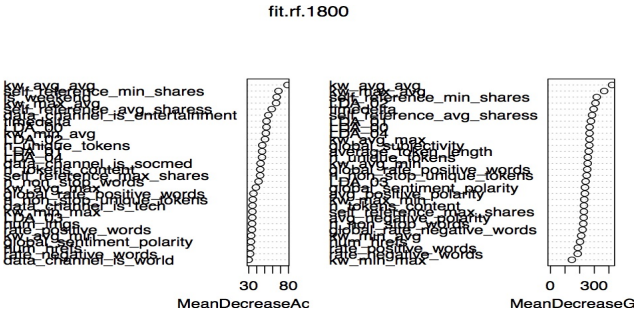

Figure 10: Variable Importance for best RF Model

(d)**Naïve Bayes Classifier:**
Naïve Bayes Algorithm is simplest classifier with an assumption of independence among the features of data set. This assumption doesnt hold practically but even assuming it over dependent feature set turns it out to perform as a good classifier in many of the cases. Naïve Bayes can be stated as following probabilistic formula

$$p(C_k|x) = \frac{p(C_k) * p(x|C_k)}{p(x)}$$

where we calculate posterior on the basis of prior, likelihood and evidence. We built a naïve bayes model with laplace smoothing as 0.01 for all 52 predictors and got an accuracy of 61.1% and AUC of 0.6059. The laplace smoothing was added to account for probabilities which were not seen during training but could be encountered upon validation and test. Other models of Naive Bayes had worse or same performance than the above model. Naive Bayes had the worse performance in comparison to other techniques applied.

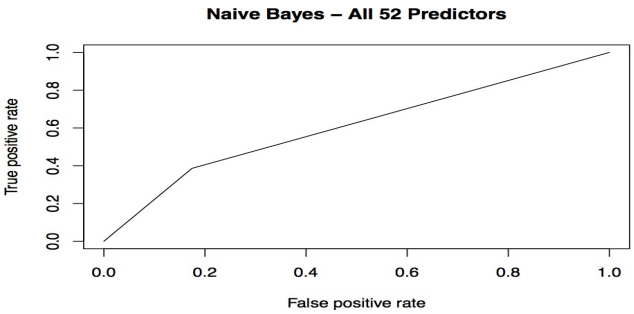

Figure 11: ROC Curve for best Naive Bayes Model

(e) **Artificially unbalancing data:**
To artificially unbalance data, news data was split at 95th percentile of target variable (shares). The shares above 95th percentile were considered as popular news and below 95th percentile were considered as unpopular. This resulted in highly unbalanced data where unpopular news forms the majority class. This data was then split into training, validation and test sets using 50 : 25 : 25 split. Upon training a logistic regression classifier using the training set of unbalanced data, the resultant classifier was highly biased towards the majority class. This is a major problem in Machine learning field where having unbalanced dataset results in degenerate models which do not account for the minority class.
Validation on unbalanced validation set:
Upon validating the results by using the unbalanced validation set, the accuracy observed on validation set was high (94.75%), however this indication is wrong as the accuracy is a result of the bias in dataset and classifier. The confusion matrix gives a clear indication of the bias in the classifier as there are barely any predictions of minority class, even though the validation set contains about 5% of popular news:

	Reference	
Prediction	0	1
0	9081	498
1	5	1

Table 2: Confusion Matrix - unbalanced validation set

The AUC, 0.7087, comparatively gave a better metric than being as misleading as prediction accuracy.

Validation on balanced validation set:

To balance the validation set, the majority class was under-sampled. The number of records present as popular news media (minority class) was recorded. The same number of records as present in popular news media were randomly sampled from the unpopular news media samples to obtain balanced data in a ratio of 1 : 1.

Upon applying the same classifier trained using unbalanced dataset, the bias in prediction was still evident and prediction accuracy became worse. The confusion matrix in a balanced validation set gives a much clearer picture of the bias than the case above as both classes are present in 1 : 1 split but the predictions for majority class are in more than 99%. The accuracy on validation set was brought down to a mere 49.39% showing how poor the classifier is.

	Reference	
Prediction	0	1
0	485	498
1	0	1

Table 3: Confusion Matrix - balanced validation set

Working with unbalanced dataset:

Unbalanced datasets can often be encountered in real world and we should be careful while working with datasets so that we do not end up with a highly biased classifier towards the majority class that the classifier was trained upon. For the starting point, we have used 1 : 19 split as discussed above between popular and unpopular news media.

We used simple under-sampling approach to balance the training dataset. Equal number of samples to the samples present in minority class were randomly chosen from the majority class in training set. This gave a 1 : 1 split in the resultant dataset between majority and minority classes. This way, we artificially rebalanced the training dataset. Logistic Regression classifier was trained on the rebalanced training set.

Validation on unbalanced validation set:

Upon validating the results by using the unbalanced validation set, the bias towards majority class seemed to be missing. The number of predictions were more for unpopular news as compared to popular news, but that is mainly because the validation set has 95% of unpopular news. In fact, there seems to be little bias favoring popular news, given that there were only 5% of records in the validation set for popular news but a much higher percentage of predictions for popular news. The accuracy observed on the validation set was about 71.25%. The confusion matrix gives a clear indication of the bias missing for majority class (unpopular news):

	Reference	
Prediction	0	1
0	6490	169
1	2576	312

Table 4: Confusion Matrix - unbalanced set

Validation on balanced validation set:

To balance the validation set, the majority class (unpopular news) in validation set was under-sampled. The same number of records as present in popular news media were randomly sampled from the unpopular news media samples to obtain balanced data in a ratio of 1 : 1.

Upon applying the same classifier trained using balanced dataset by simple under-sampling, the bias in prediction was missing and model seemed to give out even number of predictions for both popular and unpopular news. The confusion matrix in a balanced validation set gives a much clearer picture of the missing bias for majority class.

	Reference	
Prediction	0	1
0	331	169
1	133	312

Table 5: Confusion Matrix - unbalanced set

The accuracy of the model wasnt altered much and it still achieved an accuracy of 64.8% on the validation set.
Using the techniques discussed here, we can work with unbalanced dataset in the real world, and not have a classifier which is highly biased towards the majority class.

4. **Results:**

We considered 4 different machine learning models for our analysis. We explored these algorithms in depth and used them in multiple ways to understand the algorithms better and as well try to improve our results, in terms of AUC. We used, statistical regularization, choosing limited set of predictors, tuning (hyper-parameter optimization) where applicable in our implementations. From our models, Random Forest with all 52 predictors information was the best performing model on our validation set, having an accuracy of 66.8% and AUC of 0.7286. After using **test set** on this model, we got an **accuracy of 67.18%** and **AUC of 0.7382**.
We have summarized all the results in Table 6 and Table 7. Also, Figure 12 gives visual outlook of the model performance on validation set in terms of AUC.

Algorithm	**AUC**	**Accuracy**
Random Forest	0.7382	67.18%

Table 7: Best Model Prediction on test set

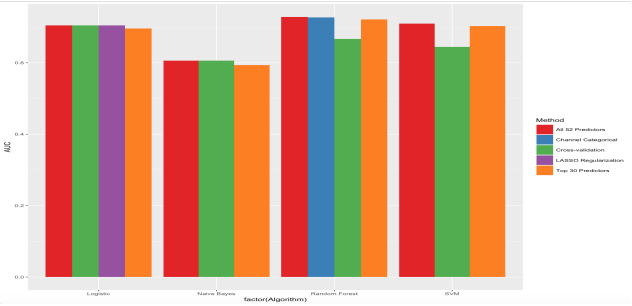

Figure 12: AUC plot for all models

Use of statistical regularization (Lasso) in logistic regression helped improve the model AUC and

Algorithm	**Method**	**AUC**	**Accuracy**
Logistic Regression	All 52 Predictors	0.7049	65.54%
	Top 30 Predictors	0.6960	64.47%
	LASSO Regularization	0.7050	65.48%
	Cross-validation	0.7049	65.54%
Support Vector Machines	All 52 Predictors	0.7100	65.27%
	Top 30 Predictors	0.7026	65.39%
	Cross-validation	0.6445	64.55%
Random Forest Classifier	**All 52 Predictors**	**0.7286**	**66.80%**
	Top 30 Predictors	0.7217	66.09%
	Channel as Categorical	0.7271	66.62%
	Cross-validation	0.6667	66.68%
Naive Bayes	All 52 Predictors	0.6059	61.10%
	Top 30 Predictors	0.5935	59.96%
	Cross-validation	0.6059	61.10%

Table 6: Results - using validation set

accuracy. This perhaps was due to the reason of bias addition which Lasso does.
The SVM model was built with radial kernel and tuned for hyper parameter optimization using cross validation. In SVM, this model delivered acceptable performance.
Naïve Bayes model did not do as well in comparison to other classifiers. This was perhaps due to the reason that Naïve Bayes makes an assumption of independence among all the predictors and needs apriori probability values for all, which wasnt suitable for our dataset.
The best performing classifier turned out to be Random Forest Classifier. The ensemble model made with 1800 trees and all 52 predictors gave the best results.
We also learned the effects of artificially unbalancing the dataset and the bias addition it makes for the majority class. It also gave an opportunity to explore on how to analyze and process such an unbalanced dataset, which we accommodated using simple under-sampling. We used logistic regression model for this purpose and learned of two more techniques, SMOTE and ADASYN, however, we did not implement the other two.

5. **Discussion:**
Real world data is prone to errors and an initial analysis is necessary to find the problems with the dataset. There could be outliers, missing data which should be identified and handled before applying any machine learning techniques. The data could have redundant information, collinearity etc and these should be handled based on the techniques being applied. We learned that real world data could have a lot of predictors making techniques like all subset selection not easy. Real world data can be imbalanced. Although we initially took balanced dataset for analysis, we even experimented with the dataset to learn what happens with unbalanced datasets and learned how such an unbalanced dataset could be used. We learned how to identify the settings to be used for different machine learning techniques regarding the real-world data and analyze these different techniques.
In the future, features could be extracted from the url and used in the classifier. The current data contains information about articles from Mashable and multiple sources could be used to collect data and then analyze them. An important parameter these days is presence of viral words in the article which boost the popularity of it. No such feature was observed in the current dataset and such data can be collected from the articles and a functionality can be built around it to help in deciding popularity of an article. We have also not tried a neural network and that is something which could be done to see how it will perform. Only undersampling technique was explored in case of unbalanced dataset. Oversampling could be explored for unbalanced dataset and techniques like SMOTE, ADASYN could be applied.

References:

1. TechCrunch - 62 percent of U.S. adults get their news from social media, says report
2. K. Fernandes, P. Vinagre and P. Cortez. A Proactive Intelligent Decision Support System for Predicting the Popularity of Online News. Proceedings of the 17th EPIA 2015 - Portuguese Conference on Artificial Intelligence, September, Coimbra, Portugal.
3. Linear kernel and non-linear kernel for support vector machine

Appendix:

Note: Please refer the Appendix file shared separately.

Statement of Contributions:

1. **Akash Singh:**
   Akash paired with Surekha to work on Data Analysis and preprocessing portion of the project. He paired with Gautam to identify the effects of artificially unbalancing dataset, as well as the portion of working with unbalanced dataset by using simple under-sampling. Moreover, he worked on Naïve Bayes classifier for the project.

2. **Gautam Vashisht:**
   Gautam paired with Surekha to work on Logistic Regression, which includes statistical regularization (Lasso) and cross validation. He paired with Akash to identify the effects of artificially unbalancing dataset, as well as the portion of working with unbalanced dataset by using simple undersampling. Morever, he worked on SVM classifier for the project.

3. **Surekha Jadhwani:**
   Surekha paired with Akash to work on Data Analysis and preprocessing portion of the project. She paired with Gautam to work on Logistic Regression, which includes statistical regularization (Lasso) and cross validation. Moreover, she worked on Random Forest classifier for the project.

***** Results, report and code integration were worked on by all three team members.