MSc in Business Analytics

Course: Data Management & Business Intelligence

Associate Professor Y.Kotidis

Supervisor: Y.Filippidou

DATA MINING TECHNIQUES

GROUP ASSIGNMENT 1

Chatzimoschou angeliki

baptXXXX

melenikou galateia

bapTXXXX

november 2016

# TABLE OF CONTENTS

[TABLE OF CONTENTS 2](#_Toc467074496)

[PART I – DATA REPRESANTATION AND EXPLORATION 3](#_Toc467074497)

[PART II – DATA PREPROCESSING AND TRANFORMATION 4](#_Toc467074498)

[PART III – DATA MINING TECHNIQUES 8](#_Toc467074499)

[CLASSIFICATION METHODS 8](#_Toc467074500)

[CLUSTERING 12](#_Toc467074501)

[PART IV – FEATURE SELECTION 14](#_Toc467074502)

[PART V - CONCLUSION 15](#_Toc467074503)

[REFERENCES 16](#_Toc467074504)

# PART I – DATA REPRESANTATION AND EXPLORATION

In this project, we are assigned to deploy several Data Mining Techniques on the “Online News Popularity” dataset provided by UCI machine learning repository. This dataset summarizes a heterogeneous set of features about articles published by Mashable in a period of two years.

Specifically, this dataset extracts 59 attributes (as numerical values) describing different aspects of each article, from a total of 39.644. It is composed by 58 independent variables, such as: the URL of the article, the number of keywords of specific types, what day of the week the article was published, the results of a Latent Dirichlet Allocation algorithm etc. and one dependent variable: the number of shares of the article. The full feature set can be mainly categorized as in Table 1.

|  |  |
| --- | --- |
| Aspects | Features |
| Words | Number of words of the title/content;  Average word length;  Rate of unique/non-stop words of contents |
| Links | Number of links;  Number of links to other articles in Mashable |
| Digital Media | Number of images/videos |
| Publication Time | Day of the week/weekend |
| Keywords | Number of keywords;  Worst/best/average keywords (#shares);  Article category |
| NLP | Closeness to five LDA topics;  Title/Text polarity/subjectivity;  Rate and polarity of positive/negative words; Absolute subjectivity/polarity level |
| Target | Number of shares at Mashable |

*Table 1 – All features*

The goal is to predict the number of shares in social networks (popularity) in order to determine which new articles will become popular or even viral and which new articles will be mostly ignored by the general public. Therefore, the goal of our analysis is to improve the generalizability of the classifier.

# PART II – DATA PREPROCESSING AND TRANFORMATION

To accomplish our analysis, we used the following tools:

* R Studio version 0.99.891
* R version 3.2.3
* Weka version 3.8

For the stage of preprocessing, we deployed various techniques using the tool R Studio. More specifically, we started the data preprocessing in order to examine at a deeper level the dataset. No missing values are found and no significant noise, except for one entry. The 31038th record, about “Ukraine Civilian Convoy Attacked” (from the URL) has values that are impossible to achieve across many variables, such as a rate of 700 in keywords where all other values are less than 1, so we decided to omit this instance.

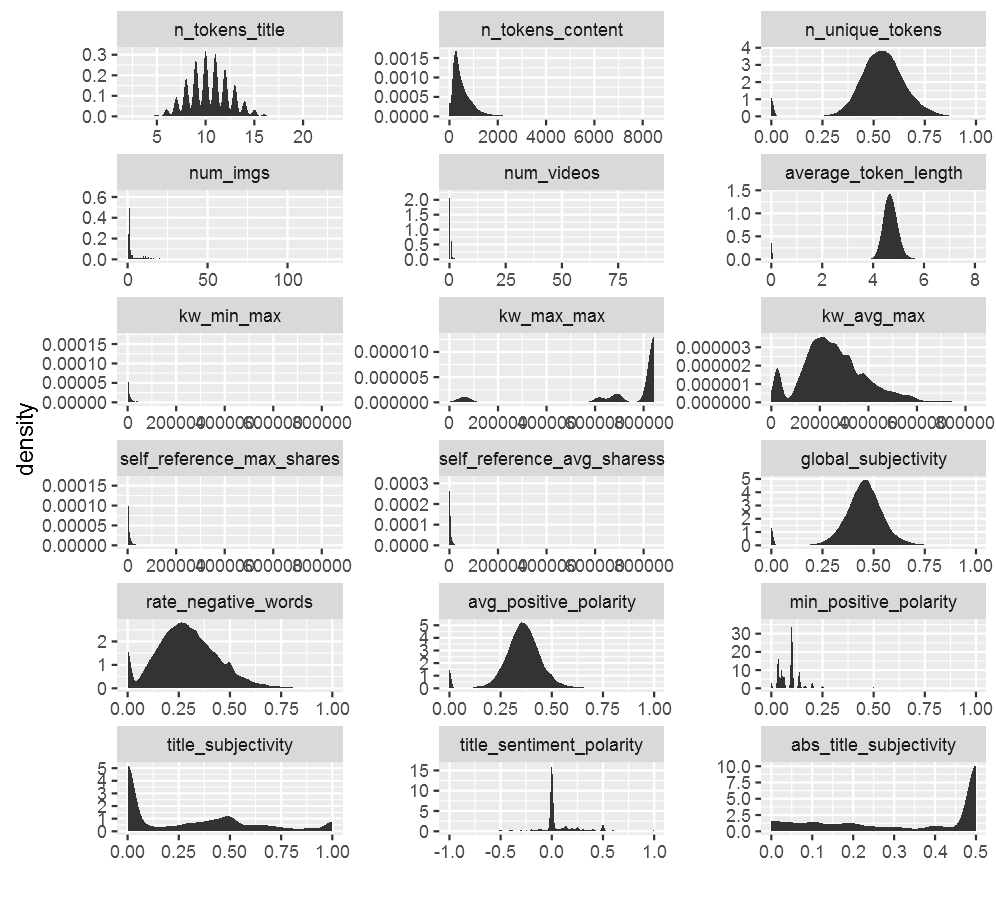
In addition, we decided to omit several attributes: five LDA variables and the non-predictive: URL (which included the title and date) and time delta variable. By observing the histogram of each feature (Figure 1-2), the numeric variables that varied in scale beyond [0,1] were standardized and also the ones with skewness > 30 were transformed using the log base 10 function.

Furthermore, we examined a core objective of data mining process, the inter-attribute correlations. Correlations are not only necessary for predictions and classifications – since rules would fail in the absence of pattern – but also for the identification of groups of mutually correlated attributes. This expedites the selection of a representative subset of attributes, from which existing mappings allow others to be derived. Actually, this dataset is well designed and correlated information between features is limited, as shown in Figure 3, where dark blue color indicates high positive correlation among a pair of features and red high negative correlation respectively. As we see, only a very small percentage of the dataset is highly correlated, thus there is no need to perform any dimension reduction technique, like PCA or SVD. This is also proved in Part IV.

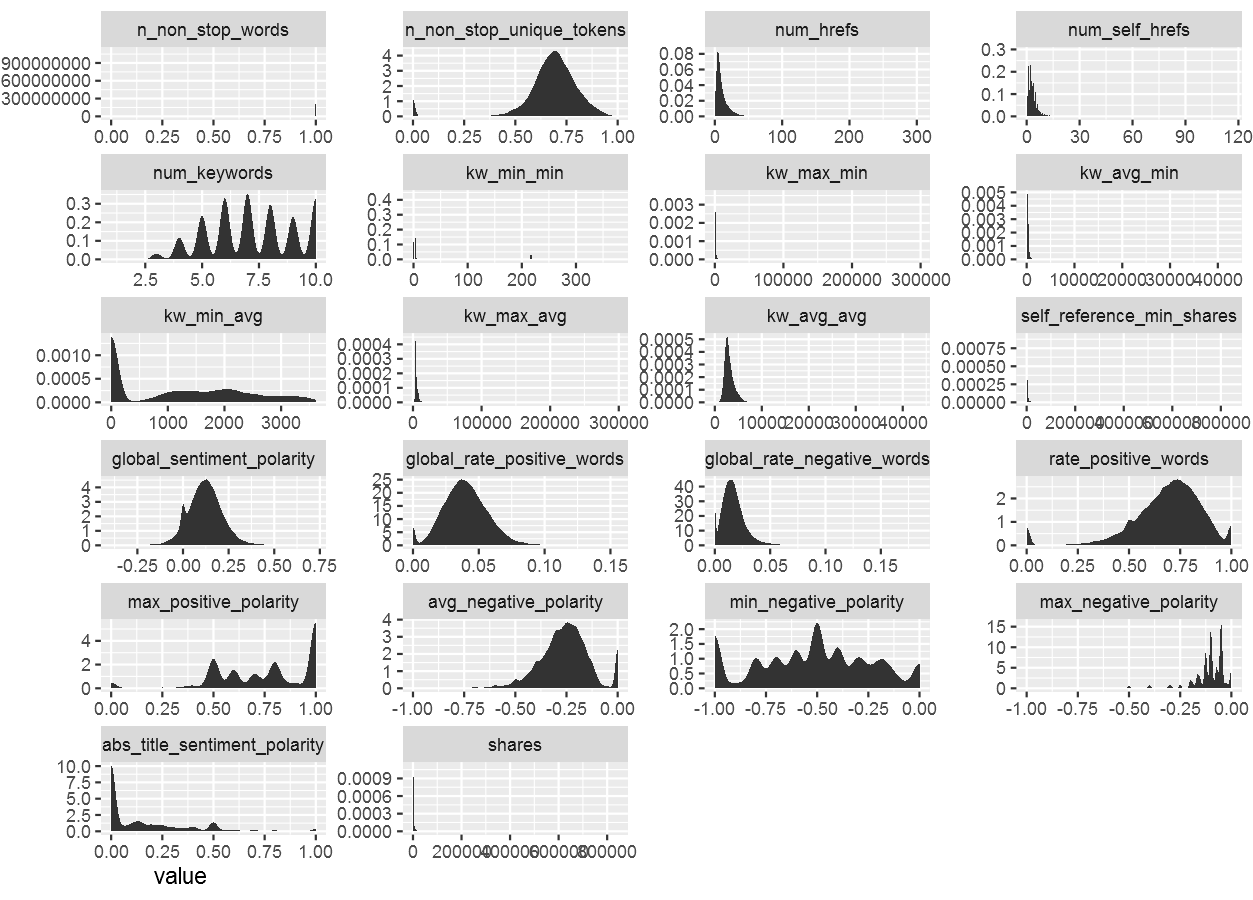
Our class variable, the number of shares, is a metric that defines how often an article is shared on social media. It is a continuous variable and we are interested, primarily, in articles with high popularity as our positive class, more so than articles with low popularity. Thus, in order to perform clustering and binary classification methods, we discretized our class variable: “shares” to binary categories. We used the median value of the shares: 1400 as the cut-off value, as shown in Table 2 below.

|  |  |  |
| --- | --- | --- |
| Num. of shares | 0-1400 | >1400 |
|  | Non- popular | Popular |
| 0 | 1 |

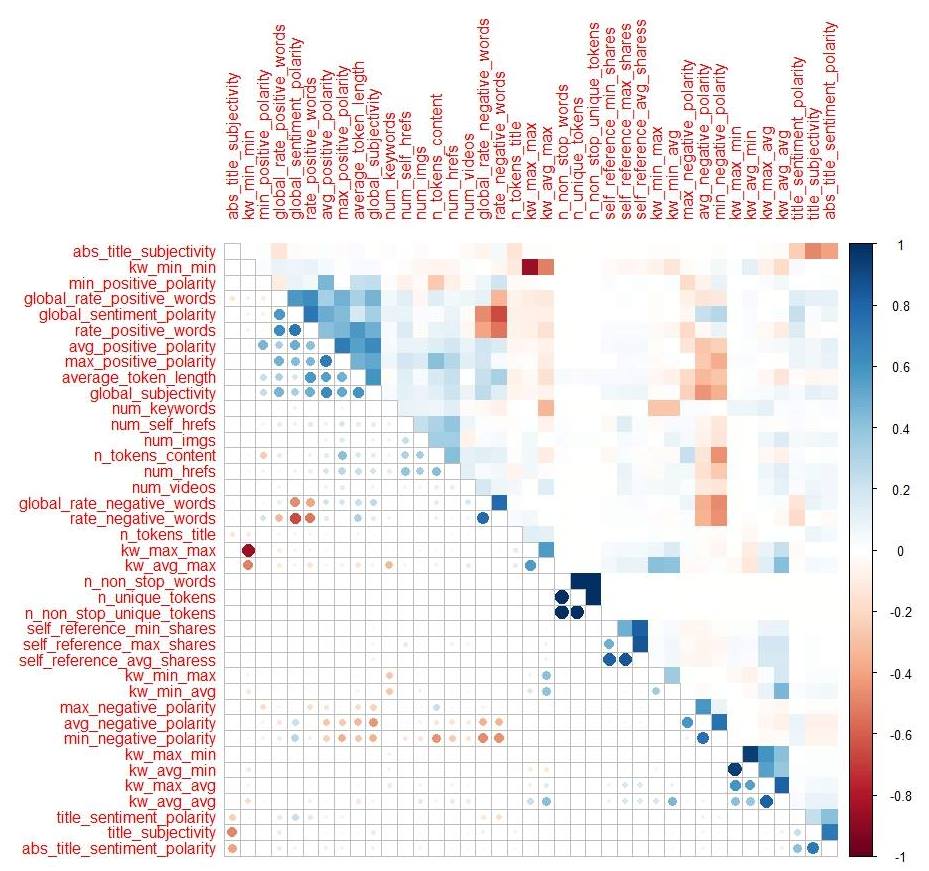
*Table 2 – Class variable formation*



*Figure 1 – Multiple histograms of raw dataset*



*Figure 2 – Multiple histograms of raw dataset*



*Figure 3 – Correlation plot of features*

# PART III – DATA MINING TECHNIQUES

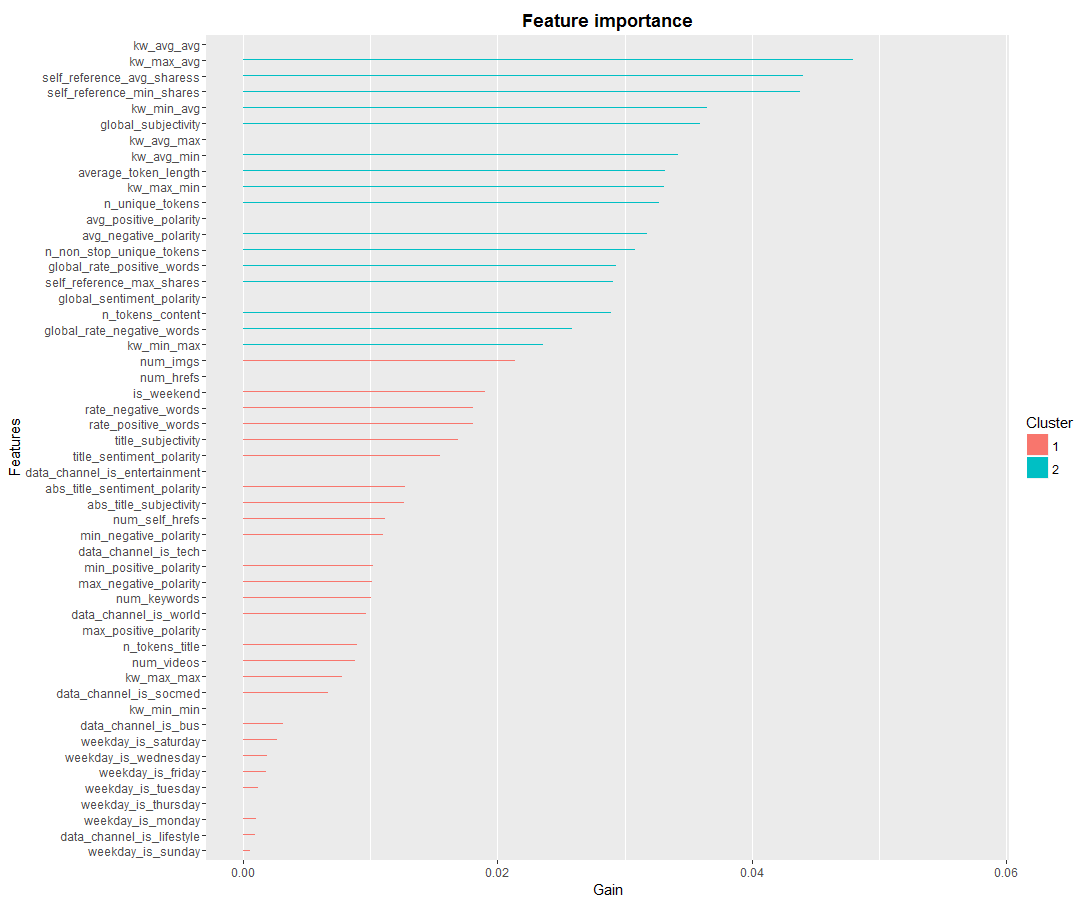
After the preprocessing of the dataset, we tried to solve the problem both in regression and classification methods to predict the popularity.

## CLASSIFICATION METHODS

For starters, our best applied method, in terms of accuracy, is the eXtreme Gradient Boosting algorithm. Boosting classifier belongs to ensemble models; the basic idea is to aggregate hundreds of less accurate tree-based models to form a very accurate model. This model iteratively generates a new tree-based model at each step. People have proposed various ways to get a reasonable base model. In Friedman's Gradient Boosting Machine, it incorporates gradient descent method to build a tree which decreases the objective along the direction of the gradient. In practice we need to generate thousands of trees to get an excellent result on a relatively large data set. However, the current implementation of the algorithm is not fast enough so that we may need to wait for a long time for the result.

In this project, we are using XGBoost. It’s a highly sophisticated algorithm, powerful enough to deal with all sorts of irregularities of data. XGBoost (eXtreme Gradient Boosting) is an advanced implementation of gradient boosted decision trees designed for speed and performance. Gradient boosted trees have to be built in series, so that a step of gradient descent can be taken in order to minimize the loss function. Unlike Random Forests, one can’t simply build the trees in parallel. XGBoost, however, builds the tree itself in a parallel fashion. Also, in return for this speed, we get a much faster grid search for optimizing hyper parameters in model tuning.

In details, after making the transformations needed and getting the best parameters from the tuning process, we built the model and we got an accuracy of 67.23 % with 67.32 % sensitivity. Figure 4 indicates the importance of each feature; in other words, we discovered which features weigh more to predict whether an article will be popular or not. As we can see, they are divided in 2 clusters: the interesting features (cluster 2) and the others (cluster 1).



*Figure 4 – Feature importance plot*

In addition, we experimented with several classifiers, trying a variety of values tuning their parameters and ended up with the best Accuracy and Sensitivity scores respectively, given at the Table 3 that follows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Classifiers | Accuracy | Sensitivity | Parameters | Test options |
| XGBoost | 67.2 % | 0.673 | nrounds = 1000, max\_depth = 50,  eta = 0.1, gamma = 10,  colsample\_bytree = 0.5 and min\_child\_weight = 100 | Percentage split |
| Random Forest | 66.2 % | 0.649 | No.trees=150,  No. features/leaf=7 | Both percentage split and  10-fold CV |
| Logistic Regression | 63.9 % | 0.613 | Stepwise feature selection and default settings by Weka |
| Neural Networks | 62.8 % | 0.615 | default settings by Weka |
| SVM | 61.4 % | 0.577 | default settings by Weka |

*Table 3 – Classification*

At this point, it is necessary to raise the disparity between the 1st classifier applied and the 2nd. Both train several decision trees for one dataset. The main difference is that in Random Forests, trees are independent and in boosting, the tree n+1 focuses its learning on the loss (what has not been well modeled by the tree n). This difference has an impact in a part of the feature importance analysis: the correlated features. Luckily, in our data we cannot talk about correlated features (except a very small percentage, which is mentioned in Part II).

In the rest of this Part, we are explaining a few more things about the classification methods used:

**Random Forests** or random decision forests are an [ensemble learning](https://en.wikipedia.org/wiki/Ensemble_learning) method for [classification](https://en.wikipedia.org/wiki/Statistical_classification), [regression](https://en.wikipedia.org/wiki/Regression_analysis) and other tasks, that operate by constructing a multitude of [decision trees](https://en.wikipedia.org/wiki/Decision_tree_learning) at training time and outputting the class that is the [mode](https://en.wikipedia.org/wiki/Mode_(statistics)) of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees’ habit of [over fitting](https://en.wikipedia.org/wiki/Overfitting) to their training set. This method increases the diversity and hence can reduce the variance.

We trained the Random Forests classifier on the balanced and unbalanced training set using different parameters such as the number of random features to be selected at each node, the number of decision trees the ensemble model would contain in Weka. We used an numFeatures parameter value of 7 based on the guideline of selecting a value that is the square root of the number of attributes, in our case 54. We chose the number of decision trees: numIterations (150) that minimized the out-of-bag error and the number of trees, which helped us to follow the law of parsimony – keeping the model simple.

**Logistic regression** or logit regression is a [regression](https://en.wikipedia.org/wiki/Regression_analysis) model where the dependent variable is [categorical](https://en.wikipedia.org/wiki/Categorical_variable). This model covers the case of [binary dependent variables](https://en.wikipedia.org/wiki/Binary_variable)—that is, where it can take only two values. Cases with more than two categories are referred as [multinomial logistic regression](https://en.wikipedia.org/wiki/Multinomial_logistic_regression), or, if the multiple categories are [ordered](https://en.wikipedia.org/wiki/Level_of_measurement#Ordinal_type), as [ordinal logistic regression](https://en.wikipedia.org/wiki/Ordinal_logistic_regression). Logistic regression measures the relationship between the categorical/dependent variable and one or more independent variables by estimating probabilities using a [logistic function](https://en.wikipedia.org/wiki/Logistic_function), which is the cumulative logistic distribution. Logistic regression can be seen as a special case of the [generalized linear model](https://en.wikipedia.org/wiki/Generalized_linear_model) and thus analogous to [linear regression](https://en.wikipedia.org/wiki/Linear_regression). However, it is based on quite different assumptions (about the relationship between dependent and independent variables) from those of linear regression.

**Neural Networks** are a computational approach which is based on a large collection of neural units loosely modeling the way the brain solves problems with large clusters of biological neurons connected by axons. Each neural unit is connected with many others, and links can be enforcing or inhibitory in their effect on the activation state of connected neural units. Each individual neural unit may have a summation function which combines the values of all its inputs together. There may be a threshold function or limiting function on each connection and on the unit itself such that it must surpass it before it can propagate to other neurons. These systems are self-learning and trained rather than explicitly programmed and excel in areas where the solution or feature detection is difficult to express in a traditional computer program.

Neural networks typically consist of multiple layers or a cube design. Back propagation is where the forward stimulation is used to reset weights on the "front" neural units and this is sometimes done in combination with training where the correct result is known. More modern networks are a bit freer flowing in terms of stimulation and inhibition with connections interacting in a much more chaotic and complex fashion. Dynamic neural networks are the most advanced and can, based on rules, form new connections and even new neural units while disabling others. The goal of the neural network is to solve problems in the same way that the human brain would. Neural networks are based on real numbers, with the value of the core and of the axon typically being a representation between 0 and 1.

**Support Vector Machines** (**SVMs**) analyze data used for classification and regression analysis. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap (maximum margin hyperplane) that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on. In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what a different kernel (Radial Basis Function, Polynomial and Hyperbolic Tangent) implicitly mapping their inputs into high-dimensional feature spaces.

## CLUSTERING

In this part we are using two different methods for clustering, which are the most common forms of unsupervised learning. This means that we hope that the distribution and the makeup of the data will determine the cluster membership. We then, describe the K-means flat clustering algorithm, and the Expectation Maximization (or EM) algorithm, a soft clustering algorithm. K-means is perhaps the most widely used flat clustering algorithm because of its simplicity and efficiency. The EM algorithm is a generalization of K-means and can be applied to a large variety of document representations and distributions.

K-means clustering is a method of [vector quantization](https://en.wikipedia.org/wiki/Vector_quantization), originally from [signal processing](https://en.wikipedia.org/wiki/Signal_processing), that is popular for [cluster analysis](https://en.wikipedia.org/wiki/Cluster_analysis) in [data mining](https://en.wikipedia.org/wiki/Data_mining). K-means clustering aims to [partition](https://en.wikipedia.org/wiki/Partition_of_a_set) n observations into k clusters in which each observation belongs to the [cluster](https://en.wikipedia.org/wiki/Cluster_(statistics)) with the nearest [mean](https://en.wikipedia.org/wiki/Mean), serving as a [prototype](https://en.wikipedia.org/wiki/Prototype) of the cluster. This results in a partitioning of the data space into [Voronoi cells](https://en.wikipedia.org/wiki/Voronoi_cell).

The procedure follows a simple and easy way to classify a given data set through a certain number of clusters (assume k clusters) fixed a priori. The main idea is to define k centroids, one for each cluster. These centroids should be placed in a tricky way because different location causes different result. So, it is for the best, to place them as distant as possible from each other. The next step is to take each point belonging to a given data set and associate it to the nearest centroid. When no point is pending, the first step is completed and an early grouping is done. At this point, we need to re-calculate k new centroids as barycenters of the clusters, resulting from the previous step. After we have these k new centroids, a new binding has to be done between the same data set points and the nearest new centroid. A loop has been generated. As a result of this loop we may notice that the k centroids change their location step by step until no more changes are done. In other words centroids do not move anymore. Finally, this algorithm aims at minimizing an objective function, here a squared error function.

In this case, the minimum number of incorrectly applied instances was given by applying K-means algorithm with 2 clusters (popular, non-popular) at 500 iterations. Also, the most effective distance function used for instances comparison is Euclidean.

The next clustering method used is Expectation–Maximization (EM) algorithm. EM is an [iterative method](https://en.wikipedia.org/wiki/Iterative_method) for finding [maximum likelihood](https://en.wikipedia.org/wiki/Maximum_likelihood) or [maximum a posteriori](https://en.wikipedia.org/wiki/Maximum_a_posteriori) (MAP) estimates of [parameters](https://en.wikipedia.org/wiki/Parameter) in [statistical models](https://en.wikipedia.org/wiki/Statistical_model), where the model depends on unobserved [latent variables](https://en.wikipedia.org/wiki/Latent_variable). The EM iteration alternates between performing an expectation (E) step, which creates a function for the expectation of the [log-likelihood](https://en.wikipedia.org/wiki/Likelihood_function#Log-likelihood) evaluated using the current estimate for the parameters, and maximization (M) step, which computes parameters maximizing the expected log-likelihood found on the E step. These parameter-estimates are then used to determine the distribution of the latent variables in the next E step. EM assigns a probability distribution to each instance which indicates its probability of belonging to each of the clusters. EM can decide how many clusters to create by cross validation, but again we specified apriori to generate 2 clusters.

|  |  |  |
| --- | --- | --- |
| Cluter | Incorrectly clustered instances | Parameters |
| EM | 48.5294% | No. of clusters=2,  100 iterations |
| K-means | 48.9204% | K=2,  Euclidean Distance,  500 iterations |

*Table 4 – Clustering*

From Table 4 above, we concluded that neither K-means, nor EM algorithm were effective in assigning the data points to right clusters. This fact brings up again the issue with the initial clusters and features.

# PART IV – FEATURE SELECTION

As there are a lot of independent variables in our dataset, we took a further step to re-examine our methods after performing Principal Component Analysis (PCA). Principal Component Analysis is a dimension reduction technique that uses an [orthogonal transformation](https://en.wikipedia.org/wiki/Orthogonal_transformation) to convert a set of observations of possibly correlated variables into a set of values of [linearly uncorrelated](https://en.wikipedia.org/wiki/Correlation_and_dependence) variables called principal components. The number of principal components is less than or equal to the number of original variables. This transformation is defined in such a way that the first principal component has the largest possible [variance](https://en.wikipedia.org/wiki/Variance) (that is, accounts for as much of the variability in the data as possible), and each succeeding component in turn has the highest variance possible under the constraint that it is [orthogonal](https://en.wikipedia.org/wiki/Orthogonal) to the preceding components. The resulting vectors are an uncorrelated [orthogonal basis set](https://en.wikipedia.org/wiki/Orthogonal_basis_set). PCA is sensitive to the relative scaling of the original variables.

However, since our dataset was well structured and there weren’t many features correlated, PCA actually worsened the performance of our classifiers used on our initial dataset. PCA provided 34 principal components which contain 88 % statistical info of the initial dataset.

An alternative evaluator used for dimensionality reduction, CfsSubsetEval, evaluates the worth of a subset of attributes by considering the individual predictive ability of each feature along with the degree of redundancy between them. CfsSubsetEval in combination with BestFirst method, which searches the space of attribute subsets by greedy hillclimbing augmented with a backtracking facility provided a subset of 10 attributes as follows:

* n\_non\_stop\_unique\_tokens
* data\_channel\_is\_entertainment
* data\_channel\_is\_socmed
* data\_channel\_is\_world
* kw\_min\_avg
* kw\_max\_avg
* kw\_avg\_avg
* self\_reference\_min\_shares
* self\_reference\_avg\_shares
* is\_weekend

After trying the Data Mining Techniques mentioned above, not even this subset of features performed better.

Last but not least, the evaluator OneRAttributeEval which evaluates the worth of an attribute by using the OneR classifier, combined with the Ranker method, offered us a list of ranked attributes. Even this method did not achieve any improvement.

# PART V - CONCLUSION

In this project, various regression and classification models, as well as clustering, were trained for predicting the popularity of an online news article before its publication, using a large and recent dataset, with 39.644 articles collected during a 2 year period from the popular Mashable news service. Overall, the work done in this project was important in understanding the main reasons that effects popularity of online news from a statistical point, not from writing style and content.

All in all, the best result was achieved by XGBoost algorithm. XGBoost achieves high speed and better accuracy from any other Classification model. But as it seems, no algorithm can reach 70 % accuracy given the data set we have, even though they are state-of-the art.

We also demonstrated the most significant features influencing the popularity of an online article. However, no subset of them could improve the prediction of any model.

In future work, we could suggest an attempt of constructing new features, derived from the initial ones or maybe an outlier removal so as to balance our dataset even better and get articles that went viral or not at all shared, out.

# REFERENCES

1. Predicting and Evaluating the popularity of online news (He Ren and Quan Yang, Stanford University)

<http://cs229.stanford.edu/proj2015/328_report.pdf>

1. Generalizing Random Forest Model for Online News Popularity Classification( Mak Hozo, Henna Shah and Julio Sotelo)

<http://blog.expertis.co/generalizing-random-forest-model-for-online-news-popularity-classification/>

1. Complete Guide to parameter tuning in XGBoost

https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/

1. Mastering Predictive Analytics with R (Rui Miguel Forte)

https://www.packtpub.com/application-development/mastering-predictive-analytics-r

1. YouTube video courses for Data Mining with Weka (Ian H. Hitten, University of Waikato)

<https://www.youtube.com/watch?v=L4Nlv-apkkc&index=3&list=LLOSIxMWS7gtzx6IlQbN3Yhg>

1. An Introduction to Statistical Learning (James, Gareth)
2. Advanced Data Mining Techniques (DL Olson)
3. Mining of Massive Datasets (Jure Leskovec, Anand Rajaraman, Jeffrey D. Ullman)