**OPIM 5604 Predictive Modeling SEC B14-1158**

**Group Project - Content Popularity Prediction**

**Team Members for Team 2:**

**Chunxue Cao**

**Monique Brown**

**Kanchi Gaikwad**

**Nirmal Chandrasingh**

**Leela Kishore Donthireddy**

**Venkata Yogesh Kumar Reddy, Gongati**

**Business Problem:**

This project is about predicting the content popularity of the articles that are published online. Recent research trends showed that predicting such popularity has direct implications to ad revenue models and content distribution markets.

We have used a large set of inputs including the information about the article’s digital content, general popularity of the news. As predicting an article’s popularity is based on the regularity with which a user focuses on content, few natural language features are also considered. These features were considered possibly relevant to influence the number of shares.

Heterogeneous properties about the articles are summarized in a dataset and we compared them with other characteristics for a better prediction accuracy. Using the popular SEMMA approach, we have explored, cleansed the data and prepared a ready-to-model dataset. Five classification methods were used to predict two popularity classes (“POPULAR”, “UNPOPULAR”) and ensemble models are tried to improve the accuracy of the predictive classifications.

**Results:**

We have a baseline accuracy of 53.1%; and from our analysis the results are:

1. Bootstrap Forest model classified POPULAR (True Positive) with an accuracy of 75.5%.
2. Nominal Logistic Regression was able to classify 70.11% of the popular urls correctly.

**Recommendations:**

1. Conventional data models like Latent Dirichlet Allocation (LDA) implicitly capture the document-level word co-occurrence patterns to reveal topics. Through this project, we observed that LDA had great influence in the learning process of model parameters and deviated the purpose of prediction towards a low classification accuracy. This shortcomings suggests that a non-supervised learning model like LDA cannot be directly used in classification. Data distribution can be considered a remedy for reliable source of information.
2. Metadata columns were important to model large volumes of information; but we had to drop 9 variables as they couldn’t clearly communicate their relation with other features. We feel that a robust data dictionary is necessary to understand the variables of a dataset.
3. Another dimension of predicting the popularity could be analyzing the social features of an author. Additional readings suggested that there is a relationship between author’s social quant and their article’s popularity.

**Brief Introduction to Business Problem:**

The articles published from Mashable are used for the analysis. Objective of predicting article popularity is not only to help authors and content providers to improve the quality of the publication; but also helps advertisers to understand or influence public opinion.

It was analyzed that lifetime is an important property about ranking an article; the data was collected during a two year period, from January 7 2013 to January 7 2015. Dataset had 39,644 records with 58 independent variables and 1 dependent variable categorized into two classes.

The variables include the information about an article’s digital media content, earlier popularity of the news referred in the article, logarithmic transformed numerical values, and particularities of other media streams.

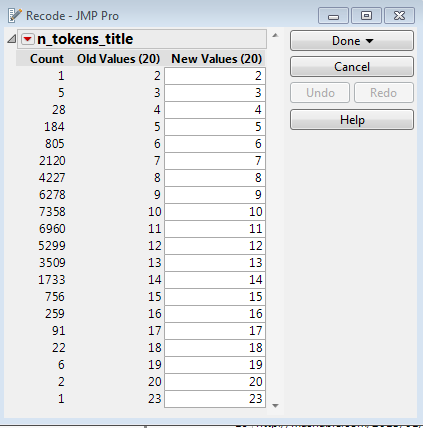
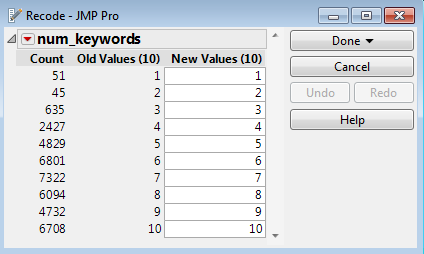
**Project Approach:**

1. ***Data Sampling*:** The data had 39,644 rows along with 61 columns. As it was well under the computational capabilities of JMP, no further sampling was required.
2. ***Data Exploration and Modification*:**Below is the detailed explanation for the 8 steps:
3. **Variable Classification**- Checked to see if variables were correctly classified as Continuous, Nominal, and Ordinal.

* **Approach -** Select Variable -> Columns -> Utilities -> Recode
* **Result -** All the variables were originally classified as Continuous. However, ***n\_token\_title*** *and* ***num\_keywords*** were ***changed to ordinal***.
* **Reasons -**

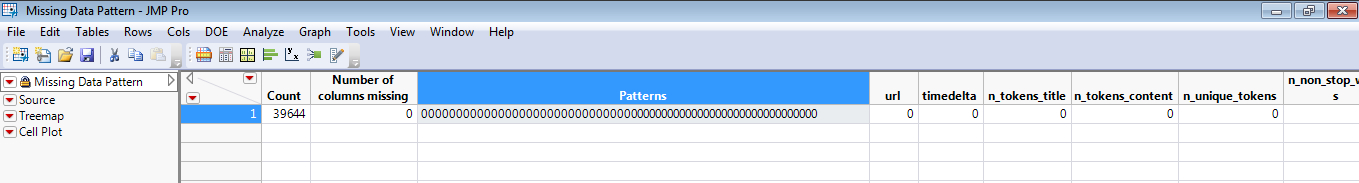
*N\_token\_title:* It gives the number of words in the title of an article. Since it had values only from 2 to 23 and has an order to it, it was made ordinal.

*Num\_keywords:* It represents the number of keywords associated with an article. A maximum of 10 keywords can be associated with an article. Since there is an upper limit and has an order to it, it was made ordinal.

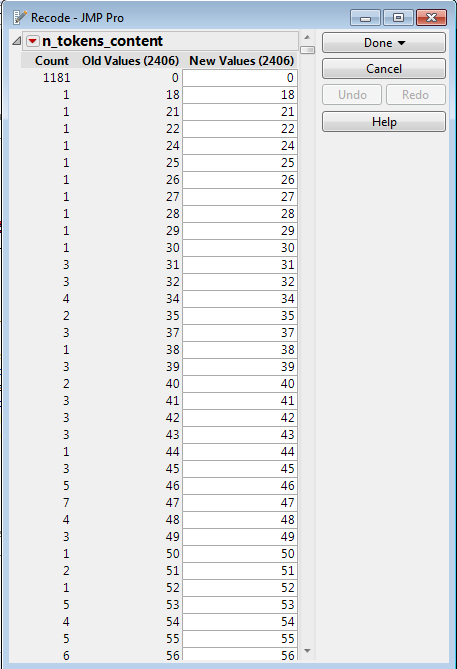
1. **Missing Value Pattern**- Checked to see if any variables have missing data

* **Approach -** Go to Tables tab- > Missing Data pattern
* **Result -** No missing data was found.

****

* **Exception -** While the missing data pattern gave no missing data, we found an exception to this case. While applying recode to the following columns we saw that there were 1181 rows which had a value of 0 for each of the columns
* n\_tokens\_content
* num\_hrefs
* num\_imgs
* num\_videos

Assuming that an article has at least 1 word in the content or at least 1 image/video/reference link; we manually checked few URLs and found that they did have words in them. Since this had discrepancy with the captured data, we flagged the 1181 rows as missing data. As there is no business logic to impute these rows, we deleted these rows.



1. **Reduce Dimensionality**- Deleted variables if
2. **The variable is a primary key**

* URL variable was removed as it is the primary key. Each row had a unique value and hence it is not a good explanatory variable.

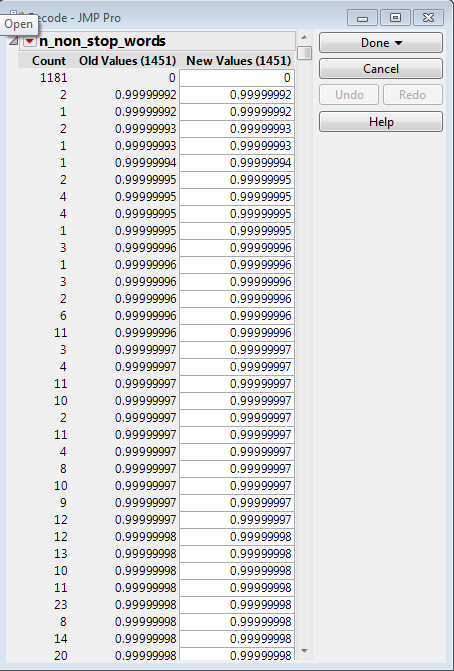
1. **The variable has values which cannot be interpreted**

* kw\_min\_min - Minimum shares for worst keyword in an article
* kw\_max\_min - Maximum shares for worst keyword in an article
* kw\_avg\_min - Average shares for worst keyword in an article
* kw\_min\_max - Minimum shares for best keyword in an article
* kw\_max\_max - Maximum shares for best keyword in an article
* kw\_avg\_max - Average shares for best keyword in an article
* kw\_min\_avg - Minimum shares for average keyword in an article
* kw\_max\_avg - Maximum shares for average keyword in an article
* kw\_avg\_avg - Average shares for average keyword in an article

Variables involving min and average shares had values like ‘-1’ which could not be explained well. Due to the unreliable nature of the column, it was removed as predicting the dependent variable based on a column that has inconsistent values would be wrong.

1. **The variable doesn't show much variation**

* Deleted **n\_non\_stop\_words** variableas all the values are nearly 1 and didn't show much variation.



1. **Non-predictive variable**

* time\_delta : It gives the time difference between the day the article was published and 27th December, 2014

Since this variable will be 0 for an article before it gets published it cannot be used as an input to a model that predicts the shares an article is likely to get. Hence, it was removed on the grounds of not being available at the time of running a model.

1. **The variable is highly correlated to other variables (Correlation Analysis)**

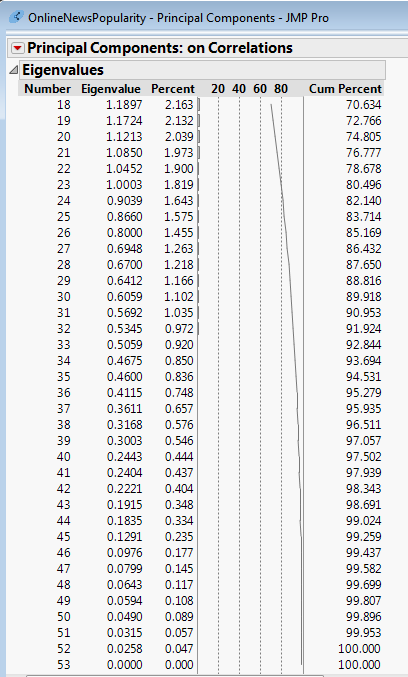
* **Approach** ­- Analyze -> Multivariate Methods -> Multivariate
* **Assumption** - 2 variables are highly correlated if the correlation coefficient is >0.9
* **Result -** Of all variables only 2 variables exhibited high correlation

Deleted n\_unique\_tokens amongst n\_non\_stop\_unique\_tokens and n\_unique\_tokens which showed a high correlation of 0.9999 as one variable can be explained by another variable.



1. **A few variables can explain all the variables by carrying out Principal Component Analysis**

* **Approach** ­- Analyze -> Multivariate Methods -> Principal Components
* **Result -** 31 variables captures 90% of the total explanatory power. The variables become tough to explain when it is replaced by principal components. As a trade-off between reducing dimensionality and not making the model tough to explain we chose to go ahead with the original list of variables.

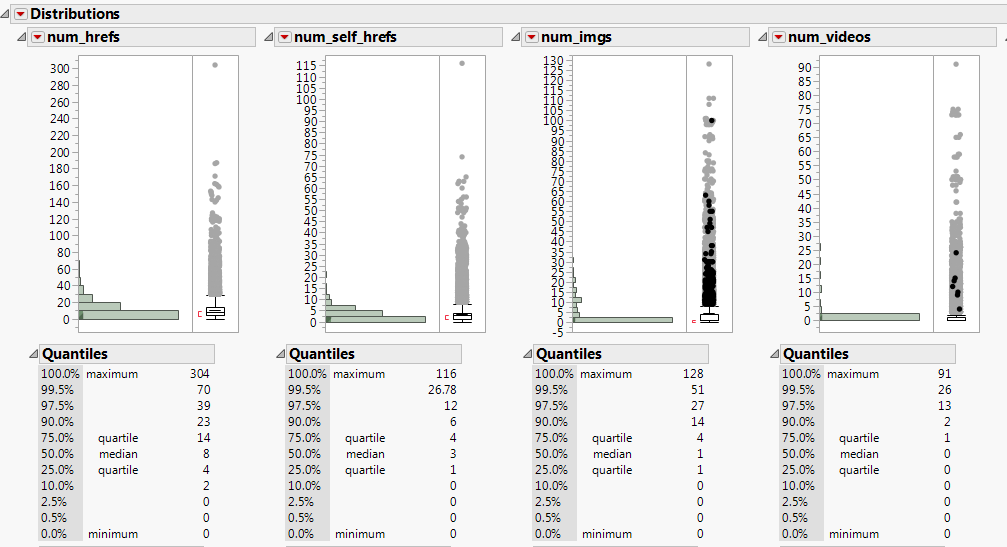


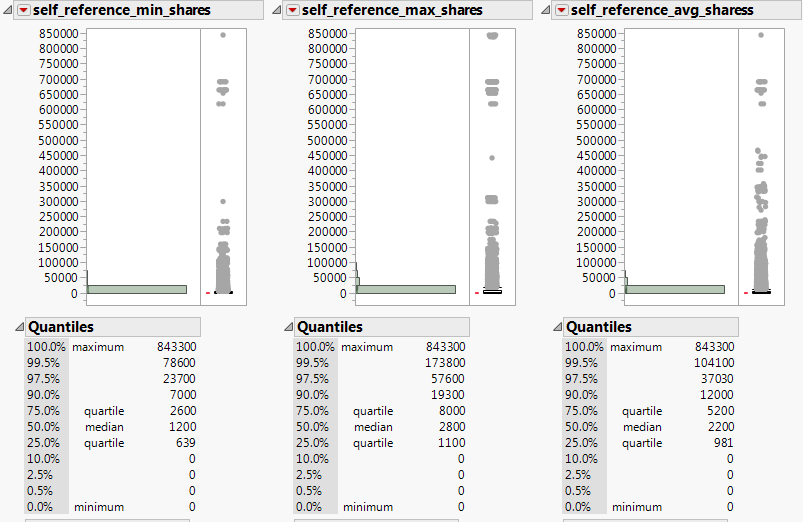
1. **Outlier Treatment**- Deleted rows which are:
2. **Located extremely far from the mean.**

* **Approach -** Analyze -> Distribution
* **Result -** All variables were checked to see if there were any extreme outliers. We removed only extreme outliers (which were sparsely populated) because we didn't want to remove many of them as it would lead to a significant reduction in the overall dataset. Amongst all variables, the following ones showed extreme outliers
* The table below shows the variables which had extreme outliers along with the number of rows affected

|  |  |  |
| --- | --- | --- |
| **Variable** | **Outliers** | **#Rows affected** |
| num\_hrefs | 304 | 1 |
| num\_self\_hrefs | 116 | 1 |
| num\_imgs | 128 | 1 |
| num\_videos | 91 | 1 |
| self\_reference\_min\_shares | >600,000 | 25 |
| self\_reference\_max\_shares | >400,000 | 89 |
| self\_reference\_avg\_shares | >200,000 | 11 |

* Snapshots showing extreme outliers amongst shortlisted variables

****

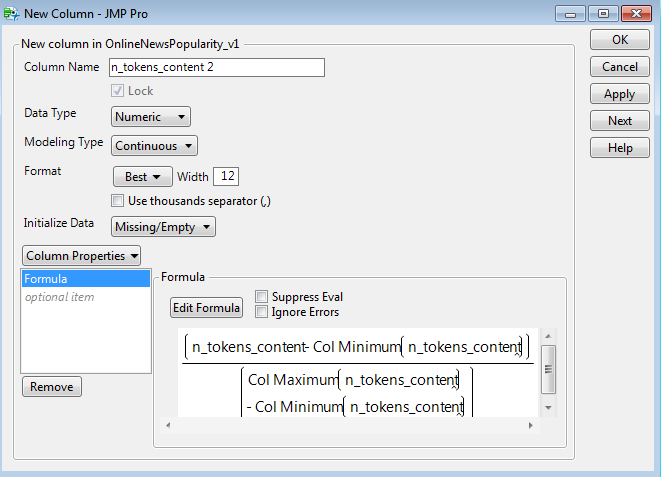
****

1. **If the data is skewed**

* **Approach -** Analyze -> Distribution -> Enter the continuous variables which are skewed (not normally distributed) -> Click on red arrow -> Continuous Fit -> All -> Select the best fit
* **Result -** This not only gave a complex transformation which was difficult to interpret but also did not improve the accuracy while modeling. So we decided to keep the original values for these variables as we wanted to interpret the values after the modeling

1. **Standardization**- Different continuous variables have different scales due to different unit of measurement and need to be brought down to a single scale so that a variable doesn't dominate other variables while building the model and are comparable. All continuous variables are transformed on to a scale of 0 to 1. A new column is created for each variable with a suffix **\_std**. The original columns were excluded. An example is shown below

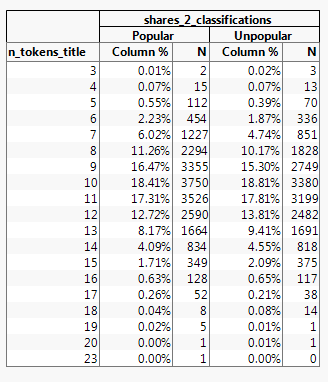
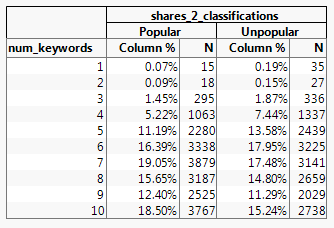
* **Approach -** Cols -> New Column -> Enter the column name & add standardization formula



1. **Variable Transformation**-Variable transformation was carried out for ordinal columns and dependent variable
2. **Ordinal Variables -** For the ordinal columns (i.e. num\_keywords and n\_tokens\_title), each category/value has to have enough number of rows to explain the dependent variable. This way the dependent variable looks more balanced and can be sampled appropriately for training and validation purposes. Else a judgment based on extremely few number of rows will prove to be false/biased.
   * **Approach -** Analyze -> Tabulate
   * **Result -** It shows a table with %distribution and number of rows of each value for popular and unpopular

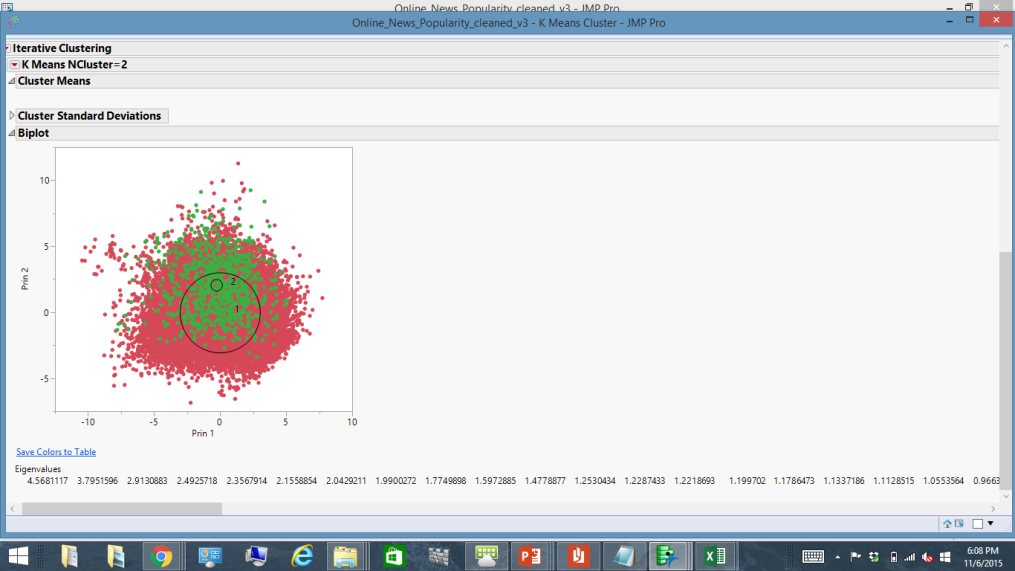
*n\_token\_title* - As values 3, 4 and 5 had similar %distribution across popular and unpopular and each of them had a very low row count, they were imputed to value 5. Similarly, as values above 15 had similar %distribution across popular and unpopular and each of them had a very low row count, they were imputed to value 15.

*num\_keywords* - As values 1, 2 and 3 had similar %distribution across popular and unpopular and each of them had a very low row count, they were imputed to value 3.

*** ***

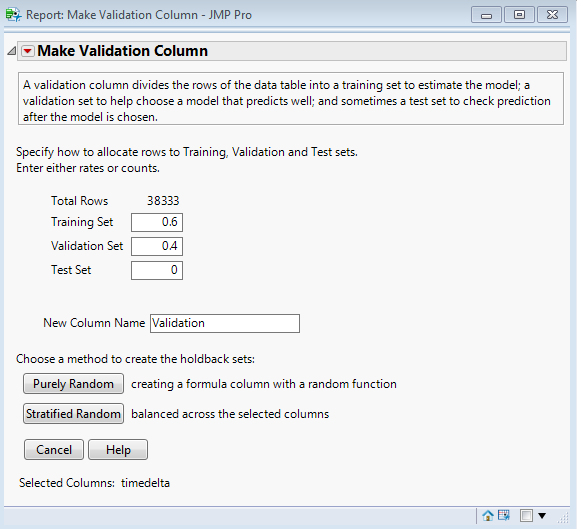
1. **Dependent Variable *-***We decided to predict our model for dependent variable as both continuous and categorical. The dependent variable (shares which is a continuous variable) was transformed into a categorical variable by forming 2 categories (**Popular and Unpopular).** We carried out a research online to find out if there were any defined industrial standard cut off for defining popularity. However, since there wasn't any, we decided on a cut off of 1400 (which is at 50th percentile). We called this new column as shares\_classification
2. **Clustering *-*** Clustering was done to check whether there were any significant groups/patterns in the data that could be modeled individually.

* **Approach *-***Analyze -> Multivariate Methods-> Select all continuous variable (No dependent variable is required to be fed in here) -> Select K-means -> Set Initial Clusters =2
* **Result -** As evident from the biplot, the clusters are not separate. Hence, the dataset needs to be modeled as a whole

******

1. **Split the data into training and validation *-*** The dataset is split into 60% training and 40% validation by creating a validation column

* **Approach -**
* Select the shares\_classification column.
* *C*ols -> Modeling Utilities -> Make Validation column
* Enter 0.6 for training and 0.4 for validation and select purely random as the dependent variable is balanced. for the training and validation data

******

*After the exploration and modeling, the dataset now has 38,333 rows which can be used for modeling. shares\_classification is the dependent variable and the rest of them are the explanatory independent variables. We need to find the best model which will predict whether a url will be popular or unpopular*

***3. Modeling***

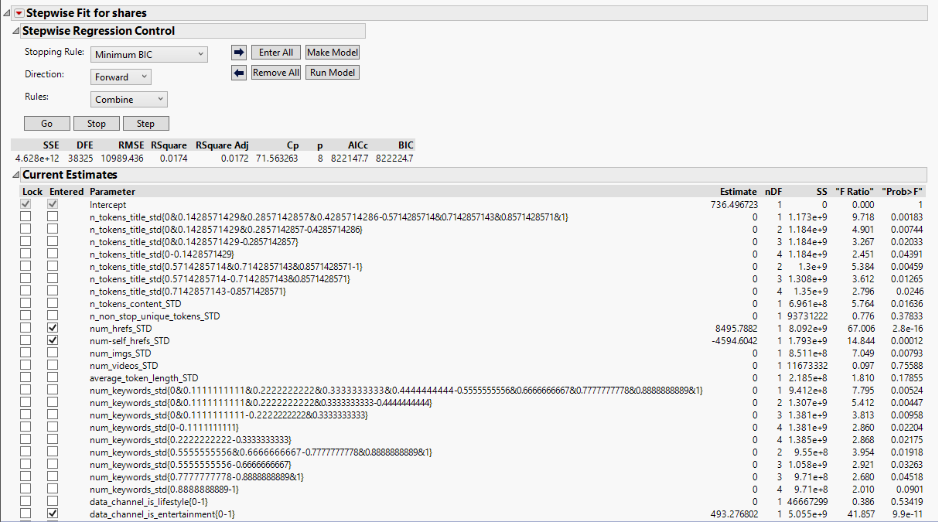
1. ***Continuous Dependent Variable (shares)***

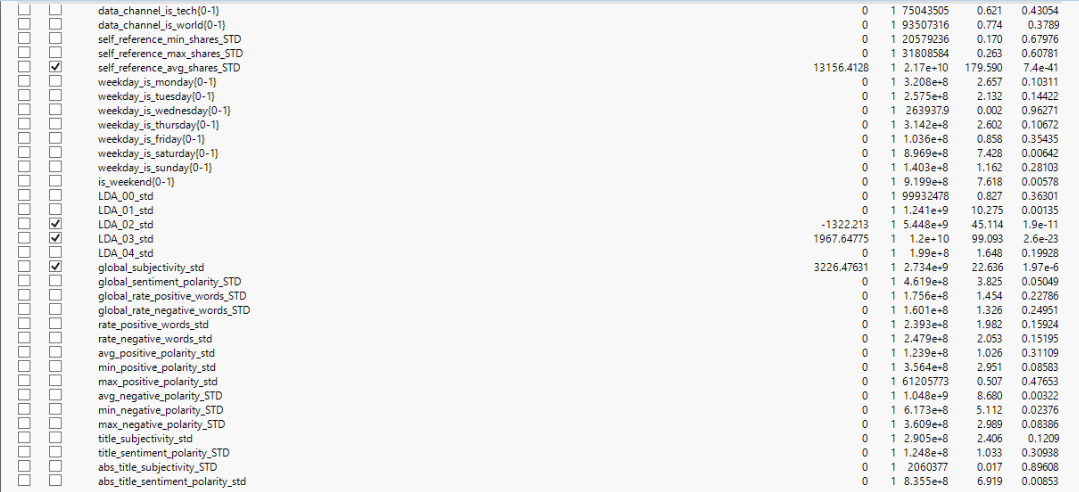
***Linear regression -*** We started with linear regression as the original shares column is continuous variable.

* **Approach -** Analyze -> Fit Model -> Continuous independent variables in X
* **Result -** Linear regression gave the following equation

Shares = 736.497 + 8495.78\*num\_hrefs\_STD + 1967.647\*LDA\_03\_STD - 4594.6042\*num\_self\_hrefs\_STD - 1322.213\*LDA\_02\_STD + 493.277\*data\_Channel\_is\_entertainment + 13156.4128\*self\_reference\_Avg\_shares\_STD + 3226.47\*global\_subjectivity\_STD

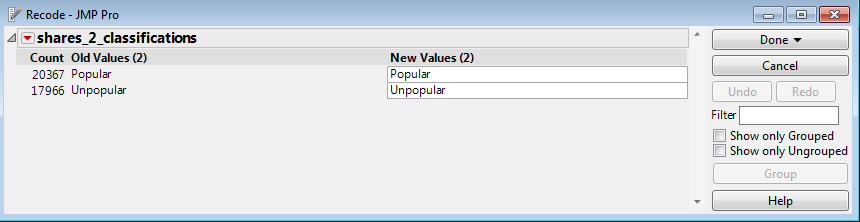
* Based on this equation, the predicted shares were calculated and compared with the actual shares. There was an overall mean difference of 250.
* We added a buffer of 250 and checked how many URLs had their predicted shares within this buffer (actual shares - 250 and actual shares + 250 as lower and upper limit).
* Only 10% had them in this range. Hence linear regression did not turn out to be a good model. This may be because most of the variables in the equation are heavily skewed. Normalizing them would yield better results

******

******

1. **Categorical Dependent Variable**

53.1% of the total urls are popular while 46.9% are unpopular. Hence, for the baseline model where a ‘Every url is expected to be popular’ the prediction% being a false positive is 46.9% (17966/38333) and percentage of the prediction being a true positive is 53.1% (20367/38333). Following models need to be compared with this model.



**Formulas to calculate false predictions:**

* Total number of false predictions = False positives + False negatives
* False positive percentage = False positives/(True positives + False positives)
* False negative percentage= False negative/(False negatives + True negatives)

**'**

1. **Decision Tree (Divide and Conquer Algorithm)**

* **Approach**
* Analyze -> Modeling -> Partition
* Keep splitting till you gain a better Rsquare and classification accuracy without overfiting the model (Rsquare of Validation shouldn’t drop)
* **Results**
* After 20 splits, the decision tree was still trying to improve the accuracy of training data but for validation data there wasn’t any significant improvement. Hence, we stopped at 20 splits as the decision tree was trying to over fit the model and R square stopped increasing post that.
* This model classified the training data and validation data with an accuracy rate of 64.13% and 63.46% respectively for true positives and true negatives calculated from misclassification rate
* Training

Total False predictions - 8249

False Positive % = 34.7%

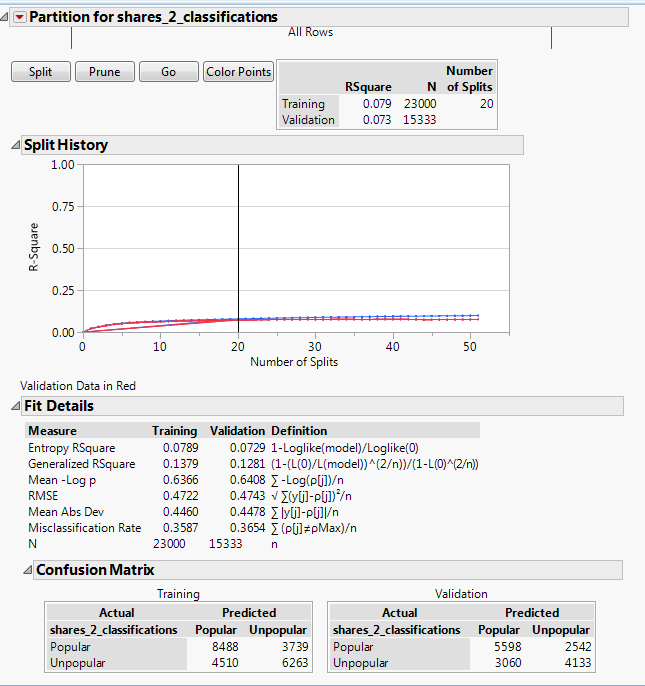
False Negative% = 37.4%

* Validation

Total False predictions - 5602

False Positive % = 35.3%

False Negative% = 38.1%

**

1. **Neural Networks Algorithm**

* **Approach**
* Analyze -> Modeling -> Neural
* To find out the best fit algorithm which gives better accuracy, we played with the number of functions at the two layers (Learning Layer and Classification Layer)

**Results**

* After trying different combinations, we found that NGaussian function wasn't giving any significant improvement in accuracy rate for validation data. Hence, we considered only TanH and Linear function ( NTanH(3) NLinear(2) NTanH2(2) NLinear2(1) ) as weights which gave significantly best possible accuracy with least complexity
* This model classified the training data and validation data with an accuracy rate of 65.52% and 64.64% respectively for true positives and true negatives calculated from misclassification rate
* Training

Total False predictions - 7932

False Positive % = 33.2%

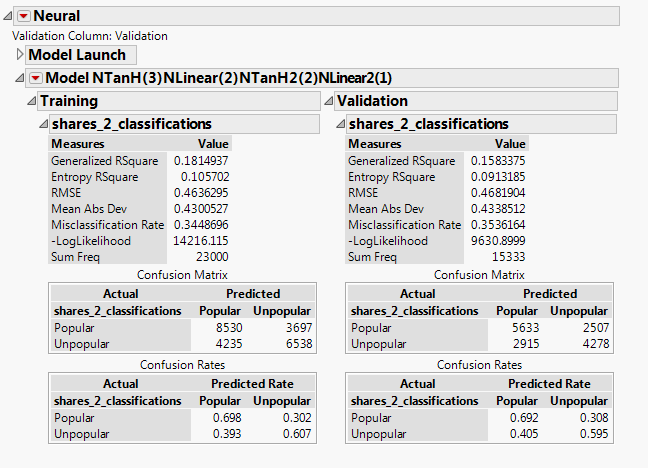
False Negative% = 36.1%

* Validation

Total False predictions - 5422

False Positive % = 34.1%

False Negative% = 36.9%

******

1. **Discriminant Analysis**

Linear Discriminant Analysis (LDA) extracts the discriminant function in which the coefficients of linear combination tells us about the discriminative ability of a variable. So to understand which variables are contributing to the classification group we go ahead with this analysis

* **Approach** 
  + Analyze -> Multivariate Methods -> Discriminant Analysis
  + Place continuous independent variables in Y,Covariates, and the categorical dependent variable in X,Categories.
  + **Results**
* This model classified the training data and validation data with an accuracy of 64.1% and 64.08% respectively for true positives and true negatives calculated from misclassification rate
* Training

Total False predictions - 8259

False Positive % = 32.9%

False Negative% = 39.0%

* Validation

Total False predictions - 5508

False Positive % = 33.33%

False Negative% = 38.7%

****

1. **Logistic Regression**

Unlike linear regression, it fits probabilities for the response levels using a logistic function and it is considered to better predict the groups when it is not reasonable to assume that independent variables are normally distributed

**Approach**

* + Analyze -> Fit Model

**Results**

* This model classified the training data and validation data with an accuracy of 65.65% and 65.65% respectively for true positives and true negatives calculated from misclassification rate
* Training

Total False predictions - 8209

False Positive % = 34.6%

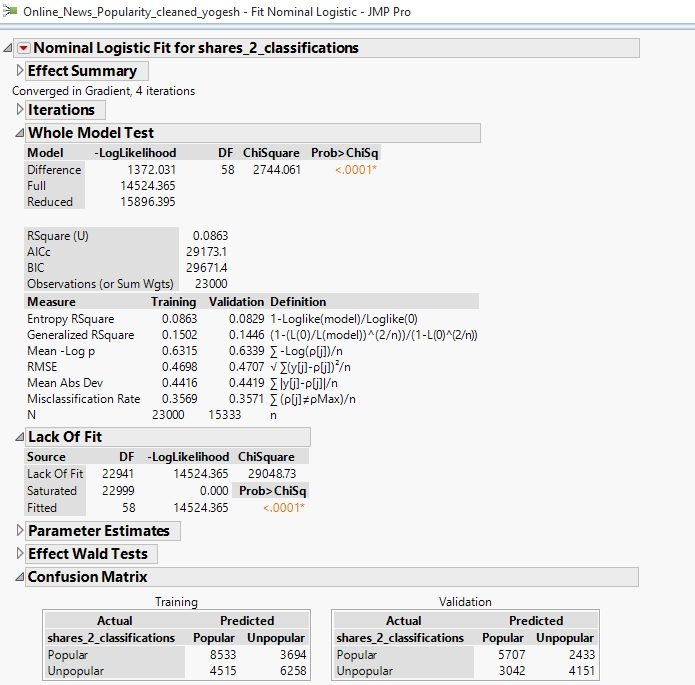
False Negative% = 37.1%

* Validation

Total False predictions - 5475

False Positive % = 34.8%

False Negative% = 37.0%



1. **Bootstrap Forest**

It makes multiple trees and averages the predicted values to get the final value

* + **Approach**
  + Analyze -> Modeling -> Partition -> Select Bootstrap instead of Decision Tree under Method
  + **Result**
  + This model classified the training data and validation data with an accuracy of 71.42% and 66.2% respectively for true positives and true negatives calculated from misclassification rate
  + Training

Total False predictions - 6573

False Positive % = 28.7%

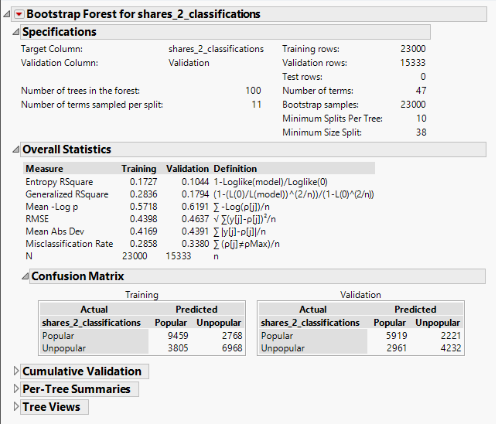
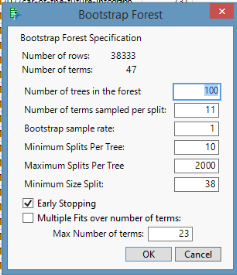
False Negative% = 28.4%

* Validation

Total False predictions - 5182

False Positive % = 33.3%

False Negative% = 34.4%



**4. Model Comparison**

All the models are better than the baseline model in terms of accuracy

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Validation | Decision Tree | Neural | Discriminant | Logistic | Bootstrap |
| False Positives% | *35.3%* | *34.1%* | *33.3%* | *34.8%* | *33.3%* |
| False Negative% | *38.1%* | *36.9%* | *38.7%* | *37.0%* | *34.4%* |
| Total wrong predictions | *5602* | *5422* | *5508* | *5475* | *5182* |

Overall, bootstrap forest has the least number of wrong predictions. But to comment on the best model we should know the cost of false positive and cost of false negative. Since, the bootstrap forest has the least number of false positives and negatives it is the best model.

**Our assumption (Assumption had to be made as there was no business standard):**

Cost of false positive (Predicting a popular for an actual unpopular article) is higher than cost of false negative (Predicting a ‘unpopular’ for an actual popular article) as the former doesn’t get back the returns that were invested in the ads along with the articles and also cannibalizes the chances of otherwise popular articles

**Ensemble model of Decision Tree, Neural Network, Logistic Regression, Discriminant and Bootstrap Forest using Bootstrap Forest**

All the 5 models give similar results and an ensemble model was built to improve the accuracy of the predictive classification. Since, Bootstrap forest gave better accuracy, the ensemble model was built using Bootstrap Forest

**Approach**

* + Analyze -> Modeling -> Partition -> Select Bootstrap under Method Option

**Result**

* + This model classified the training data and validation data with an accuracy of 72% and 66.2% respectively for true positives and true negatives calculated from misclassification rate
  + Training

Total False predictions - 6620

False Positive % = 26.9%

False Negative% = 30.9%

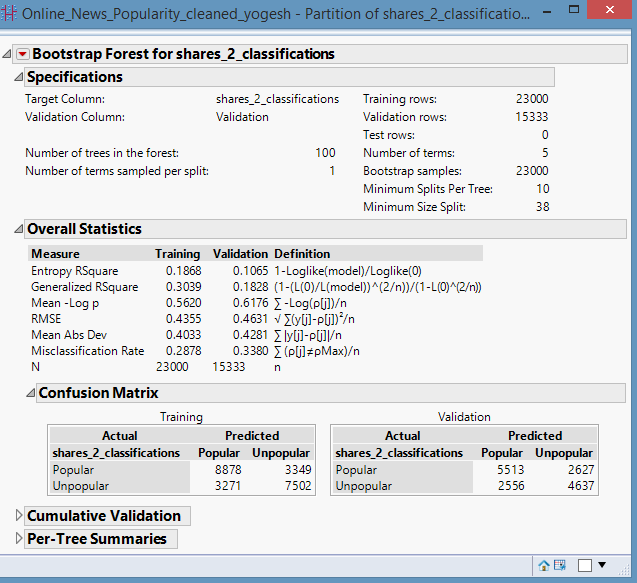
* Validation

Total False predictions - 5183

False Positive % = 31.7%

False Negative% = 36.2%

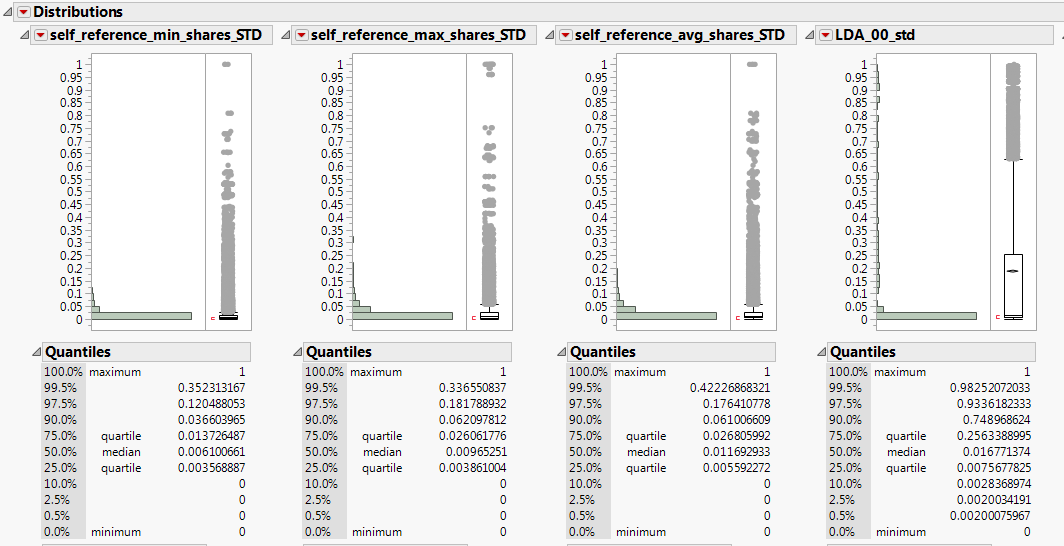
* The ensemble model improved the false positives by 1.6%. However, it performed poorer for false negatives by 1.8%. Hence, the best model can be selected based on the business cost of the misclassifications

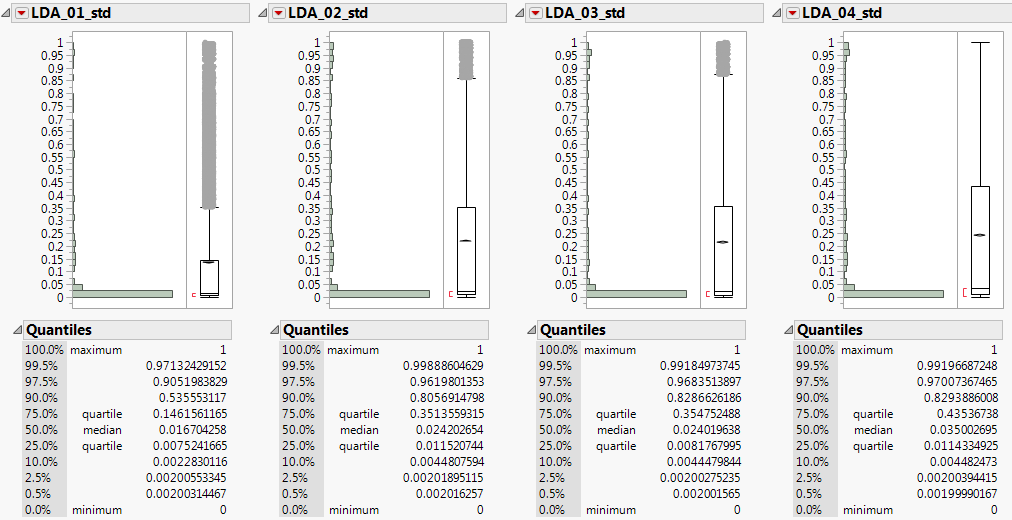


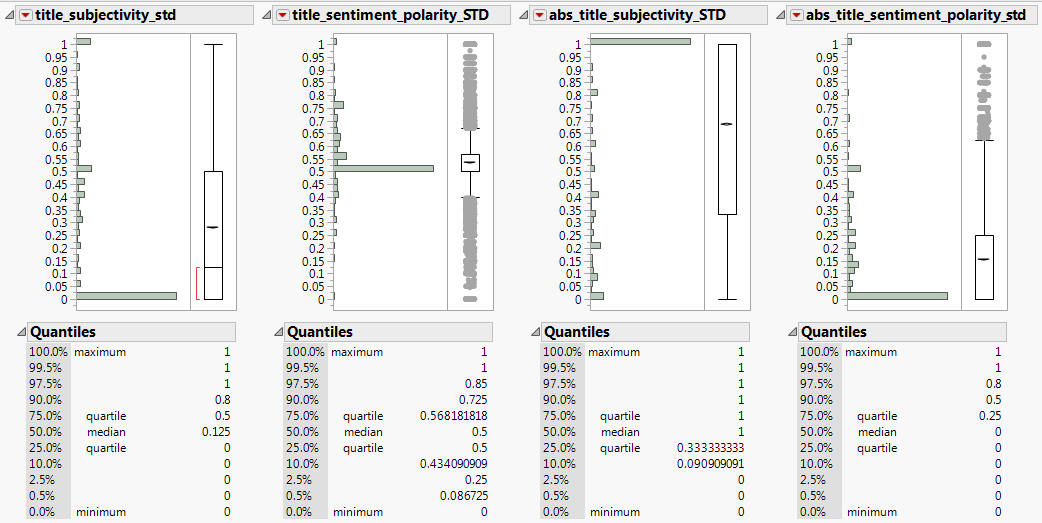
**Appendix**

**Failed Attempts to Improve Accuracy of Model**

Variable Transformation - For the continuous variables which were skewed, we created 4 buckets for each of them at 25th percentile, 50th percentile and 75th percentile for the spread to be uniform across the groups.







With used these transformed columns instead of the respective original columns in one of the best models i.e. the bootstrap forest model. However, the results

* *Training*

*Total False predictions - 6774*

*False Positive % = 29.8%*

*False Negative% = 29.0%*

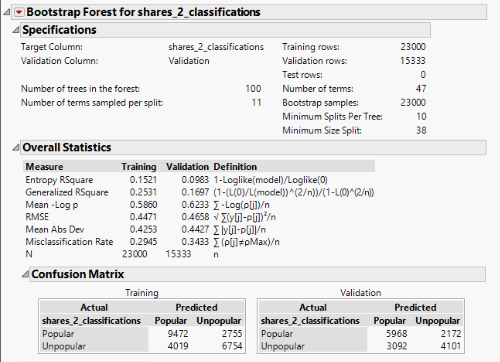
* *Validation*

*Total False predictions - 5264*

*False Positive % = 34.1%*

*False Negative% = 34.6%*

While there is an improvement in the false negative % by 1.6%, it performs poor on false positive % by 2.4%



Bivariate Analysis

We also carried out a bivariate analysis to see how the shares vary with each of the independent variables to understand the empirical relationship between them. However, not much significant results were seen.

