Predicting Online News Popularity Competition Report

Anneke Speijers, Harihara Subramanyam Sreenivasan, Max Elias van Esso

Objective

The objective of this competition was to build a model that could predict the popularity class of an article based on a set of features. Articles were published on the website mashable.com.

Analysis of the data set

One of the most notable features of the dataset was the substantial class imbalance. The percentage of each class in the training set was as follows:

```
Popularity level 1 -  31.59%
Popularity level 2 -  45.88%
Popularity level 3 -  19.04%
Popularity level 4 -  3.33%
Popularity level 1 -  0.157%
```

This posed a significant challenge, since all classification models built on this dataset were biased towards predicting classes 1 and 2 (and to some extent 3), whilst being unable to accurately predict classes 4 and 5.

There were 59 features in total, out of which 34 were continuous. There were two groups of categorical variables:

1) a binary set used to denote which data channel the article was publisehd on; Lifestyle, Entertainment, Business, Social Media, Technology or World. In the training set there were 4,658 articles not designated to any data channel and in the test set there are 1476.
2) another binary set used to denote the day of the week on which the article was released, plus an extra variable for whether or not it was released on a weekend.

The continous variables can be grouped the following way:

1) Number of images and videos in the articles
2) Number of references to the article
3) Number of shares of the articles referenced by the article being studied
4) Analyses of the number of words in the article and title of article
5) Analyses of the keywords present in the article
6) LDA00 to LDA04 : Closeness to the most relevant topics at the time of release
7) Sentiment analyses of the title and the articles

In this report, when the term “Original Features” is used, it refers to the above mentioned features which were obtained from the datasets.

A baseline model - multinomial logistic regression

One of the simplist ways to approach a multiclass classification problem is to apply multinomial logistic rgression. We did this by implementing the “multinom” function from the “nnet” package. Doing so, on the original features matrix, on a training sample of 70% of the entire training sample, gave us a baseline accuracy of 0.4823, on the remaining 30% validation set. Our goal from this point onwards was to develop a model that could outperform this.

Approach to Model Building

The models we considered throughout this competition included multinomial logistic regression, random forest, gradient boosting and (briefly) deep learning. However, our approach was to construct and transform variables with the goal of discovering new informative features with high predictive power to include in our models, and then choose between models.

Feature Transformations

We carried out the following feature transformations (continuous variables only) to evaluate their predictive capabilities.

1) x^2
2) x^3
3) $x^2 - x$
4) $x^3 - x$
5) $x^3 - x^2$
6) $log(x)$
7) $(x - mean(x))/sd(x)$

Feature Extraction

The categorical variables:

Intuitively, it made sense to us that the data channel on which an article was published should make an impact on its popularity. For example, one might expect an article published on a social media channel to have more chance of becoming viral than one that is published on a business channel. The same can be said for the day of the week on which an article is released. It is not unreasonable to suggest that people have more time to read on the weekends and thus an article published on a Sunday maybe more likely to go viral than one published on say a Tuesday. The training data in fact showed that, despite the fact that less articles were published on a Sunday, the percentage of high popularity articles was higher than other days.

We tried to capture this by mapping all the data channel binary variables to two categorical variables. We did the same for the day of the week binary variables. For each data channel (day of the week) binary variable, we divided the number of articles belonging to each popularity class in the training set by the total number of articles released on that data channel. We then multiplied these proportions with increasing weights. For example, for the lifestyle binary variable, we multiplied the proportion of articles in popularity class 1 by 1, the proportion of articles in popularity class 2 by 2, etc. These values are then summed up for each data channel. From these totals, we generate 2 new features:

1) The data channel that has the largest total is assigned a value of 6 and the data channel that has the lowest total is assigned a value of 1.

2) The data channel that has the largest total is assigned a value of 3 and the data channel that has the lowest total is assigned a value of -3.

Any article that does not belong to any of the data channels is given a 0. For the days of the week binary variables, the process is the same except that the values for the first feature range from 1 to 7.

The idea behind these new features is that the first one somehow captures the magnitude of popularity an article reveives by being published on a particular data channel or day of the week, whereas the second one captures whether the publication channel or day of the week is likely to positively or negatively affect the populariyt of an article.

Continuous variable interaction terms:

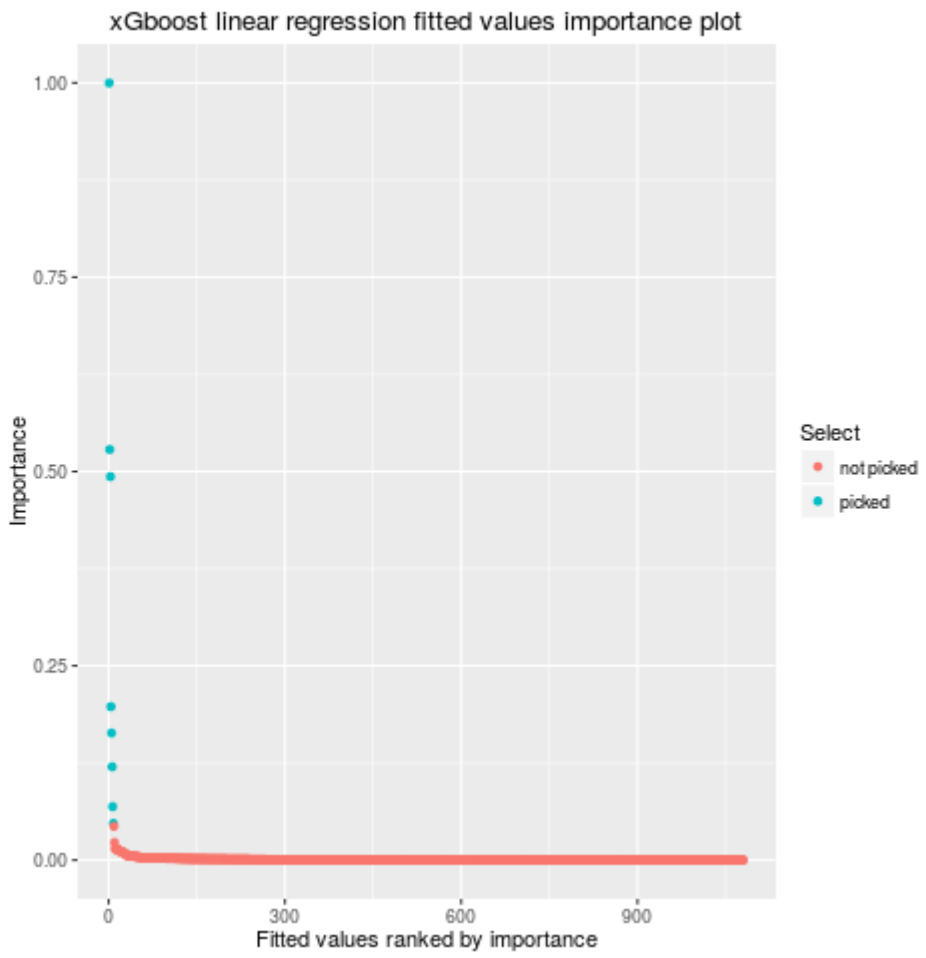
We also considered first order interaction terms between the continuous features. With 34 continuous features in the original dataset, the set of all possible first order interactions terms was very large. In order to select a reasonable subset of these, we looked for terms that discriminated well between classes.

For each potential interaction term we carried out analysis of variance (ANOVA) to determine whether the means of each of the five classes, for that particular interaction term, varied significantly. We used a Tukey test with a significance level of 5% and only accepted the interaction term as a new feature if the null hypothesis (where the null is that two means are the same) was rejected for all combinations between the classes. That is, if the means were statistically significantly different between classes 1-2, 1-3, 1-4, 1-5, 2-3, 2-4, 2-5, 3-4, 3-5 and 4-5, we accepted the new feature. This process yielded 33 new features.

Accounting for ordinality of popularity classes using xGboost linear regression:

An important feature of the dataset in this competition was that the popularity classes were ordinal rather than nominal. We tried to use this fact to our advantage via two different methods. The first method consisted of breaking the multiclass problem into four binary problems and using the random forest algorithm to compute posterior probabilities of popularity classes. This did not yield very good results and is discussed later under the 'Tested Models' section. The second method trialled was to run linear regressions on the dataset, treating the popularity classes as numeric values, and then later use the fitted values from these models as input to a classification model. These fitted value features can be thought of as capturing how close a particular article is to its neighbouring popularity classes. The regression was carried out using the xGboost package in the following manner:

1) We first determined the parameters we wished to tune (max.depth, sampsize, lambda, eta (learning rate) and colsample_bytree) and the ranges over which we wanted to test them.
2) We then tested all possible combinations of these parameters by running a series of nested loops and minimising the RMSE for each regression.
3) The output from this process was 1080 sets of fitted numeric values for the training data ranging from 1.09 to 4.92.
4) With these 1080 new features, we ran the h2o implementation of random forest, using the true popularity classes as the target labels, and extracted the top eight most important variables using the "h2o.varimp()" function. We chose eight, since variable importance reached a plateau at this point, as can be seen in the graph below.



A note on why we decided to use xGboost for this section. It provided us with many parameters to tune and also had a lower Root Mean Squared Error (RMSE) in comparison to other models at initial testing stage.

Feature Selection

After creating a large number of new features, we were faced with the task of selecting which ones we wanted to use for our classification problem. We did this by extracting the variable importance measures produced from gradient boosting classification models. We used the and h2o.varimp() function within the h2o package, using as input an h2o.gbm model. This returns the features ranked in descending order of importance. We also tried using the xgb.importance() function within the xGboost package which takes as input a xgb.train model. We found the latter method to be much more time intensive.

We started with all original, transformed and extracted variables described above, giving us a total of 388 features. We then trialled two differnt methods in order to cut this down.

1) Variable importance using an h2o.gbm model

We iteratively ran gradient boosted models and trimmed off the least important variables until accuracy levels peaked. This process yielded a final features matrix with 50 columns, consisting of:

1) 15 continuous variable interaction terms

2) 4 $x^2 - x$ features
3) The extracted data channel feature with values ranging from 0 to 6
4) 30 features from the original features matrix

2) Variable importance using an xGboost model

As above, except with xGboost models. This returned a final features matrix with 49 columns consisting of:

1) 15 continuous variable interaction terms
2) 3 linear regression fitted values features
3) The extracted data channel feature with values ranging from 0 to 6
4) 30 features from the original features matrix.

Tested Models

h2o.gbm

The h2o implementation of gradient boosting was used on the original features matrix as well as the two features matrices described above.

The parameters that were tuned in the gbm function were max.depth, sampsize, lambda, eta (learning rate) and colsample_bytree. The highest accuracies achieved were:

1. Original Features Matrix - 0.515
2. xGboost 49 Features Matrix - 0.5303
3. h2o.gbm 50 Features Matrix - 0.5283

Note. Although we used xGboost as part of the feature selection process but we didn't use it for any of our submissions. The main reason behind this is that parameter tuning is quite difficult in comparison to the h2o implementation. Results can be unstable and we found it quite difficult to pick up trends between parameter values and prediction accuracy. It is extremely popular for it's prediction capabilities but we ended up with better results from h2o.gbm in comparison to xGboost.

h2o.DeepLearning

We tried implementing a deep learning model using the function h20.DeepLearning from the h2o package. The matrix used was the Final 49 features matrix. The parameters tuned were the following:

1) The number of layers of weights
2) The number of weights in each layer (constant or decreasing with progression layers from input to output)
3) With and without "Dropout"
4) The evaluation metric: "mlogloss" and "misclassification"

The best accuracy we achieved with the tuning of parameters was 0.502.
We decided not to use this model because deep learning requires a vast amount of data and generally works better for less structured datasets. The complexity of tuning the number of weights and layers was another disadvantage with this approach.

Combination of four logistic regressions

In order to try to capture the ordinality of the popularity classes we trialled a technique based on the paper "A Simple Approach to Ordinal Classification" by Eibe Frank and Mark Hall (http://www.cs.waikato.ac.nz/~eibe/pubs/ordinal_tech_report.pdf). We broke the multiclass problem down into four binary problems:

1) Model 1: If an article has a popularity class less than 2, assign it a class of 0. Otherwise assign it a class of 1.
2) Model 2: If an article has a popularity class less than 3, assign it a class of 0. Otherwise assign it a class of 1.
3) Model 3: If an article has a popularity class less than 4, assign it a class of 0. Otherwise assign it a class of 1.
4) Model 4: If an article has a popularity class less than 5, assign it a class of 0. Otherwise assign it a class of 1.

Each of these problems was modelled using the randomForest function from the package of the same name.

For prediction, the posterior probabilities of being in a particular popularity class are as follows: P(class 1) = Probability of being a 0 in Model 1 P(class 2) = Probability of being a 1 in Model 1 - Probability of being a 1 in Model 2 P(class 3) = Probability of being a 1 in Model 2 - Probability of being a 1 in Model 3 P(class 4) = Probability of being a 1 in Model 3 - Probability of being a 1 in Model 4 P(class 5) = Probability of being a 1 in Model 4

This model was only run using the original features matrix and performed very badly. Thus is was discarded.

Final Selected Model

The final model we selected was an h2o.gbm model run on the 50 column features matrix for which we tuned the parameters. The accuracy achieved on the entire test set was 0.52633.