Predicting 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 can predict the popularity of an article on the website mashable.com based on the data available to us.

Analysis of the dataset

There is a substantial class imbalance in the dataset. The percentage of the classes in the training set are 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 raises a significant challenge as the models will frequently predict classes 1 - 3 and will be unable to accurately predict classes 4 & 5.

We have 59 features, out of which most are continuous variables and there are 2 groups of categorical variables:

1) There are a set of categorical variables which are used to denote the data channel of the article. Each feature belongs to atmost 1 data channel. There are a few articles which belong to none of the data classes. In the training dataset, there are 4,658 articles without a designated data channel and in the test set there are 1476. There are a total of 6 channels which are as follows:

    1. Lifestyle
    2. Entertainment
    3. Business
    4. Social Media
    5. Technology
    6. World

2) The other set of categorical variables are used to denote the day of the week on which the article was released. In additional to these 7 variables corresponding to each day, there is a categorical variable "is_weekend" which has a value of 1 if the article was released on a weekend and 0 if it was released on a weekday.

The continous variables can be grouped the following ways

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

- LDA topics: Latent Dirichlet Allocation was used to determine the top 5 relevant topics at the time of the release of the article. They are indexed from 0 to 4 and each of the variables has a score ranging from 0 to 1.

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

Approach to Model Building

Feature Transformations:

The following feature transformations have been done to evaluate their predictive capability. The following operations have been applied to x_{i} for all i that are continuous variables.

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

Feature Extraction has been done in 3 ways.

The categorical variables:

Intuitively, it can be concluded that there is an impact of the data channel of the article and the day of the week it was released in predicting the popularity.

For each day, we took the divided the number of articles belonging to each popularity class by the total number of articles released on that day. Then we multiplied these proportions with increasing weights such that the proportion of the highest popularity (class 5) is multiplied with the highest weight and the lowest popularity is multiplied with the lowest weight. These values are then added up for each day and arranged in a descending order. From this list, we generate 2 new features:

1) The day with the highest score gets a value of 7 and the day with the lowest score gets a value of 1.
2) The day with the highest score gets a value of 3 and the day with the lowest score gets a value of -3.

The same is done with data channels and the features get a range of 0 to 6 and -3 to 3. However, the articles without a data channel mentoined get a value of 0 in both these features.

The reason for the positive and negative features is that we wanted to examine if there is: 1) A varying amount of positive effect of being released in a certain data channel/day. 2) A negative effect of being released in some data channels/days and positive effect on being released on the others.

ANOVA interaction variables:

In our search for informative features to include in our models we consider first order interaction terms between continuous variables. Thirty-three of the original features are continuous, so the set of all possible interactions of these is very large. In order to select a reasonable subset of these, we look for terms that discriminate well between classes.

For each potential interaction term we carry out analysis of variance (ANOVA) to determine whether the means of each of the five classes, for this particular interaction term, vary significantly. We use a Tukey test with a significance level of 5% and only accept the interaction term as a new variable if the null hypothesis (where the null is that two means are the same) is rejected for all combinations between the classes. That is, if the means are 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 accept the new variable. This process yields the 33 new variables.

xGboost linear regression:

The problem that we are dealing with a multi-class classification. Intuitively, we have a reason to believe that this is not a completely categorical response. Our popularity classes are ordinal. If a new set of features predicts with a high ampunt of probability that an article is in class 5, then there is higher possibility of it belonging to class 4 in comparison to class 1.

Keeping this in mind, we decided to use the output classes as numeric values and generate fitted values by running a linear regression.

We considered several models for this section of feature selection but we decided to use xGboost. It provides us with atleast as many/more parameters for tuning in comparison to other models. In addition, it had a lower RMSE in comparison based on a few initial tests done to determine the model.

The following procedure involves using a grid in order to generate fitted values by performing linear regression on the output.

The steps executed are as follows:

1) We determine the parameters we wish to tune and the set of values they will take to use for the XGboost function.

2) The error function being minimized is RMSE and the objective is linear regression.
3) We ran a series of nested loops such that the function is built using all possible combinations of parameters.
4) At each step, the model generated with the parameter combination is used to predict fitted values for the set and these are stored in new a dataframe to be used as the predictors.

Once the whole loop has been executed we will have a total of l columns in the new dataset. Here l is the product of the number of values in each parameter that has been tuned.

In our particular code, we used the parameters max.depth with 4 possible values, sampsize with 6 values, lambda with 3 values, eta(learn rate) with 5 values and colsample_bytree with 3 values.

l was 1080.

Feature Selection

This was done in 2 ways:

1) Manual Selection:

This was done in the beginning of the trimester, various combinations of features were taken and their effects on the accuracy was measured using coefficient values from cv.glmnet and the prediction accuracy from cv.glmnet function.

This was done for the original features, scaled features and log features.

This was a suboptimal procedure mainly because it is extremely time-consuming. The advantage of this method was a very well structured uderstanding of the impacts of the variables and their transformations on accuracy. A couple of clear winner variables were: "is_weekend" and kw_avg_avg.

The final result is summarized in the next section.

2) Using h2o.varimp() and xgb.importance()

The h2o.varimp() function takes h2o.randomForest and h2o.gbm models as an input and returns the features ranked in descending order of importance.

xgb.cv() is used for determining the cross-validated performance for a set of parameters. Once we have determined the best set of parameters, the model is built using xgb,train and the xgb.importance() is used for xgb.train().

The Features used and their derivation:

The main features sets used in the approaches and models described in the next part of this report are as follows:

1) The original features matrix

2) The manual selection features matrix

This resulted in a matrix of 34 features:

1) 2 log transformed features.
2) 7 scaled features.
3) 25 features from the original features matrix.

3) The xgboost linear regression result matrix

As mentioned earlier, this matrix consisted of 1080 features, which are all prediction results for different combinations of parameters. We used a h2o.randomForest model with this matrix as the features and our classes as the output. The top 8 features were selected from this and add to our original features matrix.

4) xgboost 49 features matrix

These features were selected all the possible ones using xgboost.

The matrix consists of:

1) 15 interaction terms.

2) 3 linear regression fitted values.
3) Data Channel feature with values ranging from 0 (Data channel not specified) to 6.
4) 30 features from the original features matrix.

5) gbm 50 features matrix

These features were selected using h2o.gbm.

The matrix consists of:

1) 15 interaction terms.
2) 4 $x^2 - x$ features.
3) Data Channel feature with values ranging from 0 (Data Channel not specified) to 6.
4) 30 features from the original features matrix.

Intial Approaches

The multinom function.

One of the easiest way to approach a multiclass classification problem is applying the multinom function from the nnet pacakge in order to perform multinomial regression.

Doing so on the original features matrix gave us a baseline accuracy of 0.4823

Our goal is to develop a model that can definitively perform better than this one.

The glmnet function

The function cv.glmnet is comparatively much more robust than the multinom function as it performs cross-validation across the dataset and picks the stores the errors for various regularization parameters. It was used exensively as part of the manual selection process

The accuracies of the model built using the cv.glmnet function on the features matrices are as follows:

1) Original Features matrix -
2) Manual Selection Features matrix -
3) xgb 49 features matrix -
4) gbm 50 features matrix - 0.498

randomForest

Initial Submission

The first submission we made to kaggle is the one that currently has the highest accuracy on the leaderboard. We used randomForest with 200 trees and the public leaderboard accuracy is 0.54969

Separate Logistic Regressions

h2o

The main reason for the use of the H2o package in R is the fact that it performs parallel computation making it easier to run relatively complex models in a shorter time. This provides a significant edge while doing cross-validation and parameter tuning.

The following functions from h2o have been used:

```
1) h2o.gbm
2) h2o.randomForest
3) h2o.deeplearning
```

h2o.gbm

The h2o.gbm was used on all 4 of the features matrices that have been described above.

The advantage with the h2o.gbm is that parameter tuning is relatively easy in comparison to xgboost. xgboost is quite unstable and it is difficult to predict the trend between the prediction accuracy and the parameters.

The parameters that were tuned in the gbm function were . . . .

Parameter tuned accuracies for the three features matrices were

1. Original Features Matrix - 0.515
2. Manual Selection - 0.5257
3. XGB 49 Features Matrix - 0.5303

h2o.gbm was used in order for feature selection as well, we began with a feature matrix consisting of all possible transformations, interaction terms and the categorical variable features. Using h2o.varimp() recursively, we selected the best 50 features for prediction based on information gain and the final accuracy (after parameter tuning) was 0.5283.

h2o.randomForest

The h2o.randomForest function was used on 2 of the features matrices described above.

The parameters tuned in h2o.randomForest were . . . .

The advantage that randomForest provided over h2o.gbm were the parameters . . . .

The accuracies for the features matrices after parameter tuning were as follows:

1. Original Features matrix -
2. gbm 50 Features Matrix -

In addition, h2o.randomForest was used for selecting the top features from the xgboost linear regression output matrices.

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. The complexity of tuning the number of weights and layers was another disadvantage with this approach.

XGBoost

We used xgboost as part of the feature selection and extraction process but we didn't use it for any of our submissions. The main reason behind this is that it is quite unstable. It is extremely popular for it's prediction capabilities but we ended up with better results from h2o.gbm in comparison to xgboost.

Our submissions

Submission 1

Submission 2