Regression Techniques for Online News Share Prediction

Rino Di Chiara
Politecnico di Torino
Student id: s314931
s314931@studenti.polito.it

Alessandro Maria Feri'
Politecnico di Torino
Student id: s305949
s305949@studenti.polito.it

***Abstract*—In this report we introduce a possible approach to building a regression model to predict and evaluate online news popularity, specifically the number of shares for each article. Understanding the factors that influence the sharing of news articles on the internet is crucial in today's digital landscape. Accurate predictions can offer valuable insights into information diffusion and inform strategies for news organizations and advertisers.**

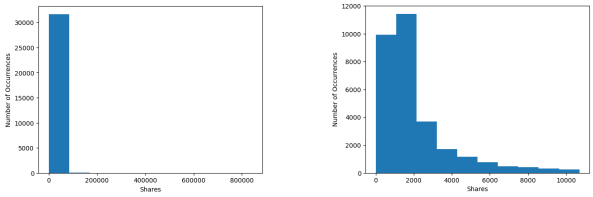
I. Problem overview

Predicting and understanding factors that influence online news sharing on the internet popularity is vital. This project aims to predict shares of described news articles on a dataset divided into two sets:

- Development set: Composed of 31,716 records and used to build our regression model.
- "Evaluation set": Composed of 7,918 records and used to evaluate the performance of the model.

Below you can find a description of the attributes in the dataset (take note that for an easy understanding of the features, we aggregated some of them and the meaning of features is easy to get from the word itself):

- ***URL*** – This attribute does not affect the prediction and represents the URL of the article.
- ***Timedelta*** – It represents the number of days between the article publication and the dataset acquisition.
- ***Tokens*** – This category, concerning the words in the articles, includes the attributes n_tokens_title, $n\_tokens\_content$, n_unique_tokens, $n\_non\_stop\_unique\_tokens$, and $n\_non\_stop\_words$.
- ***Links*** – This category includes the attributes num_hrefs and num_self_hrefs, concerning the links in the articles published by Mashable.
- ***Media*** – This category includes the attributes num_imgs and num_videos, which respectively represent the number of images and videos present in the article.
- ***Keywords*** – This category, concerning the keywords in the articles, includes the attributes $num\_keywords$, kw_min_min, kw_max_min, kw_avg_min, kw_min_max, kw_max_max, kw_avg_max, kw_min_avg, kw_max_avg, and kw_avg_avg.
- ***Self Reference*** – This category includes the attributes $self\_reference\_min\_shares$, $self\_reference\_max\_shares$, and $self\_reference\_avg\_shares$, which respectively represent the minimum, maximum, and average number of shares among referenced articles in Mashable.



(a) before length normalization

(b) after length normalization

Fig. 1: Length Normalization

- ***Weekday*** – It represents the day of the week when the article was published.
- ***LDA Topics*** – This category includes the attributes LDA_00, LDA_01, LDA_02, LDA_03, and LDA_04, which represent the proximity of the article to different LDA (Latent Dirichlet Allocation) topics.

II. Preprocessing

In order to better understand our features and target, we analysed the distribution of the range of shares. As you can see from the Figure 1 there are some outliers in the dataset, so we decided to use the *95-th Percentile* of the total shares sample.

In our ongoing analysis, we have proceeded to examine two key aspects of the dataset: null values and object variables.

Attribute	Non-null values	type
num_vimg	25340 non-null	float64
num_videos	25384 non-null	float64
num_vkeywords	25397 non-null	float64
data_vhannel	31715 non-null	object
weekday	31715 non-null	object

TABLE I: Objects and Missing values

A. Handling Null Value

One essential step in our analysis was the careful handling of null values. We meticulously identified columns with null

values and the result obtained is well defined by the table 1. This step allowed us to understand the extent and distribution of missing data in the dataset. After that, plotting the (Figure 2), we noticed that the distribution of our target compared to the distribution of the null values was following the same distribution of the entire dataset. Under this assumption we can conclude that the null values do not affect in a deterministic manner the values of the shares. For the reason above, instead of creating another feature, we preferred to fill any missing values with zero using the fillna() method. By addressing null values appropriately, we ensured the integrity and completeness of our dataset.

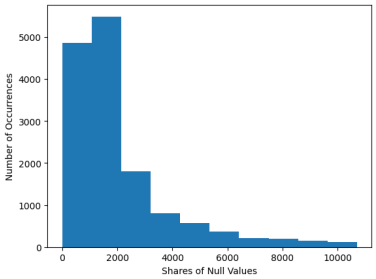


Fig. 2: Shares Null Values

B. Exploring Object Variables

Additionally, we focused on analysing object variables within the dataset. The object features in our dataset are the attributes: weekday and data_channel. We simply manage that by creating a dummy variable which converts our categorical attributes into numerical values to be used in the linear regression model.

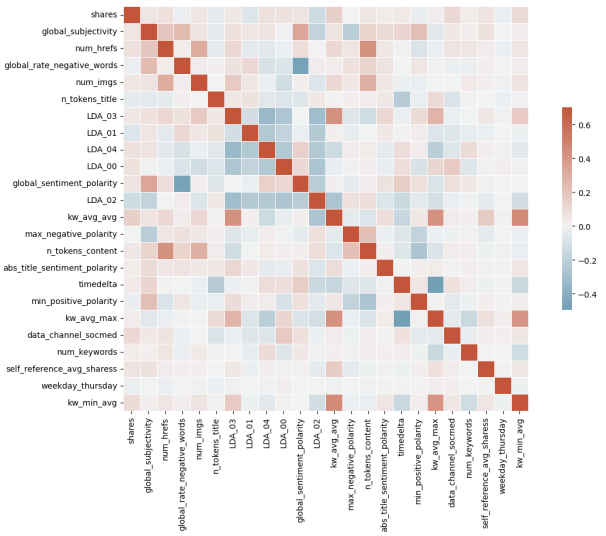


Fig. 3: Final correlation heatmap after pruning the features

C. Data Normalization

Since each attribute uses a different scale of measure, we first normalized those values so that each fill in the range [0,1] using the classic standard scaler $z = \frac{x-\bar{x}}{\sigma}$

Continuing our report, we conducted a feature importance analysis to identify the most influential variables in our predictive model. To identify highly correlated attributes, we calculated the correlation matrix for our dataset and visually inspected the correlation heatmap (Figure 3). Attributes with a correlation coefficient above a predefined threshold (0.5) were considered highly correlated. The number of pruned attributes was equal to 24.

At the end of our data pre-processing phase, in order to address issues related to the non-uniform distribution of our target variable, we employed a data duplication technique. It was carefully executed to prevent overfitting and maintain the integrity of the original dataset.

III. Proposed Approach

A. Model Selection

Among candidate models we focused on:

- A *Linear Regression* model to get a quick start and a kind of baseline to be a comparison to the other models.
- a *Support Vector Machine (SVMs)*. They work by finding a Hyperplane that maximizes the margin between different classes after applying a non linear transformation to the data. In our proposed solution, we decided to not take it into account because of its long training time and its computational cost that makes the tuning harder..
- *Lasso* models encourage the model to select only a subset of the most relevant features and set the coefficients of less important features to zero. By shrinking the coefficients of irrelevant features, lasso models help to reduce overfitting and improve the model's generalization ability.
- *Random forest* works by constructing a multitude of decision trees using random subsets of the training data and random subsets of the features. Each tree in the forest independently makes predictions, and the final prediction is determined by aggregating the predictions of all the trees.
- *Gradient descent boosting (XGBoost)* is based on Random Forest algorithm. The main difference is that it introduces leaf weighting to penalise those that do not improve the model predictability. Both decision tree algorithms generally decrease the variance, while boosting also improves the bias in order to have a better predictions.

B. Hyperparameters tuning

In the report, a thorough analysis of hyperparameters was conducted to optimize the performance of the model. The parameter grid consisted of various hyperparameters, each with specific values to be explored. For the 'max_depth' parameter, a single value of 25 was tested to determine the optimal tree depth. The 'learning_rate' was set to 0.023,

controlling the rate at which the model learns from the data. We paid much attention to this parameter to mitigate the problem of overfitting. In fact, while the other parameters were fixed, we changed the value of the 'learning_rate' until we got the best result. The number of trees in the ensemble, defined by 'n_estimators', was set to 250 to strike a balance between complexity and computational efficiency. To reduce the likelihood of overfitting, the 'subsample' and 'colsample_bytree' parameters were set to 0.5, indicating a subsampling of instances and features.This comprehensive exploration of hyperparameters aimed to find the optimal configuration that maximized the model's performance and generalization ability.

Model	Parameters	RMSE
Linear Regression		2325.10
Lasso	alpha: [0.01, 0.1, 0.5, 1.0, 10.0] fit_intercept: [True, False] max_iter: [1000, 2000]	2324.83
Random Forest	max_depth: [80, 90, 100] max_features: [2, 3] min_samples_leaf: [3, 4, 5] min_samples_split: [8, 10, 12] n_estimators: [100, 500, 1000]	1153.97
XGBoost	max_depth: [10, 20, 25] learning_rate: [0.05, 0.1, 0.02] n_estimators: [150, 200, 250] subsample: [0.5, 0.7, 0.8] colsample_bytree: [0.5, 0.3] min_child_weight: [14, 15, 16] gamma: [2, 5, 7] reg_alpha: [10, 50, 100] reg_lambda: [30, 60, 80]	1465.89

TABLE II: Parameters grid studied

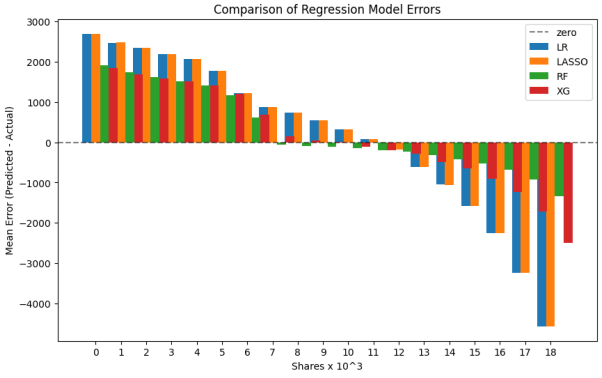


Fig. 4: Comparison among the selected models

IV. Results

As you can see from the Fig.4 the regression model which works better in our test set is the Random Forest regressor, but this is not actually true. Further investigation revealed the presence of overfitting in the Random Forest results. Although the Random Forest model exhibited lower error rates and higher accuracy during training, its performance on unseen data, such as the test set, was compromised due to overfitting. For the reason mentioned above we can conclude that the best performance for our problem regression was obtained by using the XGBoost. Moreover, thanks to the speed of the computation obtained by running the XGBoost we have been able to conduct a deeper analysis of the parameter selection. Given the imbalance in the values of the target variable, we observe that the estimated values tend to be higher for lower values of the shares and lower for higher ones. This can be attributed to the skewed distribution of the target variable, where the majority of the instances fall within the lower range. As a result, the model tends to be more confident in predicting middle target values and less certain in predicting higher target values.

Best Parameters
subsample: 0.5
reg_lambda: 30
reg_alpha: 50
n_estimators: 250
min_child_weight: 14
max_depth: 25
learning_rate: 0.023
gamma: 2
colsample_bytree: 0.5

A. Overfitting Overview

In our regression analysis, we observed that the error on the test dataset is consistently higher than the error on the training dataset. This discrepancy between the two sets of data indicates the presence of overfitting in our model. This means that the model was too complex and starts to memorise the noise and outliers present in the training data, rather than capturing the underlying patterns and relationships. To address this issue, we applied regularization techniques and fine-tuned the model's hyperparameters. By finding an optimal balance between model complexity and generalization, we mitigate the effects of overfitting and improve the model's performance on unseen data.

- Firstly, we applied regularization rules such as L1 or L2 regularization. These techniques introduce a penalty term to the loss function, discouraging the model from assigning too much importance to individual features. By constraining the weights of the model, regularization helps prevent overfitting by promoting a more balanced and robust representation of the data.
- Secondly, we lowered the depth of the decision tree in our regression model. By reducing the maximum depth, we imposed a constraint on the complexity of the tree, limiting its ability to overfit the training data. This adjustment encouraged the model to capture more general patterns and relationships, improving its performance on unseen data.
- Additionally, we employed a technique known as target duplication. This involved creating additional copies of the target variable to balance its distribution. This approach helped to create a more uniform and representative dataset, enabling the model to learn from a broader range of target values.

- Furthemore, we used a subsampling technique which randomly selects a portion of the training instances to build each tree in the gradient boosting ensemble. Reducing the number of training instances of the original data. This technique helps to introduce more randomness and diversity into the model, preventing it from overfitting.
- Last but not least, we regulate the feature subsampling which randomly selects a subset of features for each tree. By restricting the number of features at each split, we aimed to reduce the complexity of the model and mitigate the risk of overfitting caused by excessive feature importance.

V. Conclusion

In conclusion, based on our regression analysis, we have achieved a promising result with a Root Mean Square Error (RMSE) of 5967.27 using the best-performing model, which is XGBoost. The RMSE value indicates the average deviation of our predictions from the actual values of the target variable. The utilization of XGBoost, a boosting algorithm based on decision trees, has proven to be particularly effective in dealing with complexity and non-linearity in the data

References

[1] Radek Silhavy, Petr Silhavy, Zdenka Prokopova, "Analysis and selection of a regression model for the Use Case Points method using a stepwise approach", 2017, Volume 125, 1-14

[2] Michael A. Babya, "What You See May Not Be What You Get : A Brief, Non technical Introduction to Overfitting in Regression-Type Models", 2004

[3] Tianqi Chen, Carlos Guestrin, "XGBoost: A Scalable Tree Boosting System", 2016