Prediction of news popularity by regression

Suyash Singh s307798@studenti.polito.it
Harsh Lalitbhai Vasoya s308347@studenti.polito.it

***Abstract*—In this report, we introduce a possible regression approach for predicting news popularity. In particular, we build a regression model capable of inferring the popularity of the articles given the contributing features. We applied techniques such as linear regression, decision tree and reported scores based on the RMSE (Root mean squared Error) value.**

I. Problem Overview

News popularity prediction is of great value as it has broad ramifications for several academic disciplines, including media, marketing, and communication. Understanding the characteristics that contribute to the sharing of news stories on the internet is essential as the consumption and dissemination of news continue to move to digital platforms. The dataset consists of 39362 instances detailing features of various news articles along with 49 contributing features. The models were trained on development set and final evaluation was made on the evaluation set. The feature that is to be predicted is named 'shares' representing number of shares.

II. Proposed Approach

A. Preliminary data analysis

As a first step we would like to see the correlation between different features of the data set. Since the data contains a lot of features, it would be better if we take into consideration dimensionality reduction. This way we can avoid the curse of dimensionality. Figure 1 reveals the correlation between different features. We can observe that several features are correlated hence we can apply principal component analysis in the following steps. Moreover Figure 2 reveals the description of different features in the data set , For the sake of compactness we depicted some of the features here. we also need to apply some kind of features scaling. We can perform min-max scaling instead of standard scaling since we have included label encoding for categorical features as well. Another aspect of data analysis is to observe the distribution of the instances of the features. Such distribution is shown in the form of histograms in Figure 3. We can see that many of the features have biased distribution, skewed towards some particular instances.

Before applying feature scaling, we need to take into consideration if we have outliers among the features since they can negatively affect the performance of the training models.During this process, we managed to remove 2200 potential outliers. Next step is to apply principal component analysis in order to reduce the dimensions of the data set. We applied PCA considering ratios of explained variance resulting in different number of principal components. Figure 4 reveals

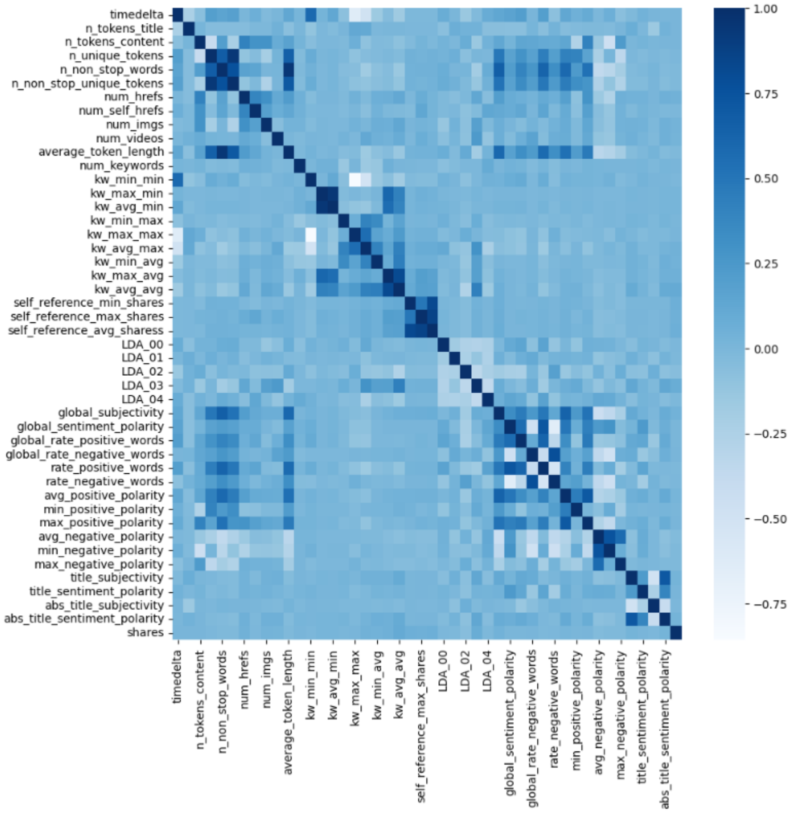
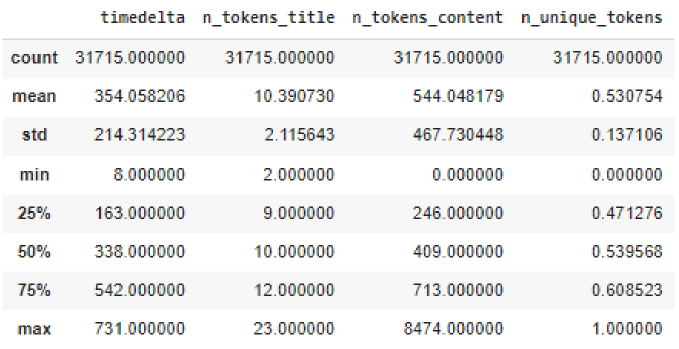


Fig. 1. Heat Map



	timedelta	n_tokens_title	n_tokens_content	n_unique_tokens
count	31715.000000	31715.000000	31715.000000	31715.000000
mean	354.058206	10.390730	544.048179	0.530754
std	214.314223	2.115643	467.730448	0.137106
min	8.000000	2.000000	0.000000	0.000000
25%	163.000000	9.000000	246.000000	0.471276
50%	338.000000	10.000000	409.000000	0.539568
75%	542.000000	12.000000	713.000000	0.608523
max	731.000000	23.000000	8474.000000	1.000000

Fig. 2. Feature Description

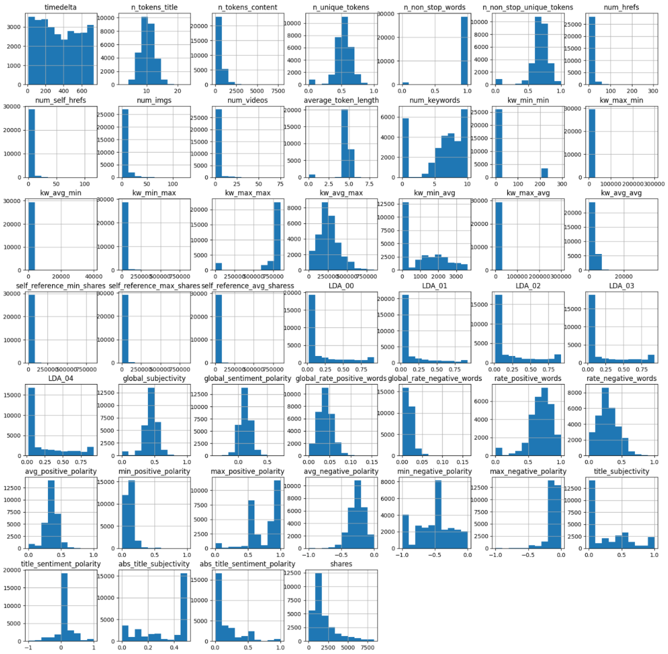


Fig. 3. Feature Distribution

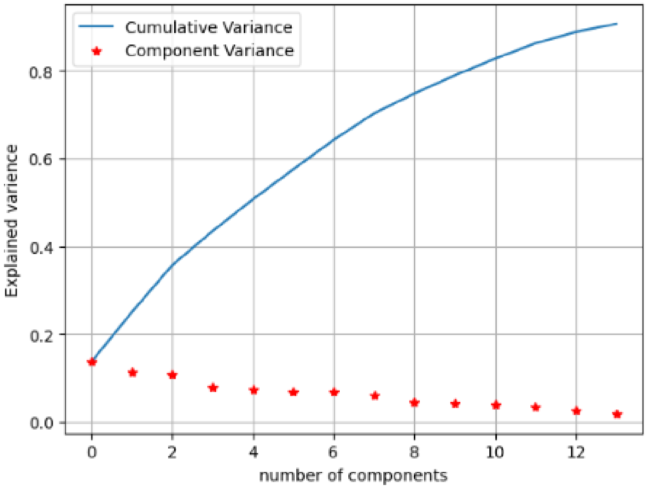


Fig. 4. Explained Variance

principal components capturing 90% of the total variance of the data set.

B. Preprocessing

We make some considerations based on the development set. Firstly, the features contain missing values. As these missing values belong to features with data types as float, a simple way is to replace them with a 0. The dataset also contains certain categorical features such as weekday and data channel. Thus we need to convert them in some numerical form to extract useful information. One possible way is label encoding that transform each instance into corresponding numerical label. The 'url' column contains web addresses,

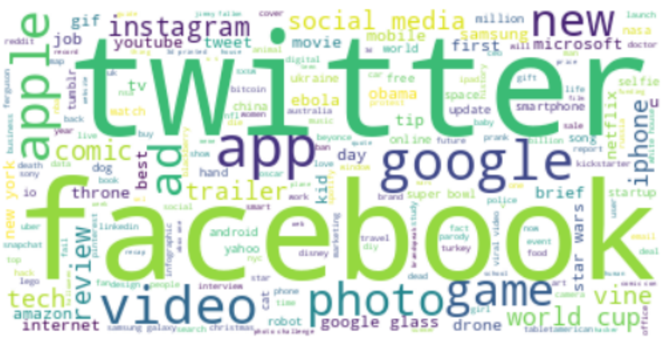


Fig. 5. Word Cloud

This column contains important information such as date and metadata regarding the article. We will need to process this data using natural language processing techniques such as TF_Idf vectorizer.
We can apply TF-Idf technique to convert the text into corresponding numerical values. This algorithm involves the computation of the term frequency (TF) of each token within its respective document. Then it computes the inverse document frequency (IDF). While the TF gives an idea of the weight of a token within a document, the IDF is used to find its significance among the entire collection of documents (i.e. our reviews). Combining the definitions of term frequency (TF) and inverse document frequency (IDF), produces a composite weight for each term in each document. The TF-IDF weighting scheme assigns to a term t a weight in the document d given by:

$$TF - IDF_{t,d} = TF_{td} X IDF_t \tag{1}$$

In addition, we can visually display the frequently appearing words in the form of a word cloud. This gives us more insights on the kind of vocabulary being used in the dataset. Figure 5 displays the word cloud regarding the metadata extracted from the 'url' feature.

C. Model selection

The following algorithms have been tested:
Linear regression: Linear regression finds the best fit line to minimize the residual sum of squares between the observed targets in the dataset, and the targets predicted by the linear model. In order to apply linear regression we make the following assumptions:

1) The relation between the input and output is assumed to be linear.
2) The input features and output variables are assumed to be free of outliers.
3) The algorithm tends to over fit on data if the features involved are dependent.
4) For better performance, Data scaling is required so that all the features are in the same range of values.

LASSO: Lasso regression is a type of regularization. It is chosen over regression techniques for making more accurate predictions. The process by which data values are shrunk towards a middle point known as the mean is known as shrinkage and is used in this model. Lasso Regression employs L1 regularization. It is utilized when there are several characteristics since it automatically selects features. By applying a penalty to these coefficients, the basic goal of Lasso Regression is to identify the coefficients that minimize the error sum of squares.

Ridge Ridge is yet another technique for regularization. Any data that exhibits multicollinearity can be analyzed using this model tuning technique. This technique carries out L2 regularization. This regression technique is expected to penalize the dependent features.

Decision Tree: Decision tree is a straightforward machine learning algorithm. The goal is to build a model that predicts the value of a target variable by learning fundamental decision rules from data features. Decision trees are used in both classification and regression tasks.

Random forest: This algorithm deploys multiple decision trees to make predictions. Thus avoiding the problem of over fitting and is less prune to the *curse of dimensionality* since it trains each decision tree only on a subset of features and instances . The performance of a random forest is proportional to the number of estimators (up to a certain point) and can be tuned via grid search. Random forests,like decision trees, work on one feature at a time, so normalization is not necessary. We chose random forest for its good performance in the general for regression tasks and the fact that it suffers less from dimensinality problem.

D. Hyperparameter tunning

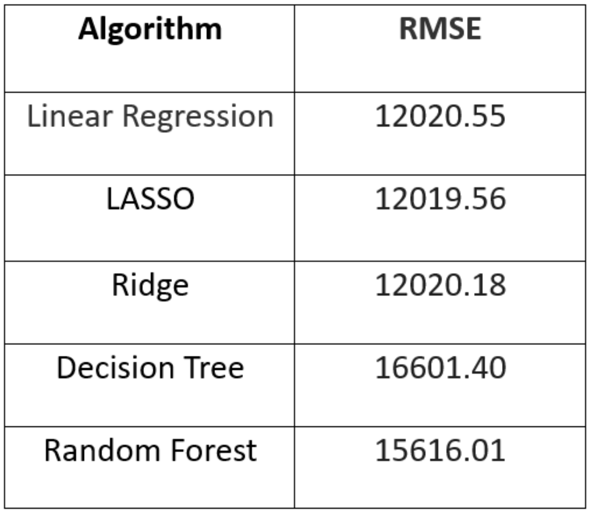
In order to improve the performance of these algorithm we can tweak the parameters of the algorithms. For example we tested the decision tree with different value of the max_depth. This can be done easily by using grid search. Grid search also allows cross validation which helps avoid over fitting. We managed to improve the RMSE score from 16601.40 to 15616.01.

III. Results

We trained the above mentioned models and tested the models using train test split. We used a test size of 0.33 of the dataset. The metrics used for evaluation in all cases is RMSE (root mean square error). The results obtained from running these tests are depicted in the figure 6.

IV. Discussion

We can observe that linear regression has better performance overall. On the other hand random forests perform better then decision tree. This was expected as random forest contains an ensemble of decision tree and the result is obtained by majority voting. In case that principal component analysis is used as part of the pipeline, we have less explainability of the original features for example in case of random forest applied to original features, we can obtain the feature importance revealing the significance of each feature in the data set.



Algorithm	RMSE
Linear Regression	12020.55
LASSO	12019.56
Ridge	12020.18
Decision Tree	16601.40
Random Forest	15616.01

Fig. 6. Results

References

[1] Manning, C.D.; Raghavan, P.; Schutze, H. (2008). ”Scoring, term weighting, and the vector space model”

[2] Jonathon Shlens, A Tutorial on Principal Component Analysis.

[3] Yan, Xin (2009), Linear Regression Analysis: Theory and Computing.