Real-time Popularity Prediction of Online News Articles

Kannan Chandrasekaran
Dept. Computer Science
Indian Institute of Technology
Hyderabad, India
cs18mds11001@iith.ac.in

Sandeep Chopra
Dept. Computer Science
Indian Institute of Technology
Hyderabad, India
cs18mds11006@iith.ac.in

Venkat Nithiyanandhan
Dept. Computer Science
Indian Institute of Technology
Hyderabad, India
cs18mds11029@iith.ac.in

ABSTRACT

In the era of internet, a user can access information through their mobile gadgets anytime and anywhere. Digital journalism (aka Online journalism) is a contemporary form of journalism where editorial content is distributed via the Internet in combination of text, audio, video, and other interactive forms. The number of 'shares' of a news article indicates how popular a news is. In this project, we intended to find a best model and set of hand-crafted features to predict the popularity of an online news article in real-time using Natural Language Processing and Advanced Machine Learning techniques. Our data is from Mashable - a well-known online news platform. We analyzed the html and text contents of articles in the training set to derive various features. These features were fed to Machine Learning algorithms to train the models.

We implemented and benchmarked 13 algorithms from traditional machine learning and modern deep learning techniques. We explored various regression techniques such as linear, non-linear and ensemble. We created a robust feature engineering and feature stitching pipeline for a real-time prediction – given a URL, pipeline crawls the web, gets the content, creates features and predicts the popularity. Gradient Boosting and Linear regression methods turn to be best models for prediction. This work can be generalized and used across digital journalism, online blogs, and social media to predict popularity before the publication.

KEYWORDS

Online News, Digital Journalism, NLP, Machine Learning, Deep Learning, Real-time Popularity prediction, Topic Modeling, Word2Vec, Clustering

1 Introduction & Problem Setup

Gone are the days people had to wait for newspapers or near radios to know the news. With the advent of internet and the advancements in mobile communication technologies, instant news is available at the palm of everyone. This created new breed of news outlets - online news portals. The popularity of the online news outlets and the individual articles are always an interesting problem to predict. The popularity of an article can be measured using many factors, like, number of times the page is read, average time spent by a user on reading the article, number of times the article is referred in internet. One such key factor for measuring the popularity is "Shares" - number of times, an article is shared by its readers. The popularity of an article can be derived in real-time by predicting its shares based on the textual content and other related features from empirical observations.

In this project, we used Mashable.com dataset. Mashable.com is a well-known online news outlet, with hundreds of articles newly published on weekly basis. With thousands of articles available online from the past decade, it is one of the major sources of news archives. The given dataset contains URL, shares count and 17 raw numerical features for selected articles.

We web-scrapped the HTML content of the articles with the URLs provided in the dataset. This HTML content was then preprocessed and features were extracted using various Natural Language Processing methodologies and algorithms. With close to 200 features generated from the data, we analyzed the features, checked the importance of them and isolated features which make higher impact on the target variable, "shares".

These features were then used to build and train various Machine Learning and Deep Learning models. All the models were finetuned by adjusting model specific hyperparameters. The outputs of the models were compared with available target values on both Training Set and Test Set to check the performance of each of the model. In this exercise we found XGBoost, Gradient boosting and Linear regression algorithms were the most efficient. This paper details the feature engineering we performed, benchmarking of different regression algorithms and the conclusion based on these steps.

Though this work is based on given Mashable.com, the learnings and observation of it are more general in nature and hence it can be generalized for other such problems in digital media.

2 Data Collection & Statistics

The given dataset contains 7795 rows of data. Each row contains Id, URL, Shares count and 17 features associated with the URL. We analyzed the given data and found that the 17 features are not sufficient for accurately predicting the shares. The efficiency of the

ML models was not satisfactory if we build the prediction model by using the limited features provided.

Hence, we decided to dive deep into the robust Feature Engineering of the independent variables. This includes, re-engineering the existing features (raw 17), adding new Numerical, NLP and Date based features, which are explained in detail in the "Experiments" section. For this feature engineering activity, information in the given dataset is not sufficient. Hence, we collected more textual information such as title, content and meta keywords of the given articles.

2.1 Web Content Mining (Crawling):

We crawled the Mashable.com website and extracted the full HTML content of every news article URL iteratively using *beautifulsoup* (an open source library in Python). From this HTML content, we extracted the following for every article; Title, Meta Keywords, text of article's content, Images & Videos count inside the article, URL links to other articles and the published timestamp of the article. This initial data collection activity, though time consuming, helped us to come up with various features related to the article.

Here are some statistics about the data:

• No. of articles	7795
• Avg shares per article	3028
• Latest article	2014/12/27
• Oldest article	2014/09/01
• Size of crawled data	531 MB
• HTML size of articles	41 to 370 kb
• Article content size	70b to 32kb
• Avg no. of tokens	646
• Avg no. of unique tokens	288
• Avg token length	~4
• Avg keyword count	7

More details on the features of the data, their statistics and their relative importance to predicting the shares are discussed in the Feature Engineering section.

2.2 Data landscape understanding:

We analyzed data landscape which gave more insights such as distributions and correlations about various features. We also checked for data quality of the dataset and there were no missing values. We observed skewness in target variable. In the 'Experiments' section we explained scaling and handling outliers.

We found some interesting correlation of input variables with target variable (example articles with 1 image are more popular than the articles with higher number of images, articles without a video are more popular than articles with one or more videos embedded).

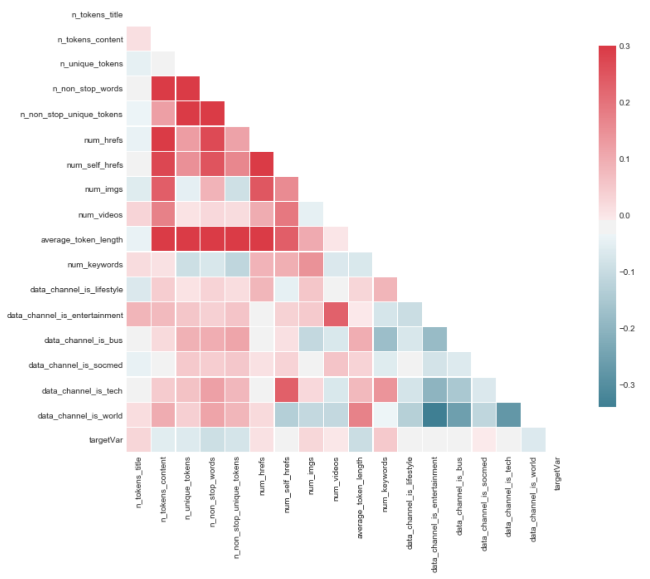


Figure 1 Correlation heatmap

3 Machine Learning Algorithms (Methodology)

In this project our goal is to predict the shares a news article would get, hence this is clearly a regression problem. Regression analysis encompasses a large variety of statistical methods to estimate the relationship between input variables and their associated features. We experimented traditional machine learning approaches and modern deep learning neural network methods.

3.1 Traditional ML Regression techniques[5]

As we were unsure about the relationship between the input and target variables, we tried various types of regression analysis in traditional machine learning.

3.1.1 Linear Regression Algorithms

Linear regression is a linear approach for modeling the relationship between the criterion or the scalar response and the multiple predictors or explanatory variables. Linear regression focusses on the conditional probability distribution of the response given the values of the predictors. The formula for linear regression is:

$$Y' = bX + A$$

Ridge regression is a technique for analyzing multiple regression data. When multicollinearity occurs (independent variables are highly correlated), least squares estimates are unbiased. A degree of bias is added to the regression estimates, and a result, ridge regression reduces the standard errors. The formula for ridge regression is

$$\beta = (X^T X + \lambda * I)^{-1} X^T y$$

$$= \underset{\beta \in \mathbb{R}^p}{\operatorname{argmin}} \underbrace{\|y - X\beta\|_2^2}_{\text{Loss}} + \lambda \underbrace{\|\beta\|_2^2}_{\text{Penalty}}$$

Lasso regression analysis method that performs both variable selection and regularization. Lasso regression uses soft thresholding. It selects only a subset of provided covariates for use in the final model. Lasso regression is

$$N^{-1} \sum_{i=1}^{N} f(x_i, y_i, \alpha, \beta)$$

$$= \underset{\beta \in \mathbb{R}^p}{\operatorname{argmin}} \underbrace{\|y - X\beta\|_2^2}_{\text{Loss}} + \lambda \underbrace{\|\beta\|_1}_{\text{Penalty}}$$

ElasticNet regression is a regularized regression method that linearly combines the penalties of the lasso and ridge methods. The penalty function is

$$\|\beta\|_1 = \sum_{j=1}^{p} |\beta_j|$$

$$\hat{\beta} = \underset{\beta}{\operatorname{argmin}}(\|y - X\beta\|^2 + \lambda_2\|\beta\|^2 + \lambda_1\|\beta\|_1).$$

3.1.2 Non-Linear Regression Algorithms

Linear algorithms rely on the assumption of linear relationships between the independent and dependent variables. In real world settings, this assumption is often difficult to meet. Hence, we tried the below nonlinear techniques as well.

Decision Tree Regression observes features of an object and trains a model in the structure of a tree to predict data in the future to produce meaningful continuous output.

KNN Regression: K nearest neighbor is a simple algorithm that stores all available cases and predict the numerical target based on a similarity measure (e.g. distance functions). KNN regression calculates the average of the numerical target of the K nearest neighbors.

Support Vector Regression: Underneath Support vector machine principles in regression context, SVR gives us the flexibility to define how much error is acceptable in our model and will find appropriate line (or hyperplane in higher dimension) to fit the data.

3.1.3 Ensemble Regression Algorithms

Random Forest Regressor is an ensemble learning method for regression that operate by constructing a multitude of decision trees at training time and outputting the mean prediction

Extra Trees Regressor: Extra tree methods stands for Extremely randomized trees) with the main objective further randomizing tree building in the context of numerical input features.

AdaBoost Regressor: Adaptive boosting is a technique of combining multiple weak learners to build a strong learner. An AdaBoost regressor is fitting a regressor on the original dataset and then fits additional copies of the regressor on the same dataset but where the weights of instances are adjusted according to the error of the current prediction. As such, subsequent regressors focus more on difficult cases.

Gradient Boosting Regressor: Gradient boosting relies on the intuition that the best possible next model, when combined with previous models, minimizes the overall prediction error. The key idea is to set the target outcomes for the next model in order to minimize the error.

XGBoost Regressor: is an optimized gradient boosting library to be highly efficient, flexible and portable. It implements machine learning algorithms under the gradient boosting framework.

3.2 Deep Learning technique

Deep learning methods[5] based on artificial neural networks with representation learning. Learning can be supervised, semi-supervised or unsupervised. In our project we used deep learning for regression problem. A deep neural network with multiple layers between the input and output layers finds the correct mathematical manipulation to turn the input into the output, whether it be a linear or non-linear relationship.

Regression only works well if the regression equation is a good fit for our data. A neural network is a flexible model that adapts itself to the shape of our data. So it can dynamically pick the best type of regression (linear, logistic, polynomial, etc.,) and if it is accurate enough, we can add hidden neuron layers to make the model more complex and improve its prediction power. Neural network works without scaling the independent variable and hence reduces the preprocessing efforts.

3.3 Evaluation Metric

Evaluation metric helps to understand how the machine learning algorithms performs or to choose best performing algorithm when we benchmark many algorithms. As we have regression problem, we decided to go with *Root-mean-square Error (RMSE).*

RMSE of predicted values y_t' for times t of a regression's dependent variable y_t, with variables observed over T times, is computed for T different predictions as the squared root of the mean of the squares of the deviations

$$RMSE = \sqrt{\frac{\sum_{t=1}^{T}(y_t' - y_t)^2}{T}}$$

RMSE is always non-negative and a value close to zero would indicate perfect fit to the data. Important note is RMSE is sensitive to outliers.

4 Experiments

We created a robust ML model benchmarking with 12 regression algorithms (linear, non-linear and ensemble techniques) including deep learning. We predicted the number of shares with raw 17 features. We observed from the plot that all regression models were underfitting as we had limited number of features hence the model got high-bias and low-variance which is unreliable.

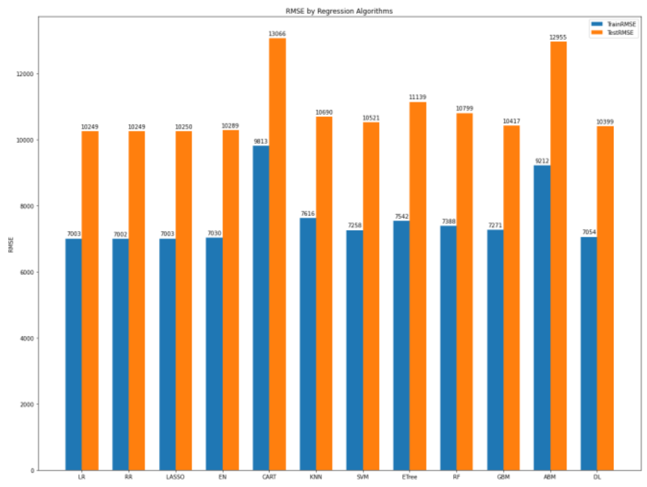


Figure 2 ML model benchmarking with raw features

4.1 Feature Engineering:

We understood from the given 17 raw features none of the regression model able to provide better predictions. Hence, we decided to create additional features using advanced Natural language processing that best represent the news articles. The following are the various features that we created, later we ran feature importance scoring to select required features to be used to build ML models.

During the data exploratory process and subsequent experiments phase, we found few discrepancies in the data. For few entries, the data_channel, href and images count were wrongly updated. Also, for new URLs for inference, we need to generate these 17 features. For these reasons, we re-engineered the 17 features.

Keyword Statistics: [3 Features] From the htmls, we extracted the meta keywords of all the entries. These keywords were used to generate variety of numerical features. The RAW features contain 'number of keywords' as one of the features. For extracting the keywords, beautifulsoup libarary is used[1].

Keywords popularity - For each keyword, average shares it received were calculated and based on it, for the given article, based on all the keywords it has, average, min and max number of shares were predicted.

Example:
For an article if the keywords are ["north-korea, uncategorized, us-world, world, detainees, matthew-miller, jeffrey-fowle"], we get features like, [kw_avg_avg:1696, kw_min_avg:879, kw_max_avg: 2224]

Day of the week: [8 Features] Extracted the weekdays from the date in the url and it turned out to be an important feature. Also extracted whether the day is weekend.

Named Entity Recognition Types: [18 Features] NER is used to analyze the named entities in the given document. Analyzed Named Entities in each document and calculated the count of entities for each type of named entities (Person, Org, Cardinal etc). This provided 18 new features about the article being analyzed. Used spacy package for extracting NER [2].

Doc2vec Based Features: [40 Features] Doc2Vec models[6] create numeric representation of the text data(https://rare-technologies.com/doc2vec-tutorial/). Numerical representation of various components of the article are created. For each component, 100-dimensional vector is calculated. As these add multiple 100 features, to reduce the number of features, from each set PCA of 10 Principal Components were extracted. As part of the Doc2Vec features, 10 PCA for each of these components of the article are calculated; 1. Entire Content 2. Keywords 3. All NER entities in the article 4. Top ten most common NER entities; adding 40 new features to the list. Gensim doc2vec package was used to create the doc2vec models.[3]

Example: if the most common NER entities of an article is found as " Thompson, Meryl Streep, Esperanza, Tan, first, second, Broadway, Heartburn, Ephron, Meryl" then the 10 principal components from the doc2vec model will be [11.2333, 10.7709, 1.75019, 0.7204, -0.5274, 0.5728, 0.6861, -0.8406, 0.6366, 2.2191]

Subjectivity & Polarity: [4 features] We thought that sentiment and subjectivity of the news article's textual data (title and content) might influence the number of shares an article would get. Hence, we created Sentiment polarity & Subjectivity using the unsupervised techniques. Sentiment polarity ranges from -1 to 1, implies negative to positive sentiment respectively and Subjectivity changes from 0 to 1. We used the open source library called TextBlob for this purpose.

Clustering: [8 Features] We also created an unsupervised clustering using Kmeans clustering algorithm and used the predicted cluster as one-hot-encoded features. We used news articles content and then created vector representation using TF-IDF (term frequency, Inverse document frequency) method and used this vector for clustering. The below plot shows the selection optimum number of clusters based on silhouette score. Silhouette score is an indicator that shows how the clusters are separated. We decided to go with 8 clusters as we get max silhouette score.

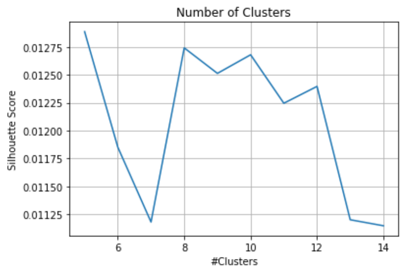


Figure 3 Optimum number of Clusters

We manually cross validated the clusters by reviewing the top-N tokens belong to each cluster. We found that clusters are nicely formed, example [hospital,virus,..] are in a cluster1, [travel, flight,..] are in cluster 2, [android, gadgets…] are in cluster3 and so on. Few examples below

Cluster 0:	Cluster 2:	Cluster 3:	Cluster 7:
movie	app	ebola	game
marvel	google	health	summer
wars	facebook	virus	nba
star	mobile	disease	kid
disney	apps	outbreak	sports
trailer	amazon	hospital	football
avengers	social	_	championship
_	_	_	_

Topic modeling: [45 Features] We created additional features through topic modeling techniques. Topic modeling is an unsupervised learning approach to clustering documents, to discover topics based on their contents. It is similar to how k-means algorithm and expectation-maximization work. We created topic modeling features on news article's title, content and keywords. We used the below topic modeling methods with 'number_of_components'=5

- LDA – Latent Derilicht Analysis: is a probabilistic model. It used two probability values P(word | topics) and P(topics | documents) to assign clusters.
- NMF – Non-negative Matrix Factorization: is a linear algebraic method, that factors high-dimensional vectors into low-dimensionality representation. Similar to principle component analysis (PCA).
- LSA – Latent Semantic Analysis: also known as truncated SVD. Objective of LSA is reducing dimension. The idea is that words will occurs in similar pieces of text if they have similar meaning.

4.2 Feature Stitching & Selection:

We generated multi-faceted features Numerical statistics, Word2Vec based principle components, Clustering, Topic modeling and Polarity/Subjectivity. We created model to stitch all these features and filter the important features to train the Machine learning algorithms. Overall, we created ~120 features in addition to the raw 17 features. We knew that all 137 features will not be required to train a ML model. We tried 3 different approaches to understand feature importance and to select features.[4]

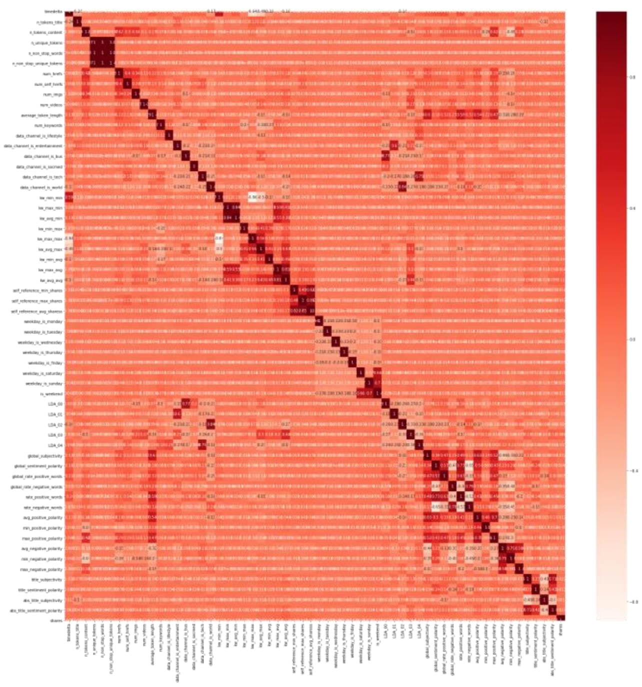
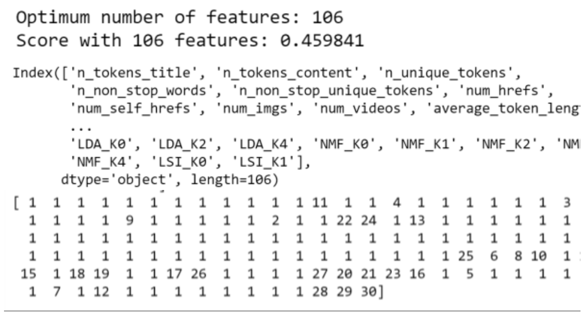


Figure 4 Correlation heatmap with all features

Filter Method / Correlation method: we filter and select only relevant features. The filtering is done using correlation matrix (Pearson correlation). We first plotted the correlation heatmap and observed the correlation of independent variables with the target variable (shares). We decided to select only features that have minimum 0.5 correlation (absolute value).

Wrapper Method: needs one machine learning algorithm and uses its performance as evaluation criteria. We fed all the features to the selected machine learning model and based on model performance we include/exclude the features. There are different wrapper methods, we tried Backward Elimination and Recursive Feature Elimination.

- *Backward Elimination:* we fed all features to the model first. We check the performance of the model and then iteratively we removed the worse performing features one-by-one till the overall performance of the model comes in acceptable range. Here we used OLS (Ordinary Least Squares) model and 90 features were selected.
- *Recursive Feature Elimination (RFE)* method works by recursively removing attributes and building a model on those attributes remain. Here we took Linear Regression and Lasso model with 137 features and RFE gave the below feature ranking and selected 106 features.



```
Optimum number of features: 106
Score with 106 features: 0.459841

Index(['n_tokens_title', 'n_tokens_content', 'n_unique_tokens',
       'n_non_stop_words', 'n_non_stop_unique_tokens', 'num_hrefs',
       'num_self_hrefs', 'num_imgs', 'num_videos', 'average_token_leng
       ...
       'LDA_K0', 'LDA_K2', 'LDA_K4', 'NMF_K0', 'NMF_K1', 'NMF_K2', 'NM
       'NMF_K4', 'LSI_K0', 'LSI_K1'],
      dtype='object', length=106)
[ 1  1  1  1  1  1  1  1  1  1  1  1 11  1  1  4  1  1  1  1  1  1  3
  1  1  1  1  9  1  1  1  1  1  2  1  1 22 24  1 13  1  1  1  1  1  1
  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1 25  6  8 10  1
 15  1 18 19  1  1 17 26  1  1  1  1 27 20 21 23 16  1  5  1  1  1  1
  1  7  1 12  1  1  1  1  1  1  1  1 28 29 30]
```

Embedded Method: are iterative in a sense that takes care of each iteration of the model training process and carefully extract those features which contribute the most to the training for a particular iteration. Here we used Lasso regularization, if the feature is irrelevant, lasso penalize its coefficient to 0. Hence features with coefficient=0 will be removed,

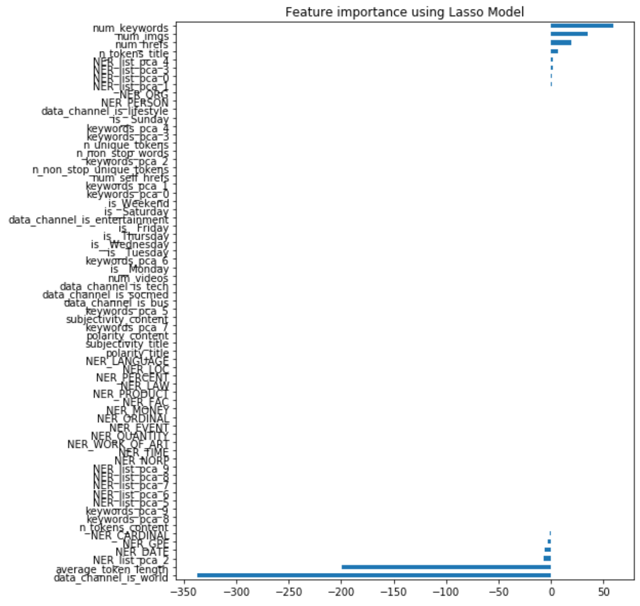


Figure 5 Feature Importance using Lasso

After exploring all the three techniques we finalized that Recursive Feature Elimination best suited for this problem. Final selected features are 106 using RFE method and Lasso as base estimator.

4.3 Scaling Features & Handling Outliers

Some ML algorithms perform better if we scale or normalize the features as it helps to reach minima or convergence quickly. We performed standard scaling on top of numerical features (excluding one-hot encoded features) before feeding the data into ML algorithms. Same scaling is performed before inference as well. The general method of calculation is to determine the distribution mean and standard deviation for each feature. Next, we subtract the mean from each feature. Then we divide the values (mean is already subtracted) of each feature by its standard deviation.

$$x' = \frac{x - \mu}{\sigma}$$

Where x is the original feature vector, μ is the mean of that feature vector, and σ is its standard deviation

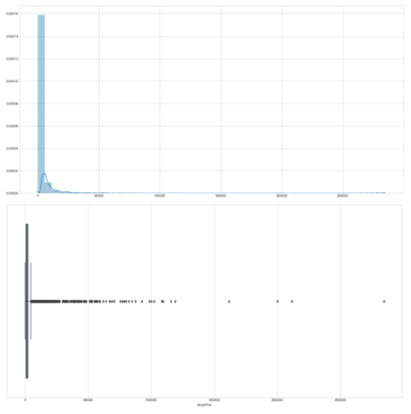


Figure 6 Distribution and Box plot with outliers

We observed skewness and outliers in the target variable (number of shares). We found that very few values ~1% of entire training sample are above 25000 shares. Remaining 99% percentage of the data is below 25000 shares. To find outliers threshold we used Inter Quartile Range (IQR) mechanism. Also, we explored the end-to-end ML pipeline without removing the outliers, however this resulted in poor prediction. Hence, we clipped the values of target variable "shares" to 95th percentile from IQR. Once this value is achieved it means article is very popular. It also helped to take care of skewness in the data.

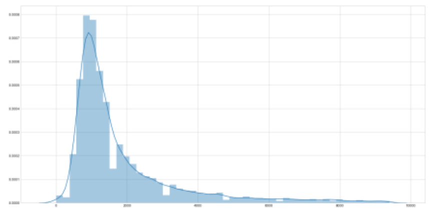


Figure 7 Distribution after outlier removal

This figure clearly shows that the distribution is improved compared to the unclipped raw data with outliers.

4.4 ML Model Benchmarking:

Once we explored existing data, engineered additional features, selected relevant features, normalized the features and handled outliers in target variable, we fed the new dataset to ML model benchmarking pipeline with 10-fold cross validation, that helps to address underfitting problem faced with limited raw 17 features. The below plot shows the significant improvement in model performance. The best (lowest) RMSE value achieved in Train and Test are 1615 and 1618 respectively.

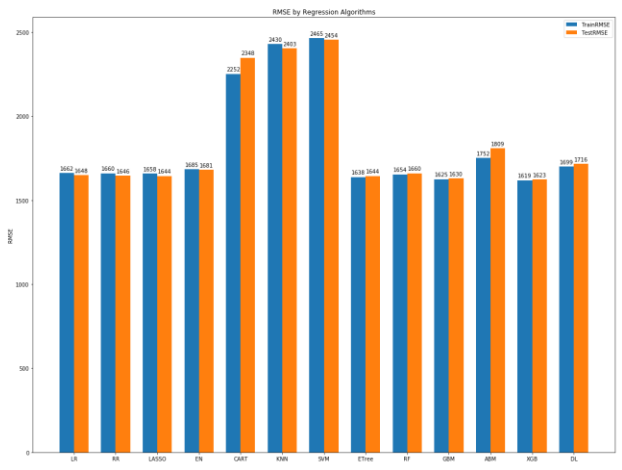


Figure 8 ML model benchmarking with all features

Model		Train (RMSE)	Test (RMSE)
XG Boost Regressor	XGB	1619	1623
Gradient Boosting Reg.	GBM	1625	1630
Ridge	LASSO	1658	1644
Extra Tree Regressor	Etree	1638	1644
Lasso	RR	1660	1646
Linear Regression	LR	1662	1648
Radom Forest Regressor	RF	1654	1660
ElasticNet	EN	1685	1681
Deep Learning NN Reg.	DL	1699	1716
Adaptive Boosting Reg.	ABM	1752	1809
CART Regression Trees	CART	2252	2348
KNN Regressor	KNN	2430	2403
Support Vector Regressor	SVM	2465	2454

The table is sorted with best performing models at the top.

4.5 ML Model Tuning:

We tuned ML models with their respective hyperparameters to improve the performance of the model using grid search approach.

Algorithm	Hyperparameters
Deep Learning	Optimizer: Adam, learning rate: 0.001, momentum: 0.2, #hidden_layer: 2, 'initialization': 'uniform'
ElasticNet	Apha=3, l1_ratio=1.0
GBM	'n_estimators'= 100, features=auto
XgBoost	booster="gbtree", n_estimators=75

4.6 Realtime Prediction:

We wanted to produce a callable API for Real-time prediction of popularity of any given url / news article. We built the real-time prediction pipeline that is capable of predicting number of shares dynamically. We dumped all the trained ML models during training phase and loaded these models in real-time inferencing on the fly. This pipeline has all components starting from data extraction, feature engineering and prediction.

In prediction pipeline notebook, we are generating every feature from unknown url which we built as part of experiment however we only used the features at real time for which model is trained on. Real time prediction is done very quickly for any URL even not known to the model during training time. Ensemble of best performing algorithms is used to predict results at real time.

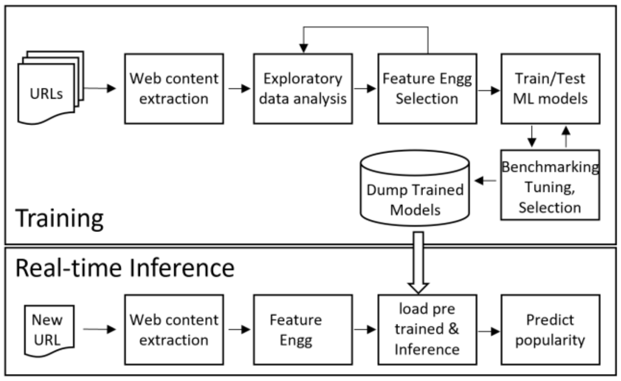


Figure 9 End-to-end real-time prediction pipeline

Conclusion

We observed from the comprehensive ML model benchmarking exercise that Linear (LR, LASSO, Ridge & ElasticNet) and Embedded regression algorithms (Boosting, Tree based regressors) were good for this problem.

Non-Linear regression algorithms (CART, KNN, SVM) did not fit well. This means for our problem, the relationship between independent variables and target variable may be linear.
The table in ML model benchmarking shows the order of ML algorithms based on their performance in Testing / validation.

Overall Gradient Boosting algorithms work the best as it relies on the intuition that the best possible model, when combined with previous models, minimizes the overall prediction error.

The exercise emphasized the importance of natural language processing in these kind of problem domains. With the help of NLP based features (such as word2vect, clustering, polarity & subjectivity, etc.,) we were able to improve our prediction capabilities considerably (from initial RMSE of 10K to 1.6K) This means NLP features gave good representation / signals about the news articles and helped to predict the number of shares more accurately.

REFERENCES

[1] Hajba, Gábor László (2018), Hajba, Gábor László (ed.), "Using Beautiful Soup", Website Scraping with Python: Using BeautifulSoup and Scrapy, Apress, pp. 41–96, doi:10.1007/978-1-4842-3925-4_3, ISBN 978-1-4842-3925-4, retrieved 2020-04-11.

[2] " Industrial-Strength Natural Language Processing - spaCy". spacy.io. Retrieved 2020-05-01.

[3] Řehůřek, Radim (2011). "Scalability of Semantic Analysis in Natural Language Processing". Retrieved 27 May 2020. "my open-source gensim software package that accompanies this thesis".

[4] Guyon, Isabelle; Elisseeff, André (2003). "An Introduction to Variable and Feature Selection". JMLR. 3.

[5] Bishop, C. M. (2006), Pattern Recognition and Machine Learning, Springer, ISBN 978-0-387-31073-2.

[6] "Google Code Archive - Long-term storage for Google Code Project Hosting". code.google.com. Retrieved 13 June 2020.