Predicting Popularity of Online News Article

Tushar Kumar
University of Rochester
Rochester, New York
tusharku@UR.Rochester.edu

ABSTRACT

This paper provides a detailed analysis of applying different classification approaches on the Online News Popularity Data Set present in UCI ML respository, with the goal of learning to predicting the popularity of online news article well. We use logistic regression, Decision trees and Neural Network as our classification approaches and analyze how they perform on the dataset. We also find out the relevance of attributes and attempt to build classifiers which use only a small subset of attributes of the dataset and yet perform better. This project was done as part of the graduate course, CSC440 taken at University of Rochester during Fall 2018.

KEYWORDS

Classification, Neural Network, Logistic Regression, Decision Trees, Online News Popularity Dataset

ACM Reference Format:
Tushar Kumar. 2018. Predicting Popularity of Online News Article. Rochester, NY, USA, 9 pages.

1 INTRODUCTION

Two-thirds (67%) of Americans report that they get at least some of their news on social media with two-in-ten doing so often. Social media and online news have become a very important medium of information and the nature of social media has the capability to make any news article reach through any corner of the world. With such relevance of online news article, the task of predicting the popularity of a news article using the different features of the article becomes quite pertinent to online mediums like online publishing houses. They rely on reaching the masses through this medium and having the knowledge as to what makes a certain article popular would be very beneficial for them.

An online article can have huge number of attributes and while using each and every of them might make sense, one must realize that the end goal of the system is not just to reach a certain accuracy number but it is ensuring that people using these systems understand the relevant attributes and attempt to improve the quality of article by removing attributes that negatively affect the articles popularity and augmenting the article with knowledge of attributes which are known to have a positive impact of its popularity. Another use of such systems can be to find which amongst the two articles is going to be more popular and using that as a decision criteria when multiple articles focusing on the same topic are available at hand.

With that objective kept in mind, I apply different classification techniques to Online news popularity dataset[1]. I implement the following techniques and apply them on the dataset and analyze the results:

(1) *Implement χ^2 and Fisher score method for understanding relevance of attributes to article popularity.*
(2) *Implement Logistic Regression and apply it on the dataset and also apply it on dataset only with relevant attributes*
(3) *Implement Decision trees and apply it on the dataset*
(4) *Implement Random Decision Forest and apply it on the dataset*
(5) *Implement Neural Network and apply it on the dataset.*
(6) *Convert URL to word embeddings using Glove* and add them as features for Neural Network to see if content extracted from url adds benefit to predicting popularity.
(7) *Implement Extra Trees Classifier and apply it on the dataset*

2 ONLINE NEWS POPULARITY DATASET

Online News Popularity Dataset summarizes a heterogeneous set of features about articles published by Mashable in a period of two years. The goal is to predict the number of shares in social networks (popularity). These are not necessarily attributes which actually impact the popularity but are ones which could possible impact it and hence one of the goals of our approach is to actually find attributes that are relevant. We first briefly define the properties of dataset in general before talking about each attributes:

(1) Dataset consists of 39797 instances.
(2) All attributes are continuous except the url attribute of article which is a string.
(3) To facilitate learning, we will divide the dataset into training, validation and test split of 60%, 20% and 20% split (train=23,878, validation=7,959 and test=7,959)
(4) There are no missing values in the dataset.
(5) Prediction task can either be treated as a regression where one would predict actual number of shares or classification task where one would predict the appropriate bin of shares.

2.1 Attributes and their Properties

The Dataset consists of 61 attributes(including the shares attribute) of which all except url attribute is continuous. The attributes can be divided into the following categories

(1) url - Provides the url of the article
(2) Date, Day and Time - These are attributes related to publishing date, day or time of article
(3) Tokens - Attributes related to tokens or words used in the article
(4) Multimedia - Attributes related to media content like images, videos and links.
(5) Channel - Attributes related to channel of article.

[1] https://archive.ics.uci.edu/ml/datasets/online+news+popularity

CSC440 Fall'18, , Rochester, NY, USA

Table 1: Categorization of Attributes of an Article

Category Name	List of Attribute
Url	url of article
Date,Day and Time	timedelta, weekday_is_*, is_weekend
Tokens	n_tokens_title, n_tokens_content, n_unique_tokens, n_non_stop_words, n_non_stop_unique_tokens, average_token_length,
Multimedia	num_hrefs, num_self_hrefs, num_imgs, num_videos
Channel	data_channel_is*
Topic	LDA_00, LDA_01, LDA_02, LDA_03, LDA_04
Sentiment/Subjectivity	global_subjectivity, global_sentiment_polarity, global_rate_positive_words, global_rate_negative_words, rate_positive_words, rate_negative_words, avg/min/max_positive_polarity, avg/min/max_negative_polarity, title_subjectivity,title_sentiment_polarity, abs_title_subjectivity, abs_title_sentiment_polarity
Reference shares	self_reference_min/max/avg_shares
Shares	shares

(6) keywords - Provide different details regarding keywords used in article.
(7) Topic - Provide closeness of article to specific topics
(8) Sentiment/Subjectivity - Polarity or subjectivity/sentiment of text in article
(9) Reference shares - Details on shares of articles referenced in the article.
(10) Shares - Target attribute which provides details on number of shares this article had.

Table 1 provides details on different attributes we have in the dataset relevant to each category.

2.1.1 ***Filtering attributes:*** The first task that I do in pre-processing this data is remove the url attribute. The reason being that url is present in the dataset more for a data completion of the dataset, it does not really impact the popularity of article. We will indeed look into this once we are building complex classifiers to identify if text in url indeed has an impact on popularity but for our baseline model we would ignore the url attribute and remove it from our dataset.

2.1.2 ***Handling Missing attributes:*** Since this dataset does not have any missing attirubte our work is simplified in this aspect as we do not really need to do any handling here.

2.1.3 ***Scaling of attributes:*** Each attribute here has continuous values and one can see from Table 2, which has information on a sample of attributes, that the value ranges vary quite a lot. Hence in order to have each feature weight within the same range for our models I scale the the values of each feature before applying any of our models on them. There are three different types of scaling we try:
1. Normalizing each feature to have zero mean and unit variance

$$Transformation(x) = \frac{x - \mu}{\sigma}$$

2. Scale each feature so that they are zero centred.

$$Transformation(x) = \frac{x - \mu}{max(x) - min(x)}$$

3. Scale each feature so that fall within the range of 0 to 1

$$Transformation(x) = \frac{x - min(x)}{max(x) - min(x)}$$

Based on our validation tests, the best feature scaling turned out to scaling them to ensure they are zero centred.

2.1.4 ***Discretization of shares:*** In order to have a classification task at hand, we need to deal with the continuous nature of target attribute "shares". We would like to divide it into different sections where each section has a different class assigned to it. We look at the distribution of the shares in the dataset to get some insight into how are the shares values distributed. Figure 1 provides the plot of the distribution. I trim the top and bottom 10% to reduce the skewness in plot for visualization purposes. As one can see that the distribution is heavily skewed towards right and hence picking the mean as a mechanism to discretize would not be correct. We therefore use median as a mechanism to find the right categorizations. We use the median of the distribution to divide the shares into two classes : popular, unpopular. The median of shares is 1400. We then simply modify the target attribute value. Every tuple where the number of shares is less than or equal to 1400 belongs to the unpopular class and every article with more than 1400 belongs to popular class.

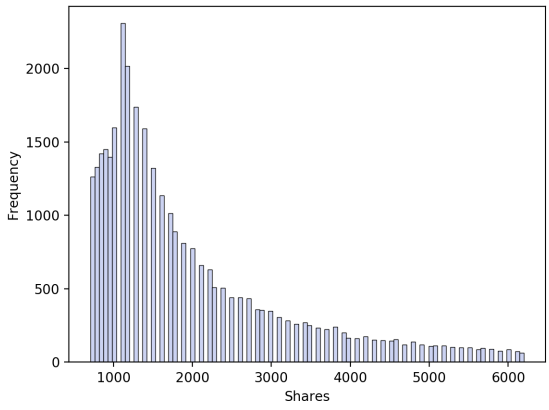

Figure 1: Histogram of shares(10% trimmed) in dataset

2.2 Relevance of Attributes

The dataset has 60 attributes excluding the url of the article and these are just aggregation of all features of this article. It might be very tempting to straight away start applying classification methods

on these attributes but we must understand that not all attributes will impact the popularity of article. Infact given the small size of our dataset if we wish to build classifiers having high accuracy we must not implode it with every attribute that we have. I therefore attemp to first find out which of these attributes are statiscally relevant to the target attribute. The reason this is necessary if because complex algorithms are very good at adapting to noise in the dataset and if the attributes are not really impact popularity, our learning algorithm would simply fit to the noise that these extra attributes are adding to the dataset and negatively affect our accuracy. We use χ^2 and Fisher Score to evaluate the relevance, about which we mention in brief below.

2.2.1 χ^2 ***statistic***. In statistics, the χ^2 test [3] is applied to test the independence of two events, where two events A and B are defined to be independent if $P(AB) = P(A)P(B)$ or, equivalently, $P(A|B) = P(A)$ and $P(B|A) = P(B)$. In feature selection, the two events are occurrence of the term and occurrence of the class. We then rank terms with respect to the following quantity:

$$\chi^2(Data, target, feature) = \Sigma_t \Sigma_f \frac{(N_{tf} - E_{tf})^2}{E_{tf}}$$

where f is the set of values for feature, t is the set of values for target attribute, N_{tf} is observed frequency of feature value f and target value t and E_{tf} is expected frequency of feature value f and target value t, assuming they are independent. I use skfeature package [6] to use this functionality and analyze the relevance for different attributes. Figure 2 shows the relevance score for each attribute using χ^2 statistic method. The top 5 features with respect to χ^2 statistic ranked top to bottom are self_reference_max_shares, self_reference_avg_shares, self_reference_min_shares, kw_avg_max, kw_max_max. One would notice that we have not really ranked all attributes and the reason for that is actually a limitation of chi square computation. Chi square computation requires the observations to be non-negative and it produces unstable result when the attributes have negative values. Hence we turned to fisher score for getting a ranking of all attributes.

2.2.2 ***Fisher Score***. The key idea of Fisher score [1] is to find a subset of features, such that in the data space spanned by the selected features, the distances between data points in different classes are as large as possible, while the distances between data points in the same class are as small as possible. We compute let μ_{jk} and σ_{jk} which are the mean and standard deviation of k-th class, corresponding to the j-th feature. If μ_j and σ_J denote the mean and standard deviation of the whole data set corresponding to the j-th feature. Then the Fisher score of the j-th feature is computed below

$$F(x_j) = \frac{\Sigma_{k=1}^{c} n_k(\mu_{jk} - \mu_j)^2}{(\sigma_j)^2}$$

To put simply, it tries to score each feature based on how well that feature discriminates the target classes from each other. After computing the Fisher score for each feature, it selects the top-m ranked features with large scores. I use skit-feature module to analyze the relevance for different attributes using Fisher score. Figure 3 shows the relevance score for each attribute using Fisher score method. The top 5 features with respect to Fisher score ranked top to bottom are kw_avg_avg, LDA_02, data_channel_is_world, is_weekend, data_channel_is_entertainment. Because Fisher score allows us to find relevance for all attributes in the dataset and because it makes much more sense to use that instead of χ^2 given the non-categorical nature of attributes in the dataset we use the ranking of provided by Fisher score as the relevance indicator for attributes in dataset. Table 2 provides the statistics about the top 10 attributes and Figure 4 provides the distribution of values of each of the top 10 features.

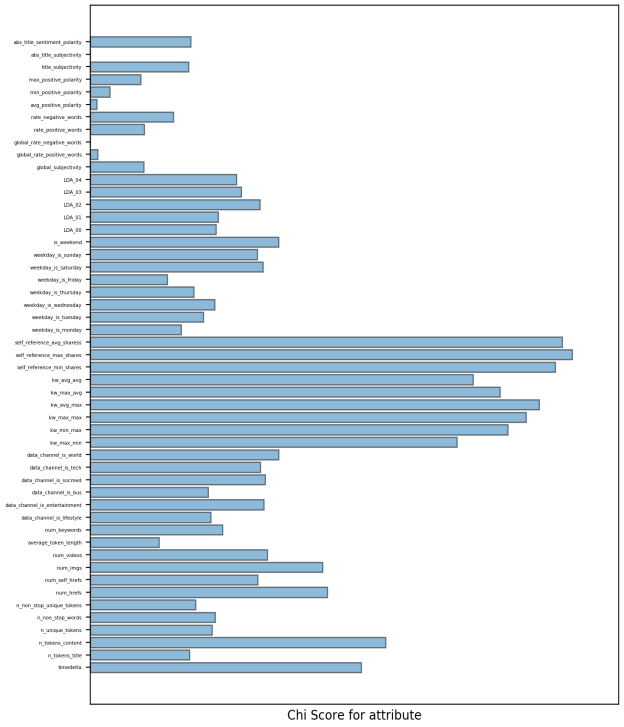

Figure 2: Plot of Chi Square(log scale) values between attribute values and target values

3 CLASSIFICATION MODELS

This section will dive deeper into different models explaining their functioning. I will first explain my baseline model and then compare the other models with respect to the baseline model.

3.1 Baseline Model - Decision Trees

My baseline model is a decision tree learning algorithm. The procedure of learning decision tree basically involves iterating through all the attributes one by one and finding the attribute that provides the maximum reduction in entropy value. A dataset which has all rows with same class value will have the minimum entropy value whereas a dataset which has equal number rows for each target

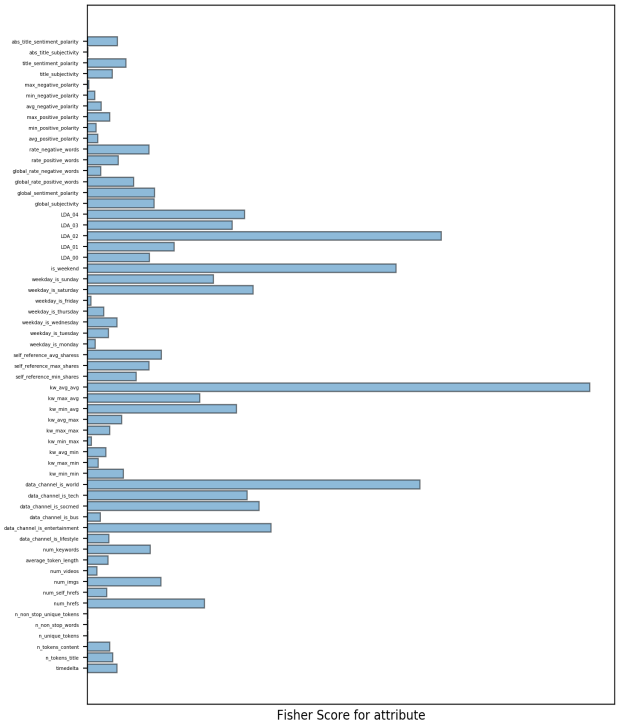

Figure 3: Plot of Fisher score value between attribute values and target values

Table 2: Mean and Variance of top 10(relevant) Continuous Attributes

Attribute Name	Mean	Variance
kw_avg_avg	3135.8586	1737476.6412
LDA_02	0.2163	0.07960
data_channel_is_world	0.2125	0.1673
is_weekend	0.13091	0.11377
data_channel_is_entertainment	0.1780	0.14632
data_channel_is_socmed	0.05859	0.05516
weekday_is_saturday	0.06187	0.05804
data_channel_is_tech	0.185299	0.1509
LDA_04	0.23402	0.08362
kw_min_avg	1117.146	1293775.679

attribute domain value will have maximum entropy. Once we find the best attribute, we partition the dataset into based on the different values of that attribute and then recursively do the same for each partition. The base case happens when there are either no rows or no attributes to test or all the rows in the partition have the same target attribute value. In case there are no rows, we use

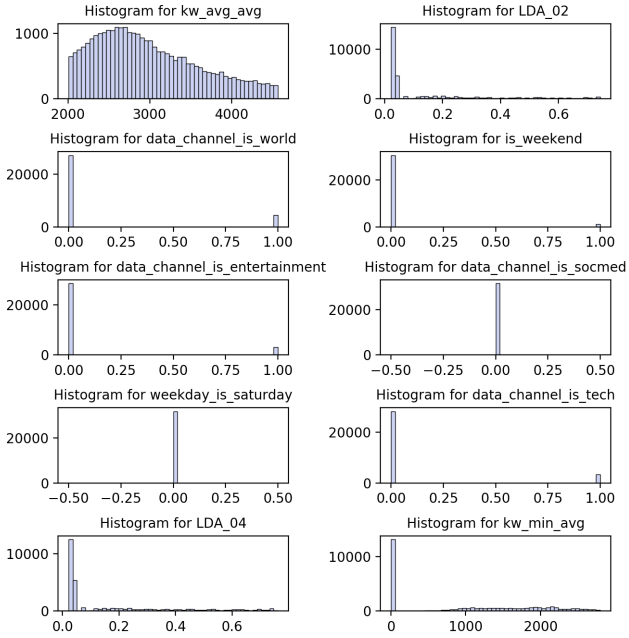

Figure 4: Histogram of top 10(relevance) continuous attributes in dataset

the majority class value from the parent node to predict the label for this node. For splitting criteria I use instead of entropy , Gini impurity which is given by

$$Gini\ Impurity = 1 - \Sigma_{c=1}^{C} p_c^2$$

where C are the different class labels for data and p_c is the probability for that class to occur. I use the implementation of Decision trees in sklearn package [8]. To ensure that decision trees do not overfit one can actually stop the tree from growing beyond a particular depth. To find out what should be the right depth for decision trees, I use validation. I will now talk about the validation setting for finding the right optimized value of all the hyperparameters. The same validation setup will be used in all the different models.

3.1.1 *Validation Setup.*

- Divide the entire data into train, test, validate split of 60/20/20
- Train the algorithm tree on the 60% and validate the hyper parameters on the validate set.
- Train the learning algorithm using the validated hyper parameters on the 80% (keeping test data untouched)
- Test and report the accuracy of algorithm on the 20% of data.

Figure 5 provides the details of my validation experiment. One can see that because of the decision tree being such a simple classifier it overfits the data and hence does not generalize well on unseen data that is data from the validation set. It would be hence best to restrict the depth of the tree to a small number so that we don't see

a huge different between training and test accuracy. Hence I use the depth as 5 for my decision tree. Because we are restricting the depth of the tree we don't need to really worry about feature selection because the depth would ensure that only top 5 features which lead to maximum split gain will be used. I also tried with other splitting criteria like entropy but gini impurity was itself resulting in the best validation accuracy. Decision tree algorithm gives us the accuracy of **64.11**. Because of the space the tree occupies , it was not feasible to provide the tree for depth 5, however I provide the tree one would get if they provide a max depth of 3 in Figure . As one can see, the attributes on which splits are made are always the ones which have high χ^2 or Fisher score value.

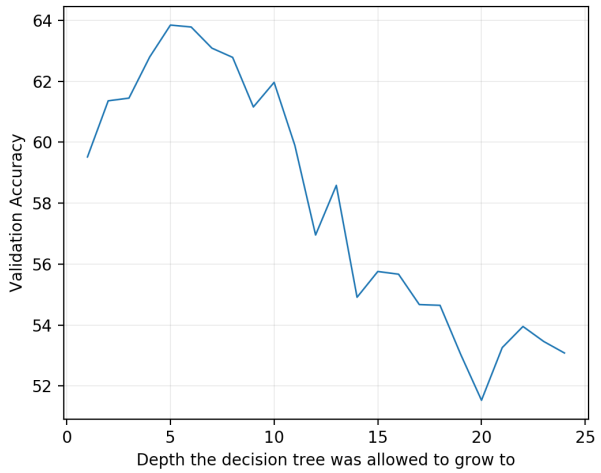

Figure 5: Plot of Accuracy vs Depth the tree was allowed to grow

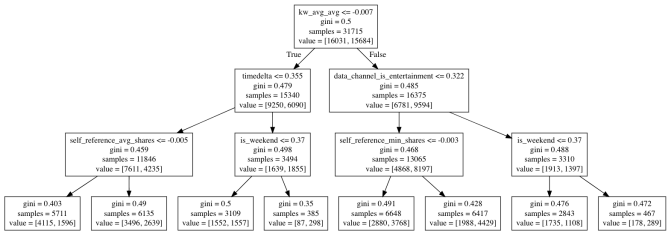

Figure 6: Decision Tree output for Max Depth 3

3.2 Logistic Regression Model

The second model that I attempt to learn and use for predicting the class labels of news articles is the Logistic Regression model. I first do not do any filtering on the attributes because I want to measure the benefit of keeping only highly relevant attributes on the very same model. Logistic Regression is a special type of regression where response variable is related to a set of explanatory variables, which can be discrete and/or continuous. In logistic regression Probability or Odds of the response taking a particular value is modeled based on combination of values taken by the predictors. It is named for the function used at the core of the method, the logistic function given below:

$$logit(z) = \frac{1}{1 + \exp^{(-z)}}$$

Since the output of logistic function is always between zero and 1, it makes sense to treat it as a probability and use it for performing classification where the value of the logistic function simply indicates the confidence that classifier has in this data point belonging to a particular class. ItâĂŹs also closely related to âĂIJexponential familyâĂİ distributions, where the probability of some vector v is proportional to $\exp^{\beta_0} + \Sigma_{j=1}^{m} f_j(v)\beta_j$. If one of the components of v is binary, and the functions f_j are all the identity function, then we get a logistic regression. Exponential families arise in many contexts in statistical theory (and in physics!), so there are lots of problems which can be turned into logistic regression.

3.2.1 ***Cross Entropy Loss.*** The loss function that we use for multi classification problem is Cross Entropy Loss. Cross-entropy loss, or log loss, measures the performance of a classification model whose output is a probability value between 0 and 1. Cross Entropy has the following form for loss when dealing with binary classification problems:

$$loss = -(y * log(p) + (1 - y)(log(1 - p)))$$

When classes are more than two, it takes the general form below:

$$loss = \Sigma_{c=1}^{M} \delta_{o,c} log(p_{oc})$$

where δ is indicator function which is 1 when the class of observation is c, and p_{oc} is the probability that our models calculates for the observation o to be of class c.

3.2.2 ***Softmax Function.*** In order to get probabilities, I use the softmax function. This way, I can just calculate some linear score based on learned weights of attributes of article and softmax will convert the M linear scores(M being the the number of classes) to M probabilities. Softmax function is defined as below:

$$Softmax(x_i) = \frac{\exp^{x_i}}{\Sigma_i \exp^{x_i}}$$

3.2.3 ***Implementation.*** I implement my model on Pytorch 0.4 [7] using the nn module. I use the dataset in its entirety, meaning there are no attributes removed other than url. I use Adam [4] as the optimization method instead of Stochiastic gradient descent because it leads to convergence much faster. I use a learning rate of 1E-3 with reducing it by one tenth every time my reduction in loss plateaus. I also use a weight decay of 1E=4. When we run the logistic regression model on the entire dataset for 200 epochs we get an accuracy of ≈ 65%.

3.2.4 ***Feature Selection.*** While using fisher score, we have ranked features, we still need to decide what features to actually choose from them. We can obviously decide to choose topK features based where is K pretty much decided based on heursitic. However, a good mechanism to do that would be to see what are the accuracy we are receiving on validation set. During valition testing, I found

Table 3: Comparison between logistic regression on entire dataset vs relevant attributes

Model	Epochs	Training Loss	Accuracy
All Attributes	150	0.517	65.11
Top 36 Attributes	150	0.519	**65.34**

that K=36 gives us the best set validation accuracy for logistic regression model when tested with values of ranging from 1 to 60. One thing to point out is that I did try selecting only the features which caused an increase in accuracy In order to understand the impact of this on the convergence of loss and accuracy , I apply the baseline model of logistic regression now on this filtered dataset which only has top 36 attributes as features. Table 3 provides the data on the run times and loss and accuracy values illustrating the benefit of feature selection. A figurative comparison between the two scenarios is provided through accuracy and loss plots on validation data in Figure 7. As is clear through the plots, not only does feature selection lead to a better accuracy but it leads to a much better convergence rate, which makes sense because we have less parameters to train and hence less chances of overfitting on a relatively small dataset.

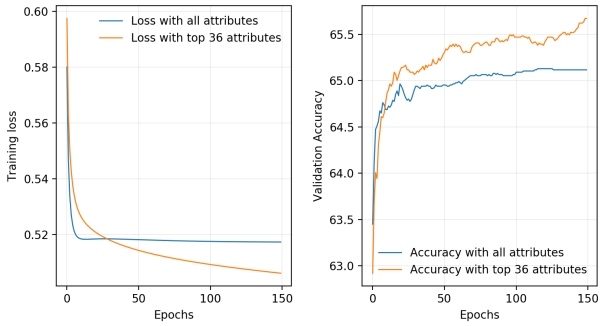

Figure 7: Benefit for feature selection on Loss/Accuracy

3.2.5 ***Analyzing weights of Logistic.*** Part of the reason we wanted to be able to predict the popularity of an article was that we wanted to also understand what actually causes an article to be popular. Well, given that our features are all scaled equally , we can analyze the weights of logistic model that it learned to understand which features are getting high positive weights and which features are getting high negative weights. Figure 8 provides the plot of each of the top 36 features weights that logistic model learned. We can get the following conclusion from this plot: Top 5 Features that have a huge positive impact on popularity are:

(1) kw_avg_avg - Avg. keyword (avg. shares)
(2) self_reference_min_shares - Min. shares of referenced articles in Mashable
(3) self_reference_avg_shares - Avg. shares of referenced articles in Mashable
(4) n_tokens_content - Number of words in the content
(5) num_hrefs - Number of links

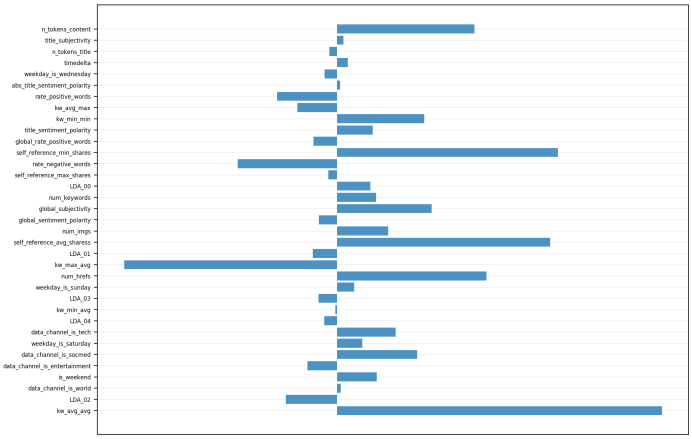

Figure 8: Plot of Importance of features for Popularity of article using logistic weights

Top 2 Features that have a negative impact on popularity are:

(1) kw_max_avg - Avg. keyword (max. shares)
(2) rate_negative_words - Rate of negative words among non-neutral tokens

Hence we see that referencing articles which are known to have higher shares increases the chances of the article becoming popular which makes perfect sense intuitively. On the other hand having huge number of negative words decreases the chances of popularity.

3.3 Neural Networks

Neural networks are a set of algorithms, modeled loosely after the functioning of human brain through neurons, that are designed to recognize patterns. Theoretically it has been proven that for any function there exists a neural net with certain architecture than can approximate that function to a very small amount of error. There are a lot of different elements of a neural network learning algorithm but the most important ones can be listed below :

(1) Number of layers
(2) Number of neurons in each layer
(3) Activation function for layers - This is the element which introduces non-linearity in the architecture (in the absence of this , neural network would just be another linear model). Popular ones are sigmoid, tanh, ReLU [5], LeakyReLU.
(4) Loss function - This function provides a measure to the neural net as to how good/bad its doing on the task as of now.
(5) learning rate - Rate at which neural net is supposed to alter the weights by a change in the direction opposite to the gradient of loss(in effort to minimize the loss)
(6) Regularization - Also called weight decay. In one version of regularization a function of weight is added to the cost in order to ensure that the neural network does not learn weights

that are very huge. Other techniques are also present like dropout [9] which randomly switch off neurons to ensure that networks do not overfit and also ensure that all neurons learn something from the data and the entire neural network does not just end up relying on few collection of neurons(and switch off the other ones) to learn the task,

A problem with neural networks is that with small dataset its very tempting to go after having a complicated layer to ensure the accuracy bumps to 100%. But what actually ends up happening is our model does not generalize at all because of it completely overfitting to the training data. Figure 9 clearly shows this impact with training a model with all attributes of dataset , we see that even though the training accuracy keeps on increasing, the validation accuracy continues to decrease with epochs. The reason is simple that our model is overtting to data. There are two ways with which we can prevent this from happening. We can use weight decay which ensures that the model does not learn weights that are huge in size and we can add dropout which will randomly switch off few neurons disabling the network from overfitting. Figure 10 shows the benefit of adding a weight decay of 1E-3 and dropout of 0.2 to the same architecture which was earlier overfitting. Though we have lesser training accuracy but we have much better generalization. Both the networks were trained on the entire dataset as in all attributes with the same learning rate and same number of epochs. The only difference was one had regularization and one did not.

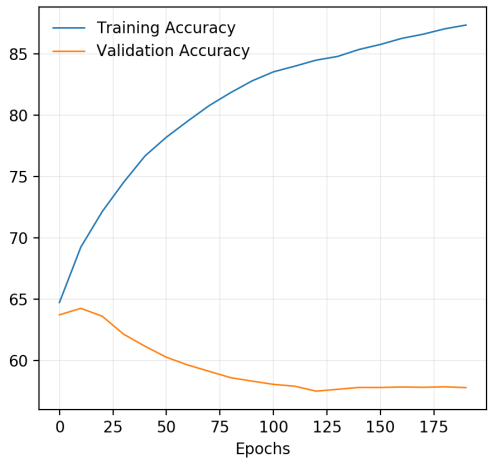

Figure 9: Effect of overfitting by training on Online News Dataset with no regularization

In order to train the neural net I try with few different activation function, but the best activation which gives the maximum stability while training and higher validation accuracy is LeakyRELU. Leaky ReLU unlike RELU has a small slope for negative values, instead of altogether zero. For example, leaky ReLU may have y = 0.01x when x < 0 rather then y = 0 when x < 0 (in case of RELU). Since it does not have zero slope parts it fixes the dying neurons problem that's been observed with RELU because of slope being zero for certain parts of functions causing gradient to be zero. I also validate with a

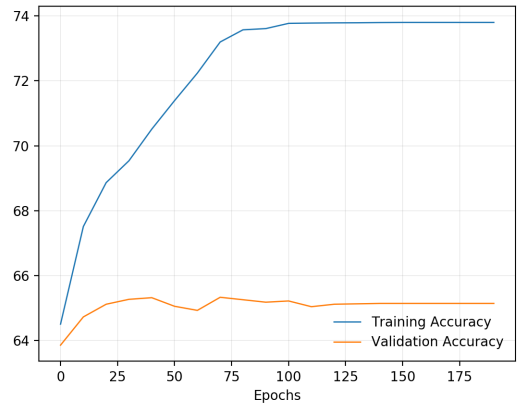

Figure 10: Preventing overfitting by weight decay and dropout on Online News Dataset

lot of different architectures in order to figure out the one to use for training of my online news data set. I try with the following architectures:

(1) 1 layer with 10 hidden neuron
(2) 1 layer with 50 hidden neurons
(3) 1 layer with 100 hidden neurons
(4) 1 layer with 500 hidden neurons
(5) 2 layer with 10 hidden neurons each
(6) 2 layer with 50 hidden neurons each
(7) 2 layer with 100 hidden neurons each
(8) 2 layer with 500 hidden neurons each
(9) 3 layer with 10 hidden neurons each
(10) 3 layer with 50 hidden neurons each
(11) 3 layer with 100 hidden neurons each
(12) 3 layer with 100 hidden neurons each

Figure 11 shows the validation accuracy of all these architectures on the validation set after training them with a learning rate of 1E-3 , weight decay of 1E-4 , dropout of 0.2 and batch size of 100. We see that the best architecture is having 1 layer with 100 neurons each which also makes sense sense we do not have enough data to train a really complicated network. On training our neural net with the following specifications we get an accuracy of **66.65**

(1) 1 Layer NN with 100 neurons
(2) Learning rate of 1E-3
(3) Weight decay of 1E-4 and dropout of 0.2
(4) LeakyRELU as activation function
(5) Batch size of 100
(6) Training testing split of 0.2

3.4 Using URL with Glove word embeddings

Since there are urls of each article present with us in the database, one idea that I had was why not use the url attribute and see if that can improve the accuracy. The reason why this could benefit the learning process, was that the url for article in dataset tells us very strongly about the content of article. For example the first rwow in the dataset has the url as

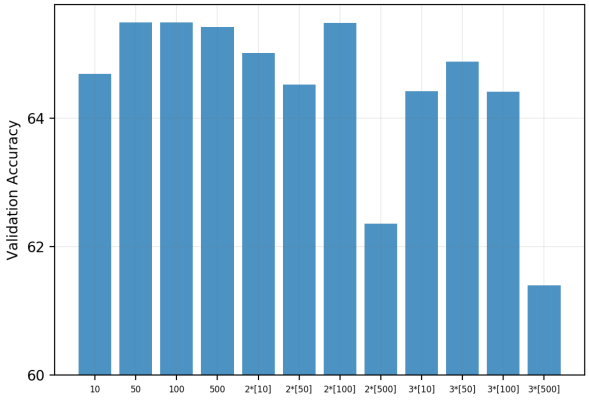

Figure 11: Comparison of different architectures for neural net

Table 4: Impact of using glove embeddings for url as features

Setting	Accuracy
Same settings as that 1NN	61.11
Increased dropout(0.35) and weight decay(1E-3)	**64.00**

http://mashable.com/2013/01/07/amazon-instant-video-browser

. One can easily see that this must be an article about the amazon instant video. While it might be debatable that I am assuming the content of the article has a relationship with the shares but thats exactly my hypothesis which we will try to find out. I use the Glove vectors [2] specifically the 25 dimensions word embeddings obtained from tweets . The reason I go with 25d because I would not benefit from having a high dimensional vector for url because we dont have enough data to learn any relationship. The way I obtain the word vector for a url is I strip off and keep only the part after the last '/'. So in our above url, this would give me amazon-instant-video-browser. Now I try to find the mebeddings of the 4 words (amazon/instant/video/browser) , if the word embedding is present I use that, else I mark it as zero vector. The reason I do that is because at the end I sum all the word vectors to in a way form the embedding of the url. And hence adding a zero vector won't make any difference. Moreover this scheme ensures that if there are 3 words which are exactly same in two urls but the fourth word is different yet very similar to each other, we end up having two url vectors which are very close to each other. Once I have the url vectors of all the urls I just append it to my existing data.

I then train the neural network with the same specifications.

Table 4 provides the results of my execution. Initially I find that the accuracy is not at all good meaning that the url provides no information but I find that the generalization keeps on getting worse meaning that there could be overfitting. This actually ended up being the case because when I increase dropout to 0.35 and weight decay to 1E-3 I receive a much better result. However overall the accuracy is still less than that when not using the url vectors. This could be becasue of less data and we might need more data to conclusively tell whether url and hence contents of the article has any impact on its popularity.

[2]https://nlp.stanford.edu/projects/glove/

Table 5: Results of using ensemble techniques

Ensemble Method	Depth	Estimators	Accuracy
Random Forest	26	550	66.84
Extra-Trees Classifier	22	500	**67.24**

3.5 Ensemble of Trees

In this section we analyze the benefit of using ensemble of trees as a classifier. Ensemble methods, which combines several decision trees to produce better predictive performance than utilizing a single decision tree. The main principle behind the ensemble model is that a group of weak learners come together to form a strong learner.

Bagging (Bootstrap Aggregation) is used when our goal is to reduce the variance of a decision tree. Here idea is to create several subsets of data from training sample chosen randomly with replacement. Now, each collection of subset data is used to train their decision trees. As a result, we end up with an ensemble of different models. Average of all the predictions from different trees are used which is more robust than a single decision tree.

Random Forest is an extension over bagging. It takes one extra step where in addition to taking the random subset of data, it also takes the random selection of features rather than using all features to grow trees.

Another ensemble tree classifier called **Extra Tree classifier** takes randomness one step further: the splitting thresholds are randomized. Instead of looking for the most discriminative threshold, thresholds are drawn at random for each candidate feature and the best of these randomly-generated thresholds is picked as the splitting rule. This usually allows reduction of the variance of the model a bit more, at the expense of a slightly greater increase in bias. These are the two ensemble classifiers I use. I use validation set to find out two important attributes for both ensemble methods:

(1) Number of trees to learn : This is the number of estimators attribute in the sklearn package. This basically means how many different weak learners do we wish to learn
(2) Max Depth : This has the same meaning as in the case of decision trees. It implies what should be depth to which a tree should be allowed to grow to.

Table 6 provides the different the details of results I was able to obtain using the ensemble techniques. This ensemble technique ends up resulting in the maximum accuracy received. I also parallellize the trees learned using 10 threads so the overall learning process is quite fast In both the techniques I only use the top 36 attributes as in Logistic regression. I also tried with Boosting methods like GBM,s but they were giving lesser accuracies than Extra Tree classifier. Using the ensemble techique I actually am able to get better accuracy results than obtained by [2] who got 67% accuracy

Table 6: Results of all classifier

No.	Method	Attributes Used	Accuracy
1	**Decision Trees**	All	64.11
2	**Logistic Regression**	All	65.11
3	Logistic Regression	**Top 36**	65.34
4	**Neural Networks**	All	66.65
5	Neural Networks(**With glove embeddings for Url**)	All	64.00
6	**Random Forest**	Top 36	66.84
7	**Extra Trees Classifier**	Top 36	**67.24**

4 CONCLUSIONS

In this project, I implemented 5 learning algorithms and analyzed their accuracy on the test set after using validation set for optimizing all the hypoer parameters of all models. Table provides the accuracy achieved by all the classiers. Extremely Randomized classier called Extra-Trees classifer gives the best result in terms of accuracy. I also analyzed the benefit of feature selection on the logistic regression model and importance of regularizations when learning complex models on relatively smaller dataset. While the usage of glove embeddings for url did not really giv us huge benefits, the fact that it has comparable accuracy to other classifiers means that urls are indeed adding relevant information and given enough data and also given the specific LDA topics we could build a dataset of what exactly is the content of that article and then see if the content solely is a good enough driving factor of popularity and can act as a featore for prediction tasks. I also analyzed the importance of each feature in making an article popular using the weights that were learned by logisitic regression model. I came to the conclusion that the features like referencing articles which are known to have higher shares increases the chances of the article becoming popular which makes perfect sense intuitively. On the other hand having huge number of negative words decreases the chances of popularity. The more number of links we have in an article, the better our chances of the article becoming popular. Figure 8 and Section 3.2.5 talk more about this in detail.

ACKNOWLEDGMENTS

I would like to thank Professor Ted Pawlicki for giving me this opportunity to implement these algorithms and analyze them which has fostered a much deeper understanding of these methods. This report was created as part of the project requirement given to us for a graduate course in Data mining at University of Rochester taught by Professor Ted Pawlicki

REFERENCES

[1] Richard O. Duda, Peter E. Hart, and David G. Stork. 2000. *Pattern Classification (2Nd Edition)*. Wiley-Interscience, New York, NY, USA.

[2] Kelwin Fernandes, Pedro Vinagre, and Paulo Cortez. 2015. A Proactive Intelligent Decision Support System for Predicting the Popularity of Online News. In *Progress in Artificial Intelligence*, Francisco Pereira, Penousal Machado, Ernesto Costa, and Amílcar Cardoso (Eds.). Springer International Publishing, Cham, 535–546.

[3] Jian Pei Jiawei Han, Micheline Kamber. 1993. *Data Mining: Concepts and Techniques (3rd Ed.)*. Morgan Kaufmann; 3 edition (July 6, 2011).

[4] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980 (2014). arXiv:1412.6980 http://arxiv.org/abs/1412.6980

[5] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2017. ImageNet Classification with Deep Convolutional Neural Networks. *Commun. ACM* 60, 6 (May 2017), 84–90. https://doi.org/10.1145/3065386

[6] Jundong Li, Kewei Cheng, Suhang Wang, Fred Morstatter, Robert P Trevino, Jiliang Tang, and Huan Liu. 2016. Feature Selection: A Data Perspective. *arXiv preprint arXiv:1601.07996* (2016).

[7] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. (2017).

[8] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[9] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research* 15 (2014), 1929–1958. http://jmlr.org/papers/v15/srivastava14a.html