Understanding the Interplay between Content, Users and Popularity using Reddit Submissions

Feihua Fang, Jiaqi He, Jiazhuo Qin, Yifan Qin

Abstract

Reddit is a quite popular discussion website in the United States. Hundreds of thousands of people share interesting images with a title on reddit and receive various number of votes and comments every day. Which kind of the posts will be more likely to receive high upvotes and large number of the comments? For a certain image, what is the most appropriate time to post it in order to become popular. Is there so-called celebrity affection in reddit like that in Twitter? How could the title influence the popularity of the posts? By looking into the dataset of reddit submissions, we can find the answers about these questions and confirm whether our assumption is correct or wrong. In this paper, we use k-means, linear regression, TF-IDF models, LDA model to analyze these the pros and cons about the given features when predict the popularity of each posts.

Keywords: regression, TF-IDF, clustering, LDA, MLPRegression

1 DATASET

In this case, we use the dataset of reddit submissions[1] provided by Stanford website. This dataset provides us with record about the posts on the reddit. Reddit is an American social news aggregation, web content rating, and discussion website. Registered members submit content to the site such as links, text posts, and images, which are then voted up or down by other members. Posts are organized by subject into boards called "subreddits", which cover a variety of topics including news, science, movies, video games, music, books, fitness, food, and image-sharing. Submissions with more up-votes appear towards the top of their subreddit and, if they receive enough votes, ultimately on the site's front page[2]. We conduct deep research about the characteristic about this website and do some prediction in this report.

1.1 Review Data Formula

We have 132309 rows in the dataset. That is, there are 132309 records of submissions in total. Each record has several features, including the time of the submission, id of the image, title and user information along with the number of votes and comments this submission has. The detail information about the each row of record is given in Table 1.

Table 1: Data formula

name	description
image_id	id of the image, submissions with the same id are of the same image
unixtime	time of the submission (unix time)
rawtime	raw text of the time
title	submission title
total_votes	number of upvotes + number of downvotes
reddit_id	id of the submission on reddit
number_of_upvotes	number of upvotes
subreddit	subreddit
number_of_downvotes	number of downvotes
localtime	local time of the submission (unix time)
score	number of upvotes - number of downvotes
number_of_comments	number of comments the submission received
username	name of the user who submitted the image

Among all these features, the most valuable one should be the image because people will votes and comments mainly on what the image expresses. However, the detail information about the image is not included. Apart from the image feature, the title is another important one because a fascinating title will attract more attention from people so the posts with interesting titles will be more likely to be visited.

1.2 Dataset analysis

In this section, we will conduct research on the dataset itself by raising some problems.

1.2.1 Celebrity affection

In Twitter, it is easy to find that we have such celebrity affection, that is, the popular people will be more likely to get more votes and comments in a post than an ordinary person. We use Pearson correlation coefficients to see if such affection also occurs in Reddit.

We use the number of the posts to bipartite celebrity from non-celebrity. That is, if a single user has larger posts than others, he/she is more likely to be a celebrity. We will do experiment to find if these people with large number of the posts will have more votes/comments in the future posts. We calculate the correlation coefficient matrix of posts verse votes as matrix1 and posts verse comments as matrix2.

$$\begin{bmatrix} 1.0 & -0.00137 \\ -0.00137 & 1.0 \end{bmatrix} \quad (1)$$

$$\begin{bmatrix} 1.0 & -0.001169 \\ -0.001169 & 1.0 \end{bmatrix} \quad (2)$$

For the statistics we find that, the correlation coefficient between the number of the post and votes/comments are -0.00137 and -0.001169, which are near to zero. We plot the relation between posts and votes as Figure 1 and comments as Figure 2. The statistics and figure shows that there is hardly any relation between the number of the posts a user had and votes and the comments the user obtained. Thus, there is no celebrity affection in Reddit. So we have the conclusion that Reddit users focus more on the content itself of the post rather than the user who posted.

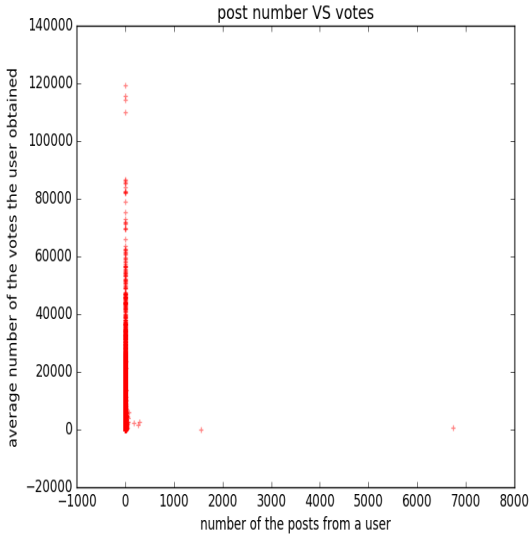

Figure 1: The relation between number of posts and obtained votes.

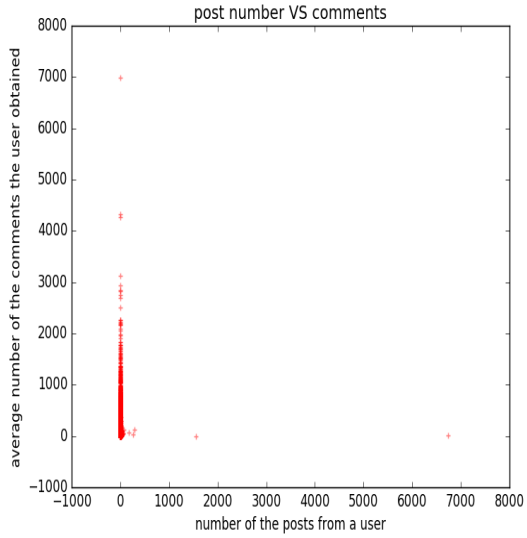

Figure 2: The relation between number of posts and obtained comments.

1.2.2 Appropriate time for submission

Intuitively, a right time to submit the picture can help attract more attentions which are represented by more comments and votes. In this section, we are trying to explore the time effect by studying the overall dataset.
First, Let's consider that the popularity change rule with time when users submit the same image at different times. we chose the 5 images that have the largest numbers of repetitions and then plotted the relation between time and comments as well as upvotes as Figure 3 shows.

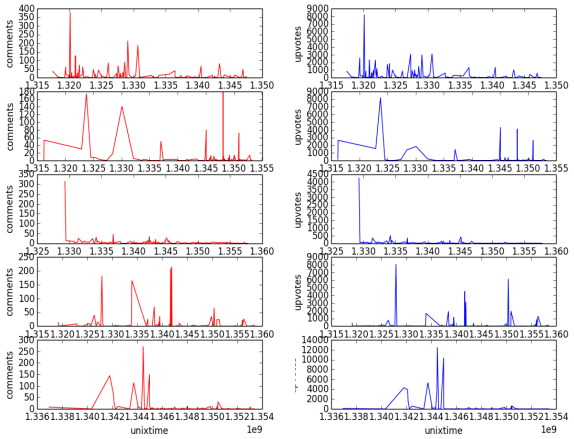

Figure 3: The relation between time and upvotes & comments

From the figure, we can see two things. Firstly, although the absolute values are different, the comments and upvotes have the same change trend as time increases. Thus later we can just discuss one of them. More important, we can see that there are a series of sharp peaks which means, when a image becomes a hotspot, it will also take away attentions on the image with other different titles in a period of time. According to the data set, we find the period is around one month. Actually, it's adhere to our intuition. User flow is huge on Reddit. After a month or more time, when the former hot image is "forgotten", new users can be attracted by the same image again.

Next, we consider the upvotes change trend at a single day. Similarly, we chose 5 days with most images and plotted the relation between time and upvotes as Figure 4 shows.

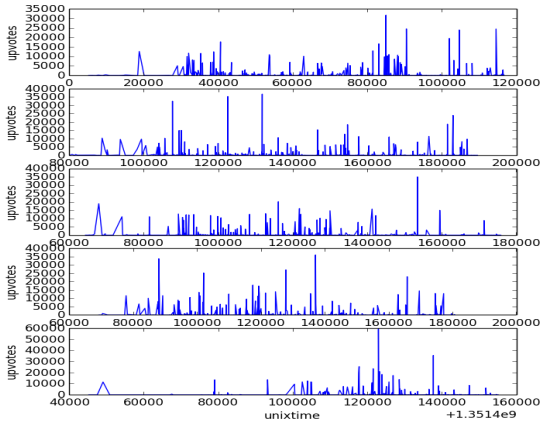

Figure 4: The relation between time and upvotes on 5 single days

We see that the "hotspot-focusing" effect we described above exists here, too. And the peak is more dense here as there are large amounts of image posted in a single day. Still, if you find that there is a hot image on Reddit now, you'd better wait for sometime to get more upvotes and attentions.

What about the trend in macroscopic scale? Trying to explore upvotes change during a year, we summed up upvotes of each month in year 2010, 2011 and 2012, respectively. The result is shown in Figure 5. And from top to bottom, the year is 2010, 2011 and 2012.

A interesting truth: People tend to give an upvotes more on later half of the year. Maybe it's because of the cold weather, when people have less choices of activities, visiting the Reddit website, watching those images for fun can be an attractive way to relax.

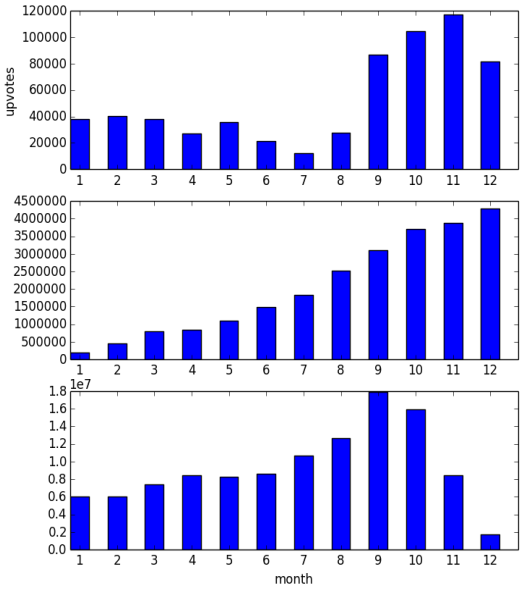

Figure 5: The relation between upvotes and months in different years

Finally let's see how the upvotes change in a longer time period. We use the Figure 6 to show the upvotes trend during 2010 2012. Obviously, later images tend to get more upvotes. It's easy to understand it since the Reddit website becomes more and more popular later. The more people use the website, the upvotes are more likely to be higher.

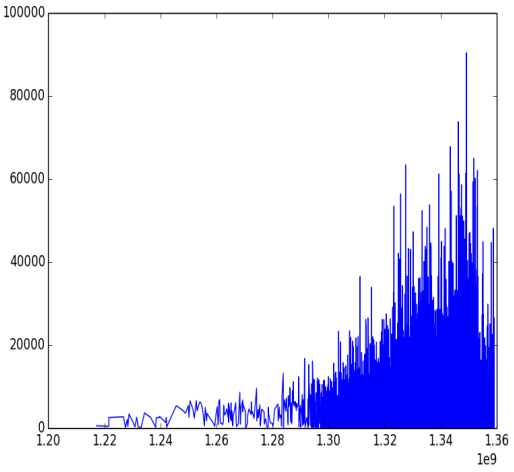

Figure 6: The relationship between upvotes and time

2 PREDICTIVE TASK

2.1 Task Description

We will identify a predictive task which could be trained on our dataset. We would like to predict the number of the comments or the number of the votes when given the rest of the features.

$$\begin{gathered} f(title, username, time, image, comments) \\ = votes \\ g(title, username, time, image, votes) \\ = comments \end{gathered} \tag{3}$$

In the coming sections, we will analyze the features and generate its numerical expression.

3 FEATURE ANALYSIS

3.1 username

This is the section about how to transform the feature 'username' to numerical vector. We know that the vector must represent some characteristic of the feature 'username'. Each distinct username is the representation of a user. And we should analyze which numerical features can reflect the characteristic of a user.

It is tough to extract useful information about the username itself because most of the usernames are meaningless, which are just the combination of either the use's real name or random numbers or the name he/she likes. So we start to find whether there are some valuable information about the user's behavior.

The behavior of the users we can extract from the given dataset is the information of their posts, including the number of the posts, the number of the upvotes/downvotes/comments of the posts. So we can use the combination of these features to generate the corresponding vectors.

From our research in the section 1.2.1, we find the number of posts cannot represent the characteristic of a certain user when our prediction is votes/comments. However, there might be some relation between votes and comments. We assume that the average number of the votes have positive correlation with the average number of the comments a user has. We calculate the correlation coefficient of the average number of the comments and votes and we get 0.73843, which indicates higher comments are likely to give higher votes. So we can use comments to predict votes and use votes to predict comments. The relation between votes and comment is shown in Figure 7.

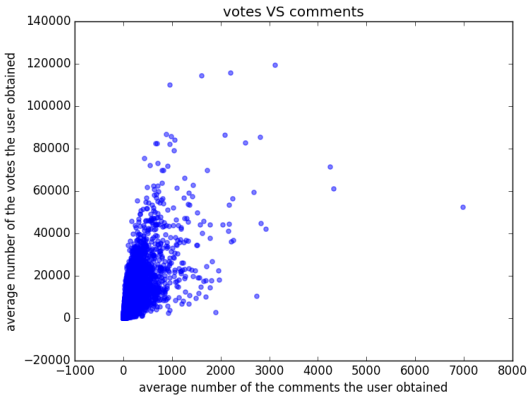

Figure 7: The relation between obtained comments obtained votes for a certain user

To achieve the fast convergence when applying the gradient descent, all of the features should have the same scale during the training. We notice that the number of the comments/upvotes/downvotes per post is about $10^2 - 10^4$. This is too large for a numeric vector for we want to keep the value of every numerical feature between -1 and 1. Thus, we choose zero-mean normalization on our statistics. Let us assume x is the original value and the x' is the value after normalization. $\bar{x}$ and σ are the mean and the standard deviation of all value set.

$$x' = \frac{x - \bar{x}}{\sigma} \tag{4}$$

3.2 title

3.2.1 Linear Regression

Analyzing titles involves text mining techniques. One way of doing this is to dissect titles into

words and find the relationship between words and upvotes, downvotes and scores.

For the first step, we gather all existent words in the title database. If we only wipe out punctuation, there are 38064 words. However, stopwords like "I", "a" don't have many important meanings. If we erase them, this won't affect the final result. After we remove English stopwords, there are 37926 words now. In particular, we plot a wordle for words that appear in titles shown in Figure 8.

Figure 8: Words in Titles

Another approach to shrink the size of the word set is to use stemmer by merging different inflections of words. If we do this, we can progressively reduce the size of words to 29712. However, using stemmer may lose some useful information.

Next, we cannot consider all these words because it is too large. We take top 1000 frequent words as objectives and continue the work.

Next step is to generate title features. For each title, we count the number of appearance for each word. We can speculate that this feature must be very sparse because every title is roughly 15 words and yet the word set is 1000.

Lastly, we can use a linear model to conduct sentiment analysis. We built a predictor of the form:

$$f(text) \to rating \tag{5}$$

using a model based on linear regression:

$$score \simeq \alpha + \sum_{w \in text} count(w) \cdot \theta_w \tag{6}$$

Table 3.2.1 shows the top 10 words that have largest θ and Table 3.2.1 shows the top 10 words that have the smallest θ.

Table 2: Top 10 Words with Largest θ

Word	θ
thi	375.36689533327387
design	370.77194586518766
poster	262.3362706628007
pic	251.98774743710447
knight	218.09458076401967
wife	177.94441315048164
card	167.94558778365197
sign	161.5614698065107
nation	160.15347064822578
appear	159.79323863114442

Table 3: Top 10 Words with Smallest θ

Word	θ
alway	-158.7537515603311
dark	-165.97637820032725
sound	-166.83268609096297
hater	-168.84254646515029
noth	-169.33669158148982
favorit	-172.81530630839512
everytim	-179.69965700008913
elect	-203.09605660731921
edit	-221.74216777775746
everi	-309.22441574319481

3.2.2 TF-IDF feature

TF-IDF is short for term frequency-inverse document frequency, and this feature is widely used in text mining and information retrieval. We can use this feature to depict the importance of a word in a document or corpus. The importance increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus.

Typically, TF-IDF is composed by two terms: the first computes the normalized Term Frequency (TF), aka. the number of times a word appears in a document, divided by the total number of words in that document; the second term is the Inverse Document Frequency (IDF), computed as the logarithm of the number of the documents in the corpus divided by the number of documents where the specific term appears[3].

TF measures the frequency of a word in a document. Since every document differs in length, there is possibility that a term would appear much more times in long documents than it appear in shorter ones. Thus, the term frequency is often divided by the document length as a way of normalization.

$$TF(t,d) = \frac{\#\ of\ t\ in\ d}{Total\ \#\ of\ terms\ in\ d} \quad (7)$$

IDF measures the importance of one term. Since stopwords like "I", "a" occur frequently without important meanings, we need to weigh down these frequent words and scale up some rare words.

$$IDF(t,D) = log_e \frac{\#\ of\ documents}{\{d \in D : t \in d\}} \quad (8)$$

Using library functions, we can get the titles' TF-IDF features. If we only consider single words, then the dimension of each vector is approximately 4600, and if we take 2-grams into account, the dimension would expand to roughly 7800, which is too large to be used as an input feature. Hence we cannot directly utilize this feature and instead do some extra work to limit the dimension of a feature to reasonable size.

3.2.3 Intuitive Text Features

For the mission of predicting the score of the post, we are actually predicting whether a post is popular or not. It is easy to come up with features that may directly influence whether a title is attracting or not. For example, a title can kill readers' interest if it is too long. Similarly, if the title is written in all upper letters, it can convey a strong feeling, maybe a mood of surprise to attract readers. Some other features we collect are whether particular punctuations like '...' and '!' are in the title because they express special feelings of the author as well.

The subreddit is kind of implication of the text title's type which should influence the audience group a lot and thus affect the feedback the post get. We try to previously peek the distribution of subreddit in different feedbacks. To do this, first we use the score as representation of the 'feedback', sort all the score from low to high and plot the score - percentage figure.

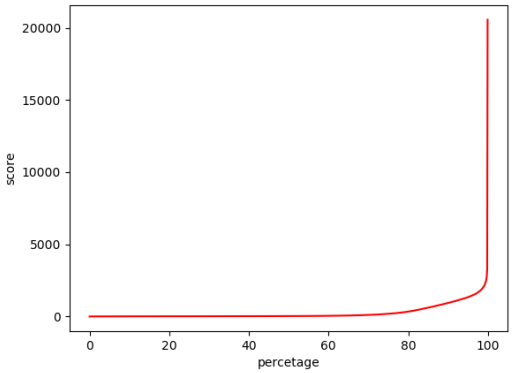

Figure 9: Score Distribution

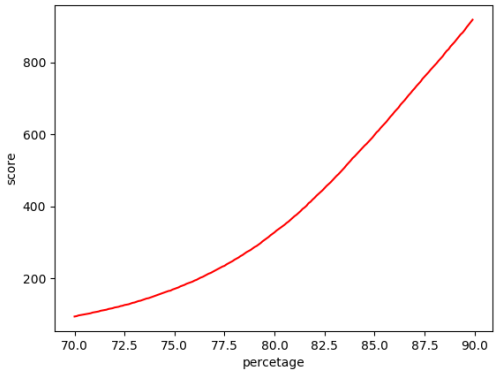

Figure 10: Local Score Distribution

From the picture, it is safe to say that most posts are low-scored and very few of them can get score up to one thousand. The local picture shows one part that is almost linear. Based on the results, we set four bounds : percentage 65, 80, 90 and 99. Thus we get 5 classes of low, good, high, hot, top posts. Then we count the different subreddit for each of the class and sort them from high to low. As we assumed, although there are fixed popular subreddit in all classes, their distribution and ranks in each class are different.

Table 4: Top 10 subreddits in 5 feedback classes

	Top 10 subreddits in 5 feedback classes
0	real, ve, seen, eat, life, bitch, pleas, watch, one, old
1	'funny', 'pics', 'WTF', 'gifs', 'aww', 'atheism', 'gaming', 'reactiongifs', 'AdviceAnimals', 'trees'
2	'funny', 'pics', 'WTF', 'gifs', 'aww', 'atheism', 'gaming', 'reactiongifs', 'trees', 'AdviceAnimals'
3	'funny', 'gifs', 'pics', 'WTF', 'aww', 'atheism', 'gaming', 'GifSound', 'AdviceAnimals', 'reactiongifs'
4	'funny', 'gifs', 'pics', 'WTF', 'aww', 'GifSound', 'atheism', 'gaming', 'AdviceAnimals', 'reddit.com'

3.2.4 k-means clustering

One way of transforming titles into feature vectors is to divide titles into several groups and we then can use its cluster information to create corresponding features.We first decided to use k-means clustering algorithm to do this work.

Here we use TF-IDF features as inputs for k-means algorithm. We set the number of ideal clusters to be 10, namely we expect to separate all data to 10 clusters and thus we can use at most 10 dimensions to describe the title.

Using sklearn libraries, we can implement this algorithm. In order to check the clustering results of k-means, we first examined the top 15 words in each cluster. The result is shown in Table 5.

Another way is to check the distribution of data. The problem is that our word matrix has thousands of words for thousands of documents. Hence, in order to plot these clusters, we can use t-SNE tool to visualize high-dimensional data. It converts similarities between data points to joint probabilities and tries to minimize the Kullback-Leibler divergence between the joint probabilities of the low-dimensional embedding and the high-dimensional data[4].

However, t-SNE can be very time-consuming when the input data is large. To accelerate to computation of t-SNE, we first do dimensionality reduction using SVD to shrink the dimension of TF-IDF features. Truncated SVD works well on TF-IDF matrices and this case is known as latent semantic analysis (LSA). For LSA, a desired dimensionality of output data is 100[5]. Second, plotting the total 130,000 data doesn't help at all. Instead, what we do is, from each cluster, we randomly select some data as representatives and we compute t-SNE only on those selected data. In our process, we selected 250 data in total.The clustering result is shown in Figure 11.

Table 5: Top 15 Words in 10 Clusters

#	**Top 15 Words in Cluster**
0	quot, googl, post, say, search, reddit, disappoint, feel, pictur, just, like, look, someon, i'm, best
1	day, cake, feel, reddit, miss, realiz, today, work, everi, just, post, bad, karma, gif, happi
2	feel, post, reddit, upvot, downvot, page, tri, new, work, brows, comment, someon, r, make, i'm
3	don't, know, think, whi, like, just, want, say, understand, worri, care, mess, anymor, mind, rememb
4	time, everi, favorit, gif, make, laugh, reddit, tri, r, post, feel, dinner, think, someon, play
5	fuck, given, polic, yeah, singl, i'm, shit, don't, oh, just, actual, like, guy, zero, jesus
6	like, look, boss, feel, reddit, heard, guy, imagin, cat, girl, i'm, just, think, thought, friend
7	just, cat, reddit, oh, gif, i'm, friend, guy, think, look, whi, best, love, dog, make
8	true, stori, love, sad, hero, good, veri, come, funni, dream, know, friendship, sure, mean, wish
9	r, post, x, funni, xpost, pic, wtf, gif, aww, reddit, upvot, spacedick, new, atheism, gonewild

From Figure 11, we can see that k-means works as we expected.

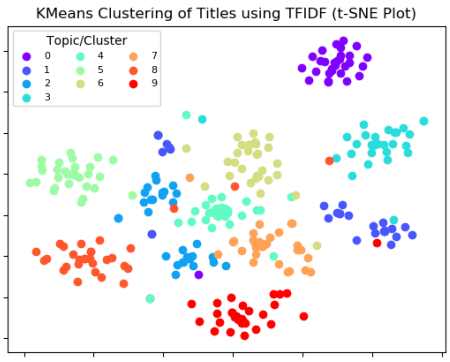

Figure 11: k-means Clustering of Titles

Apart from the above, we also have some interesting discoveries. When we try to validate the clustering result using "subreddit" information in the original data, we find that those posts with the same "subreddit" feature don't squeeze into one or several specific cluster. Instead, they spread in all clusters, namely the distribution is somehow even, as is shown in Figure 12.

However, when we switch to "image_id" information, the result turns out to be exciting.As is shown in Figure 13, we can see that same images seems to gather in one or several clusters. That's to say, the clustering is at least reasonable from the perspective of images.

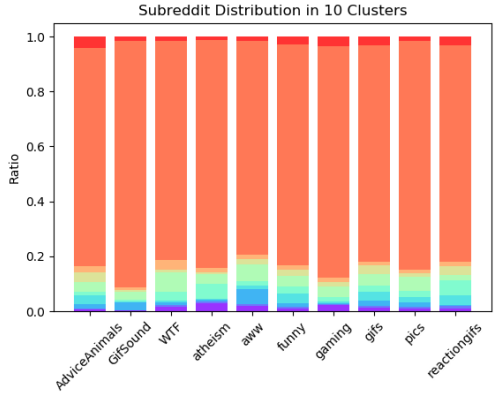

Figure 12: Top 10 Subreddit Distribution in 10 Clusters

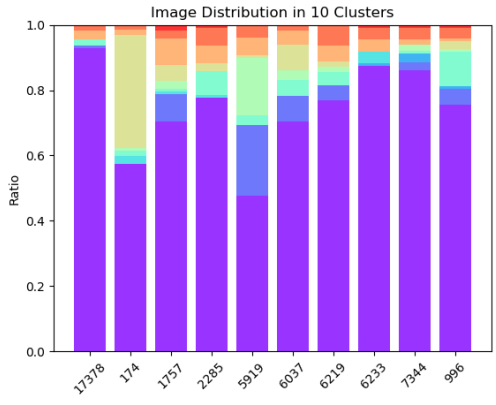

Figure 13: Top 10 Image Distribution in 10 Clusters

3.2.5 LDA model

The fifth way to draw features from the titles is to get the latent topics behind the titles using Latent Dirichlet Allocation model. LDA model assumes there is a latent random variable decides the title's topic with distribution P(topic) and based on the topic, the words in the title are derived by the conditional probability P(word|topic). The task of LDA model is to learn the prior probability P(topic) and conditional probability table P(word|topic) from the titles (Usually by EM algorithm). After the structure of the model is learned, we can then using a specific title and the observed words in it to predict it's latent topic distribution, namely, the posterior distribution of P(topic|observed words). The extracted features have the length of the number of topics N and each dimension of a specific title is represented by the posterior probability of the title belonging to that topic.

Compared with k-means method, the LDA model's feature is not an exclusive one. It acknowledges that a title might be a combination of different topics by their posterior probability given observed words in that title. But to make a visualized result, we can still choose the most likely topic of each tile to be it's 'cluster' and draw pictures similar to those in the k-means part.

Before implementing this feature extraction, we use the title texts after dropping the stop words, filter them with the word stems and turn the words into the TF-IDF representation. Then we set the topic number to be 10, 20, 50 and train our model using the LDA model. To show topics we learned, we give the top 10 words (top 10 posterior probabilities) for all the topics under 10-topic and 20-topic models.

In the 10-topic model, we can find the top words in each topic can make sense if put together, for example the topic 3 includes downvot, upvot, post, so titles related to this topic may have strong correlation with the posts on reddit.

We then show the result of 10-topic model using the same method in the k-means part. First, we show the scattered points in t-SNE.

Table 6: Top 10 Words in 20-Topic model

	Top 10 Words in Topics
0	pole, realiz, seem, brows, got, kill, home, camel, legit, food
1	best, ever, happen, find, ask, kitten, american, imagin, call, man
2	cat, cute, break, realli, say, level, stand, aww, morgan, titl
3	day, cake, hate, gonna, reaction, redditor, karma, hater, soon, link
4	wtf, quot, happi, page, front, googl, birthday, awesom, costum, boss
5	head, danc, girlfriend, meanwhil, guess, show, better, enough, imag, boy
6	oh, downvot, shit, get, upvot, post, comment, god, feel, want
7	time, everi, cakeday, come, still, laugh, alway, first, favorit, gif
8	parti, today, night, see, saw, last, year, stori, use, found
9	love, obama, hit, noth, bill, moment, bee, romney, rainbow, point
10	nom, funni, yeah, said, think, word, us, told, ll, view
11	give, walk, parent, idea, everyon, bro, mind, drop, rais, question
12	fuck, facebook, dog, littl, post, feed, internet, face, repost, just
13	need, bad, forev, hard, ass, alon, porn, brother, keep, 5
14	real, ve, seen, eat, life, bitch, pleas, watch, one, old
15	well, true, right, now, favorit, kitti, end, vote, run, everytim
16	nsfw, let, done, play, job, style, back, never, game, around
17	stop, miss, morn, hey, hous, won, meet, 2, class, guy
18	don, know, even, babi, much, pretti, wrong, re, fight, jesu
19	wait, kid, tell, sure, cool, anyon, thought, great, ride, rememb

Table 7: Top 10 Words in 10-Topics model

	Top 10 Words in Topics
0	real, ve, seen, eat, life, bitch, pleas, watch, one, old
1	cat, cute, break, realli, say, level, stand, aww, morgan, titl
2	fuck, facebook, dog, littl, post, feed, internet, face, repost, just
3	oh, downvot, shit, get, upvot, post, comment, god, feel, want
4	love, obama, hit, noth, bill, moment, bee, romney, rainbow, point
5	give, walk, parent, idea, everyon, bro, mind, drop, rais, question
6	best, ever, happen, find, ask, kitten, american, imagin, call, man
7	day, cake, hate, gonna, reaction, redditor, karma, hater, soon, link
8	wait, kid, tell, sure, cool, anyon, thought, great, ride, rememb
9	parti, today, night, see, saw, last, year, stori, use, found

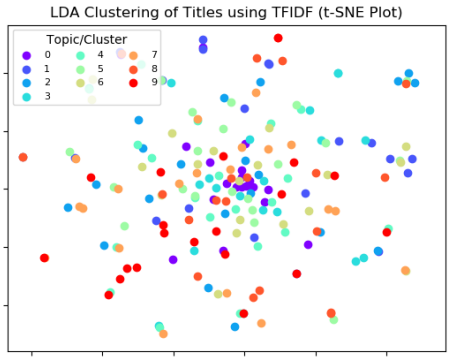

Figure 14: LDA topics of Titles

As LDA model does not cluster the titles based on direct features, the result is not clustered like k-means above.

Next, we would like to find whether the subreddit or images fall into a specific topic. The distribution of topics on them are shown below.

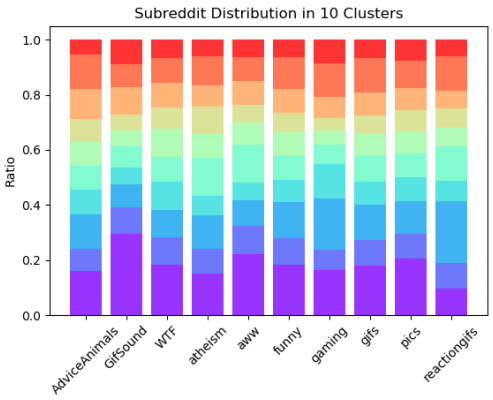

Figure 15: Top 10 Subreddit Distribution in topics

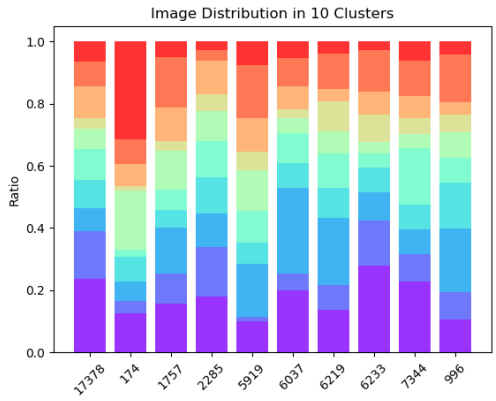

Figure 16: Top 10 Image Distribution in 10 topics

The distribution of topics on subreddit is very uniform, implying that the topics are relative independent of subreddit titles.

4 MODELS

4.1 Initial Linear Regression Model

Firstly, we consider the easiest model to predict the comments. The feature for prediction consists of three parts: Title features, User features and Time features. Using 5 classification methods, we can get the extracted title features respectively. User feature comes from analysis of username from former section. In this model, we used normalized unix time as time features. Then we established a linear regression model:

$$y = X\theta \tag{9}$$

y is the number of comments we need to predict. X are the features. After splitting the data set into 3 parts: Training set, Validation set and Test set, we compute the MSE to evaluate the model. The result are shown as Table 8:

Table 8: MSE of predicting upvotes using Linear Regression Model

Title feature	**Training Set**	**Validation Set**	**Test Set**
TF-IDF feature	62685876.02	78969653.21	59304321.57
k-means feature	84248823.33	68937312.98	68374582.84
LDA feature	93841239.31	75344245.17	67482931.34

Table 9: MSE of predicting comments using Linear Regression Model

Title feature	**Training Set**	**Validation Set**	**Test Set**
TF-IDF feature	549932.32	899312.12	734249.41
k-means feature	603123.94	834293.25	684341.33
LDA feature	713942.03	674302.64	784309.37

We can see it is a trivial model. According to the raw data we could conclude the reasons for bad performance: i) Most user only posted a popular image once. Later they never posted any other images or the later images attracted few interests. There are no celebrity affection in this data set. Thus the user feature would have negative influence on model. One successful image increased the user feature value, however, the comments on later images from the same user were low. It actually makes the prediction worse. ii) The title cluster is not that useful too. In raw data, we can find many resubmitted images with similar titles. We use an example image to show this as Table 9 shows:

Table 10: Different titles of image 10005

Title	**Comments**	**Upvotes**
Troll Cat	33	2664
Troll Cat: Piracy (x-post from r/gifs)	14	10
Troll Cat: Piracy	11656	245
TrollCat	45	0
Cat = Master Troll	41	2
Troll cat	33	2
When you poke someone and they think it was somebody else.	134	0
Trollin	227	5
Troll Cat	2527	19
trolling cat	7	0

Table 11: MSE of predicting comments using Linear Regression Model

	Training Set	Validation Set	Test Set
MSE	10504.61	18942.94	11235.94

Through the title we could infer that the image shows a cat. In a useful classifier, these titles should be classified into one topic. However, Subtle differences in title can lead to huge change in numbers of comments and upvotes. In fact, simple classifications of titles still lose key information to predict the popularity of the image. Thus, later we will add intuitive text features for prediction. iii) In this initial model, we didn't take time into full consideration. Actually, as analyzed, we need to consider the case that the same picture was submitted and attracted many attentions a short time ago. Thus we used a "resubmit penalty" shows below:

$$penalty[i] = exp(-\sum_{j=1}^{i} \frac{1}{(time[j] - time[*])}) \tag{10}$$

time[*] can refer to two case. One is the first time we submit one image and the other is the time when we judge the image as "popular", with a prediction larger than some special threshold value. time[j] refers to the time we submit the image at j th times. It can be seen that after the image becomes a hotspot, the penalty would be large in a short time. And as time goes by, the penalty gradually decreases until there comes a new popularity peak.

4.2 Improved feature

We improved our feature in the following aspects:

- Deleted the user feature.
- Added the intuitive title features.
- Added the community feature (subredit)
- improved the time feature.

Then we applied the linear regression model to predicting upvotes and comments. The results are shown in Table 11.

After improving the features, we can see that result is better than before. Then we made more efforts and tried a non-linear model.

4.3 Multi-layer perceptron Regression

In this section, we used a Multi-layer perceptron Regression (MLPRegression) that trains use backpropagation with no activation function in the output layer.

A Multi-layer perceptron is kind of neural network consisting of at least three layers as shown in Figure 17.

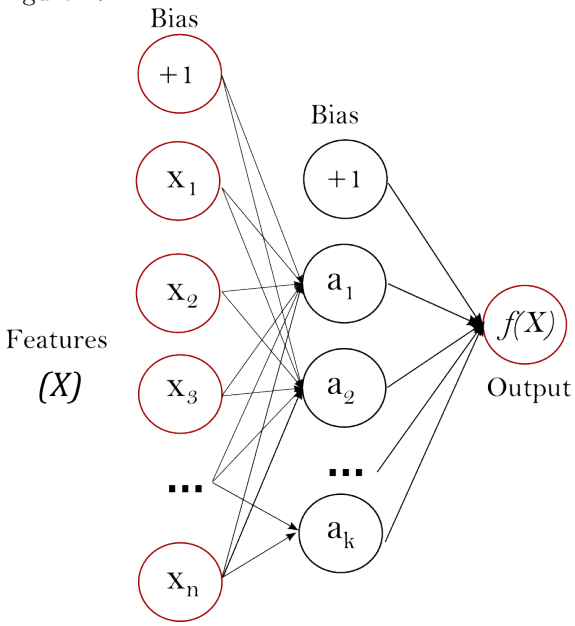

Figure 17: The structure of a 3-layer MLP

In this model, we finally chose a three-layer network with (60,40,20) nodes respectively. There are four activation functions for the hidden layer and we chose the logistic function finally:

$$f(x) = max(0, x) \tag{11}$$

Then with a training set $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), ..., (x^{(m)}, y^{(m)})\}$, We define the overall cost function as:

$$J(W,b) = [\tfrac{1}{m}\textstyle\sum_{i=1}^{m}(0.5||h_{W,b}(x^{(i)}) - y^{(i)}||)^2]$$

$$+0.5\lambda \sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}(W_{ji}^{(l)})^2 \tag{12}$$

Our goal is to minimize J(W,b) as a function of W and b. To achieve this, we use the iteration of gradient descent to update the parameter W, b as follows:

$$W_{ij}^{(l)} = W_{ij}^{(l)} - \alpha\frac{\partial J(W,b)}{\partial W_{ij}^{(l)}} \tag{13}$$

$$b_{ij}^{(l)} = b_{ij}^{(l)} - \alpha\frac{\partial J(W,b)}{\partial b_{ij}^{(l)}} \tag{14}$$

α is learning rate. These partial derivatives are complicated and here comes the backpropagation algorithm, which gives an efficient way to compute these equations. We will skip this part because of the space.

As the result in Table 12 shows, this model performs well compared to previous linear regression model.

Table 12: MSE of predicting comments using Linear Regression Model

	Training Set	**Validation Set**	**Test Set**
MSE	5349.22	6493.02	8526.22

5 CONCLUSION

In this project, we first explore the properties of the overall data set and try to make clear what features can be used to predict the popularity(upvotes, comments, scores, and so on) of one image. Next, we respectively extracted user features and classified the topics of the titles by 5 different methods. After establishing feature vector we built a linear model and a non-linear model for prediction. Improving feature and model itself step by step, we finally get a good result. In fact, there is still much room for improvements:

- We can extract "bad words" used as features. The best titles vary a lot for different images, but those "bad titles" can have some commonness.

- The better model to describe the time effect is not so good. We believe if given more time, we can achieve a better prediction model.

At last, in this course, we learned many classical methods for web mining and recommendation systems. These knowledge is a good basic for our future study in Machine Learning field. Thank Professor and all the TAs!

References

[1] H. Lakkaraju, J. J. McAuley, J. Leskovec What's in a name? Understanding the interplay between titles, content, and communities in social media. ICWSM, 2013.

[2] https://en.wikipedia.org/wiki/Reddit,2017-11-26

[3] http://www.tfidf.com/

[4] http://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html

[5] http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.TruncatedSVD.html

[6] http://scikit-learn.org/stable/modules/neural_networks_supervised.html#regression

[7] http://ufldl.stanford.edu/wiki/index.php/Backpropagation_Algorithm