

CS579 Final project Report

Jiada Tu
Lv Zhang

1. Introduction

With the enormous growth in social networking, Twitter is significant service that allows users to share information called “re-tweet” for personal and commercial utilization. There are many tools for Twitter data analysis. For instance, company may want to know how does their promotion news spread and the individual would like to care about how popularity of their personal tweet. They need a tool that capable to create the link between the data on virtual network and the real world. In this project, we were trying to build a system that can predict how popular a tweet will be based on a user’s Twitter profile or the history. In addition, a methodology for predicting re-tweets in a manner patterns would be presented. And we also can judge what terms in the tweet contents would most likely affect the number of re-tweets by sentimental analysis.

Hypotheses and assumptions:

- 1) It is possible to predict the retweet-count
- 2) Some attributes like location info, amount of followers, images/videos, posted URL or number of days passed by, etc. will affect the number of retweets

2. Data

When we get a tweet from the twitter API, the tweet contains a field names “retweet_count”. This means how many people click the “retweet” button under the tweet. We call this “system retweet”. But many people likes to “manually retweet” the tweet, which means that they retweet a tweet by post a new tweet: typinh their comment/reply and add “RT @xxx [original tweet content that be retweeted]” behind. This kind of “retweet” is not recorded by twitter system and it's hard for us to collect the “manual retweets” of a tweet. So, we just focus on the training and predicting “system retweet” with our model.

We didn't trace a certain tweet and record it's retweet-count changing as time goes by. Instead, we collect discrete data points which contain tuple (“time passed”, tweet) in our data set.

Because the retweet count of a tweet increase as time goes by, retweet count is related to when we collect that tweet. For each single tweet that we collect, we record the timestamp when we get that tweet. So, through “(the time we collect the tweet) – (the time the tweet is posted)”, we can get the “time passed” information. Tuple (“time passed”, tweet) become a data point in our dataset.

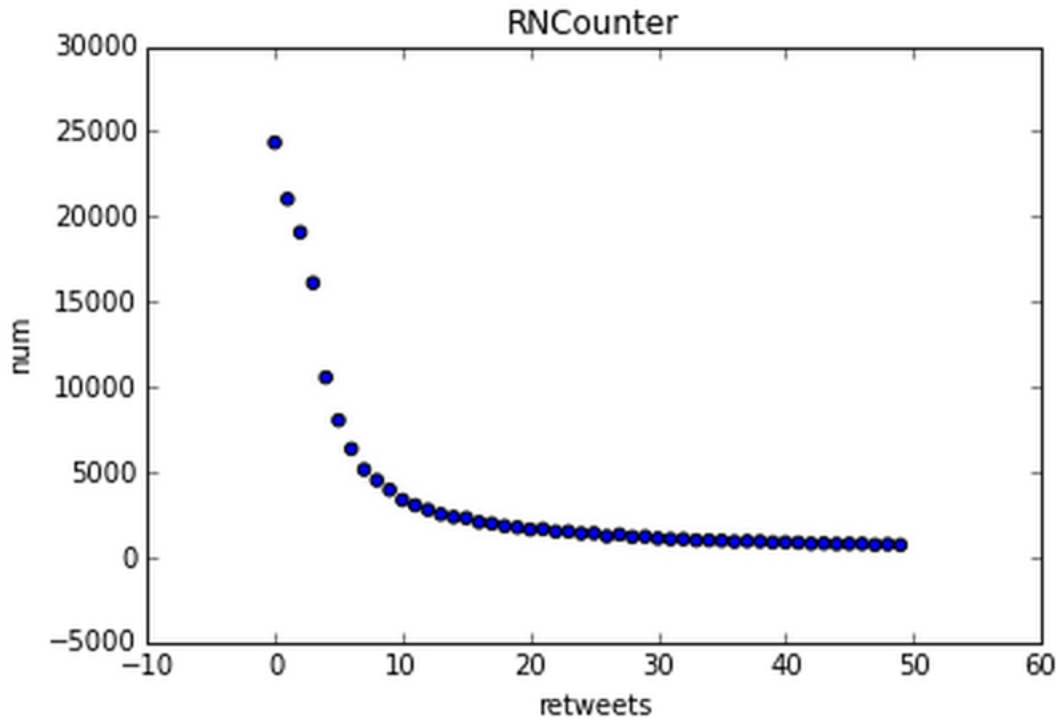
We only collect the tweets that's in english, because some languages are hard to tokenize and it eases the rest work.

We collect data using both Stream API and REST API, because each of them has some benefits and they can make up each other.

For Stream API, it doesn't have the 15 min limitation. But it only returns tweets which are in the most recent few seconds. So, the data we collected by this API will never contains data points that has 0 retweet count and “time passed > 30 sec”, because its original tweet is the only chance that can be shown by Stream API.

REST API don't have this problem. We use Stream API to get user id from the recent sample tweet and get their user_timeline(600 at most). But it's slow base on the 15 minutes limitation.

One problem of the data we collected is that it's highly imbalanced. More than 70% of all tweets in REST API has 0 retweet count, and more than 90% of the rest have 1 retweet count. This **may** make our model learned wrong parameter (Especially linear regression. Because as times passed, many tweet still gets 0 retweet. Then the linear regression will think “time passed” is not related to the retweet count at all, when in fact, it is). So we try to decrease the ratio of the data points that has 0,1,2 retweet counts, but not so much that makes our data set totally unrelated to the real ratio. Then we get the below retweet-count distribution:



The x axis is the retweet count and the y axis is the number of the tweets that has this certain retweet count.

We mix the data collected by these two API. After weighting, balancing and sampling(the data set is too large to fit in memory when do the training and predict) the data, we get 350K tweets.

One more thing. Actually we classify the tweets into 24 different files base on which hour (UTC time) in a day that the tweet is posted. We do this in the assumption that the retweet pattern of at tweet is different if it's posted in midnight from if it's posted at afternoon. So then we can train the model separately with these 24 data sets. But the tweet is global, so this assumption maybe wrong. Until now, we haven't do the training and predicting base on this assumption, what we do is just using all 24 datasets. We put this into future work to figure out if this assumption can improve the accuracy of our model.

3. Methods

Because it's hard to predict the exact number of a tweet, we train and predict the $\log_2(\text{retweet num})$. Then if our predict is 2 different from the real num, then it's within $[\frac{1}{4}x, 4x]$ range, when x =the real retweet num.

We figure out some attributes that may related to the retweet count. Mainly: the followers of the user, the friends of the user, the time passed by, the list that user is a member of, the number of status

that user have (total number of tweets), the sentiment of the tweet / user description, if the tweet has url, hashtag and image, etc.. The number of some attributes can be very large (like time-passed, followers number, etc.) and we think it will not make much difference if it's "very large" or "very very large", so we do $\log_2()$ of them.

For the sentiment analysis, we use word list with sentiment score. For each tweet / user description, we tokenize them, match the tokens to the word list and add the sentiment scores (separate positive and negative score). These scores became a part of the attributes of the dataset (X matrix).

For the model, we do both regression (mainly) and classification. To evaluate the regression, we use Pearson correlation coefficient, mean squared error and relative squared error. To evaluate the classification, we use the accuracy score.

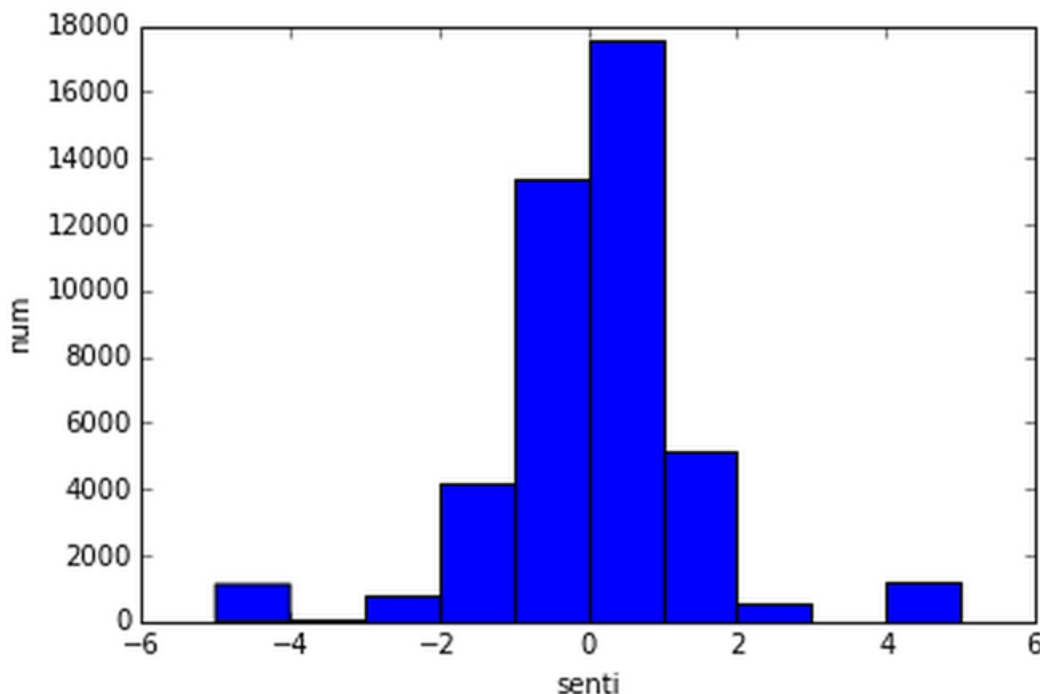
For regression, we use linear regression at first. Then we figure out that most of the parameter is not linearity with the retweet count. For example, assume $y = \text{retweet count}$, x_1 and x_2 are some parameters, then y may not equals " $x_1 + x_2$ " but " $\log(\sqrt{x_1})\text{pow}(x_1, 2) + x_2$ ". Also, y may not in relationship with them as $y = ax_1 + bx_2$, but $y = cx_2 * x_2$ or $y = d x_1^{x_2}$. So in these situations linear regression is not good and we need some method that's not linear. Then we find decision tree. To refine the result of decision tree, we mix bagging method with it. So it becomes the random forest. What it basically does is has many decision tree, each of them are trained on subset of the dataset and attributes. Then each of them will do the predict and the final prediction is the weighted result of predictions by all of them.

For classification, we also use decision tree with bagging.

4. Experiments

A. Sentiment analysis:

We use the word list from [1]. It is based on the Sentiment 140 [2], a twitter sentiment analysis product. It's in the same form as AFINN word list, but it has much more terms: totally 62,468 terms. Because there're some hashtags, mentions and urls in it, it has 43,904 terms after I delete all of them. Here's the sentiment score distribution of the terms in it:

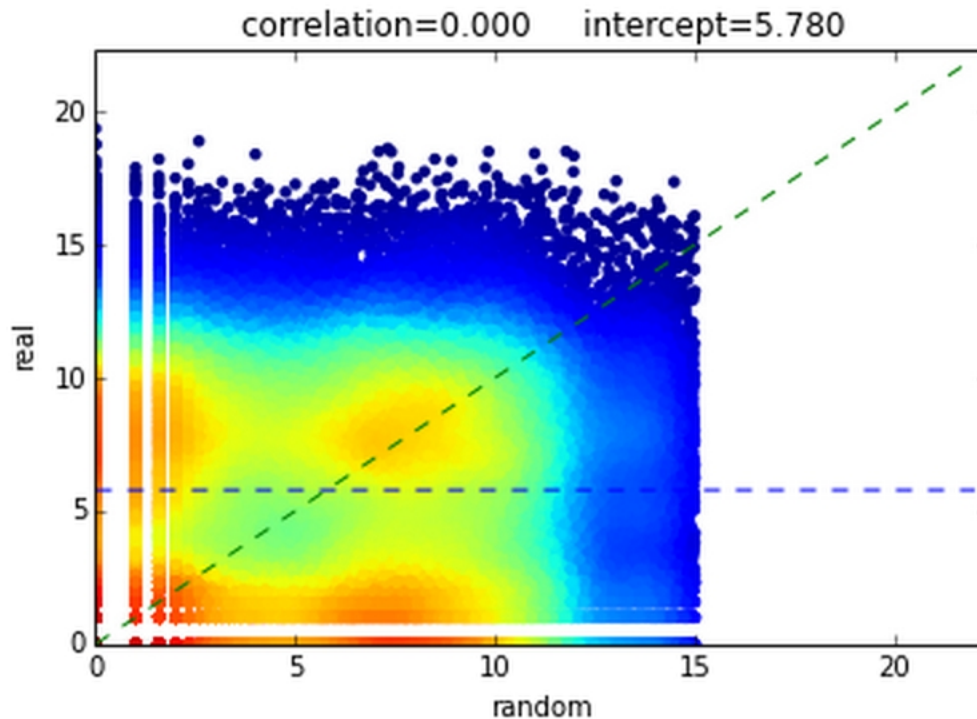


The x axis is the sentiment score, and the y axis is the number of words that has this certain sentiment score.

B. Regression

Baseline:

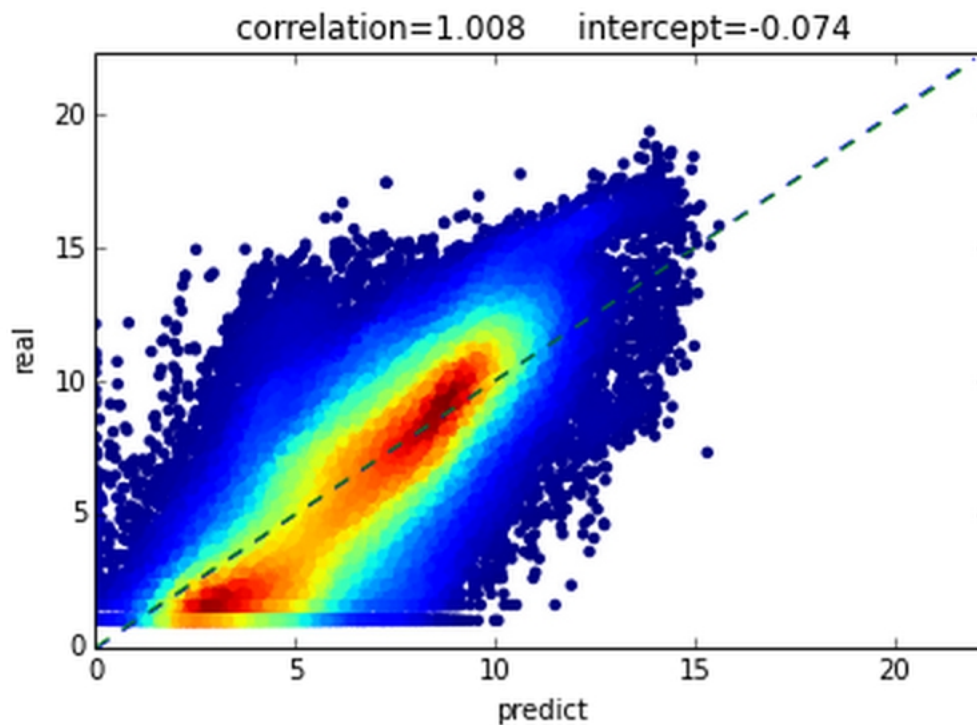
To establish a baseline, we make a naïve predict model that randomly predict the retweet count base on the distribution of the retweet-count of the dataset. So if 20% of the tweets has 0 retweet, then the model will have 20% chance to predict 0 for any tweet. Here's the density chart of the baseline model:



The x axis is the predict $\log_2(\text{retweet_count})$, the y axis is the real $\log_2(\text{retweet_count})$. The green line is just a diagonal $y=x$. The blue line is a line ($y=ax+b$) that try to fit the points in the figure, and the “correlation” is 'a' of the blue line ($y=ax+b$), “intercept” is 'b' of that. **The Pearson score is (0.000156, 0.967), the mean squared error is 5.85, and the relative squared error is -0.986.**

Linear regression:

For regression, we first use linear regression. We train on 80% of the dataset and predict on 20% of the dataset. I didn't do the cross validation because the score is basically the same as when I train the model on 20% and predict on 3% of the dataset (for test), so I believe no cross validation is need now. But we may add that in the future. We get this result:



The Pearson score is (0.730, 0.0), the mse is 2.56, and the rse is 0.533.

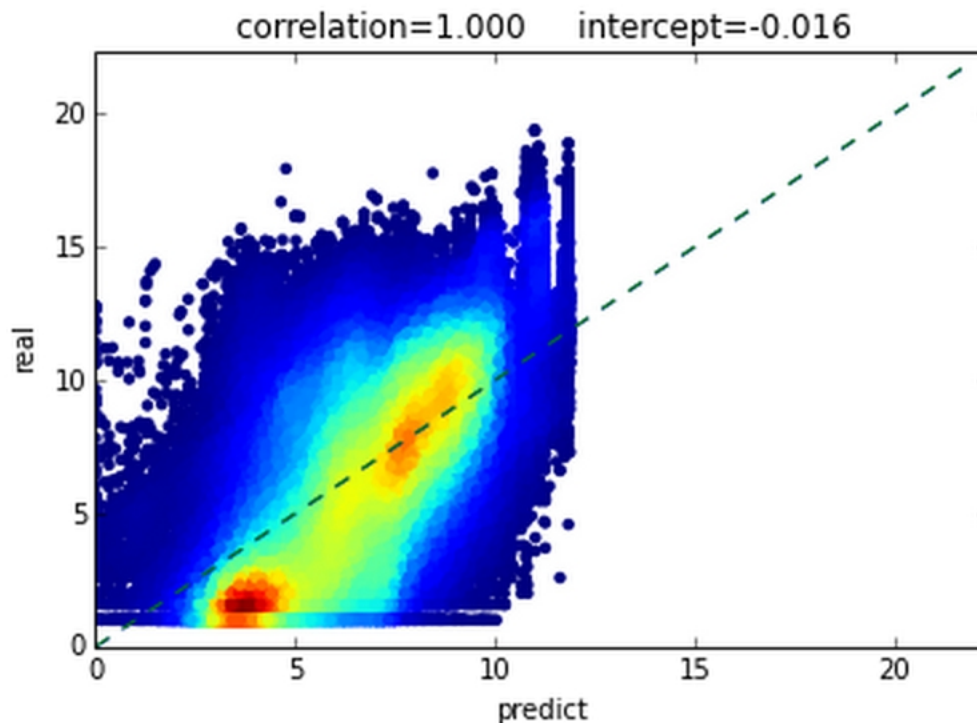
We can see the blue line is basically match with the green line, which means the bias is small, maybe. And the total predict trend to be much better than the random predict model. The mse is still large. Basically it will predict 6 times larger or smaller than the real retweet-count. Also, the tilt of the red area seems to be not match with the $y=x$. We will talk about that later, now let's see the coefficient of the attributes that the model learns:

```
coef=
('log_user_followers_count', 0.65294717363975885)
('if_images', 0.54354283169348672)
('log_time_passed', 0.21827139491805703)
('sentiment_text_positive', 0.20160009284540756)
('sentiment_description_negative', 0.098972500599630003)
('user_geo_enabled', 0.087374527799844101)
('log_user_friends_count', 0.014008642346259292)
('user_favourites_count', 5.28476183718527e-06)
('user_protected', 0.0)
('sentiment_description_ratio', -0.032287350992543741)
('sentiment_description_positive', -0.062946146963049079)
('user_verified', -0.073891436319871023)
('sentiment_text_negative', -0.099132132742753468)
('log_user_listed_count', -0.12081977928248049)
('sentiment_text_ratio', -0.1757830928418638)
('if_hashtags', -0.2372191834298851)
('log_user_statuses_count', -0.26912988491082512)
('if_mentions', -0.61377569617960048)
intercept= -1.97650203225
```

We can see, the log(follower count) matters the most. Basically, if you have 8 times followers as now you have now, all your tweets trends to be have 4 times retweet-count as now. Also, if your tweet has images in it, it will more likely to be retweet. The time passed should be some kind of significant

and it is, so it's good. Also, if `sentiment_text_positive` shows that if you trend to say something positive, then your tweet is more likely to be retweeted. Your friends count doesn't help your retweet-count much. If you mentions someone, then you decrease the possible number that your tweet will be retweeted. I think it's because that if people see a mention in a tweet, normally they will think the tweet is not tend to speak to them, so they have less motivation to retweet that. The more tweet that you post, the less possible retweet number you will get. I think it's because that if you keep posting tweet, people may start get boring with you and pay less attention to you. $\text{Log}(\text{user_listed_count})$ is negative only because it may highly related to $\text{log}(\text{user_follower_count})$. I remove the follower and re-train the model, the $\text{log}(\text{user_listed_count})$ get +0.51, the second highest attribute. Same situation happens to “`user_verified`”. Actually there was a “`if_url`” here, and it's highly negative (-0.46). I thought it's negative only because it trend to be show with image in the same time (actually the image and url are totally separated part), but I was wrong. Even if I remove the `if_image`, the `if_url` still get highly negative value. And totally 12.5K tweets has image in it and 7.9K tweets has url in it, so it trends not to be a noise? I can't figure out why, maybe it's an example that “machine learning wins experimentalism”? Not sure.

As mentioned before, Linear regression has some problems. If we only train and predict base on the `user_follower_count`, we get the below graph:

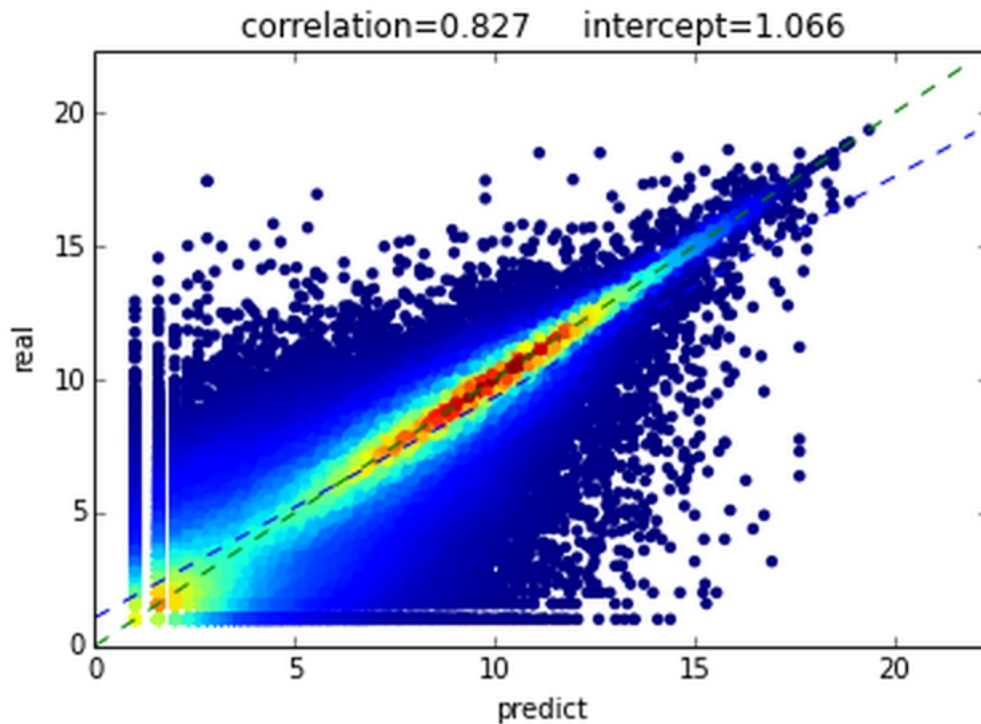


By looking the red field, we can see that $\text{log}(\text{user_follower_count})$ is not in linear relationship with retweet-count. It trend to predict more as real retweet-count is small, and trend to predict less when retweet-count is large.

Also, if two attributes are usually shows together, and one is “a little bit” more predictive than the other, although both of them may be strong predictive attributes, the less predictive one will still get negative coefficient by linear regression. The `user_listed_count` is the example, as it may highly related to `user_follower_count`. So we try to use other machine learning methods.

Decision tree (regression version)

Decision tree (regression version) is the one we choose. There's the result that it learn and predict on the same dataset:

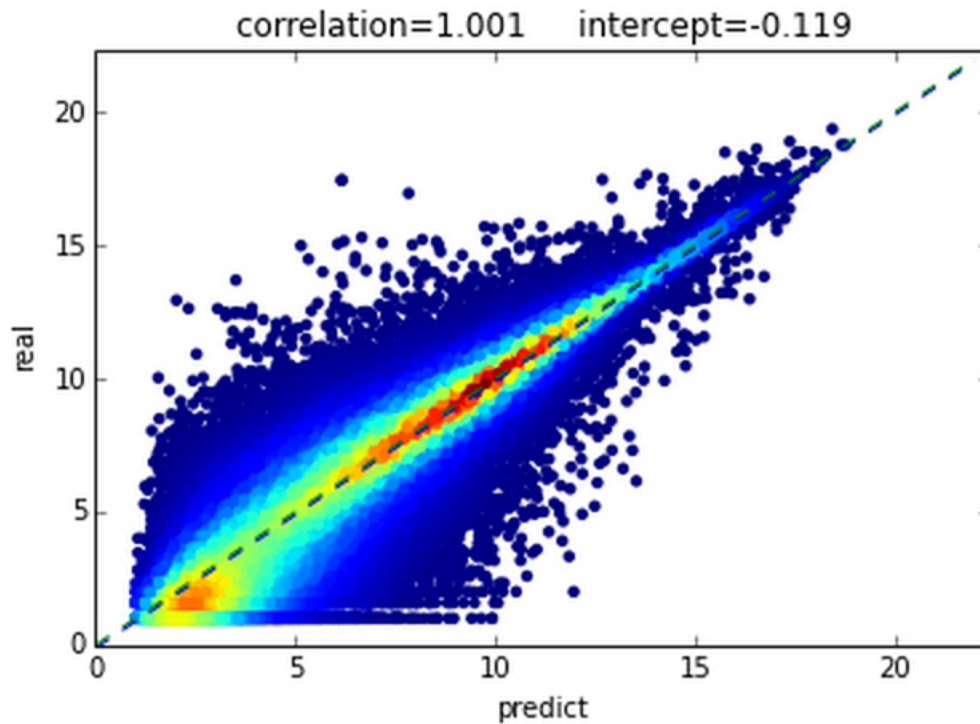


The Pearson score is (0.829, 0.0), the mse is 2.19, and the rse is 0.657.

We can see that the model seems to be nicer, and all the evaluate values are better. But the model seems to be less predictive when the real number is small.

Decision tree with bagging (regression version):

Then we use decision tree with bagging (regression version). The result:



The Pearson score is (0.902, 0.0), the mse is 1.6, and the rse is 0.813.

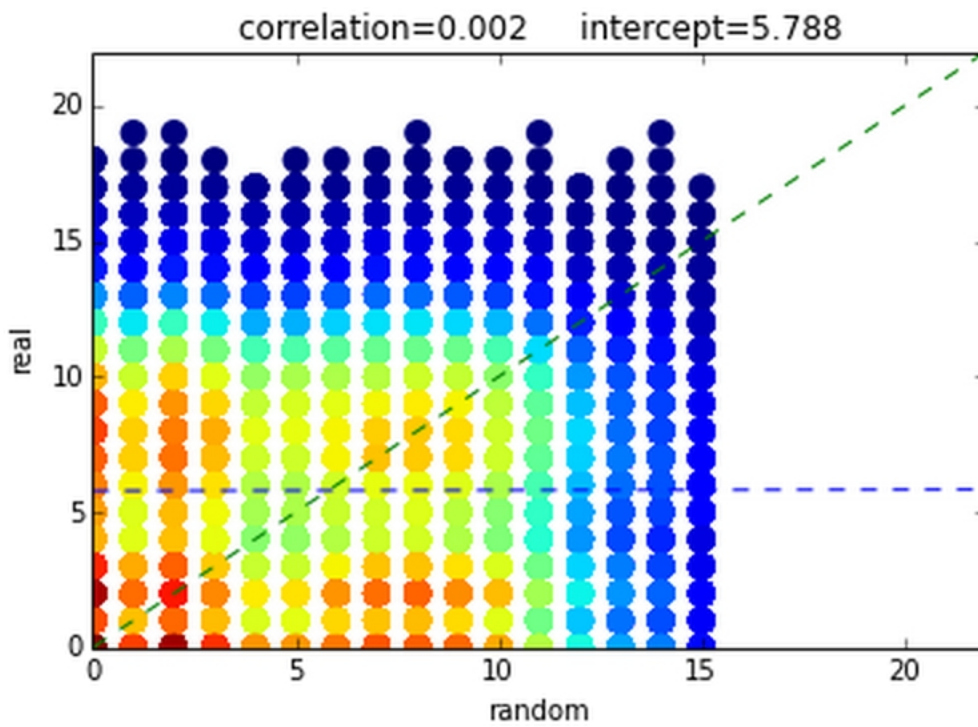
Much better.

D. Classification

To do the classification, we map the $\log_2(\text{retweet_count})$ to $\text{floor}(\log_2(\text{retweet_count}))$ and try to predict the exactly magnitude level (base on 2) of the retweet count a tweet will have (in a certain time passed). So now the value are integer instead of float-point number.

Baseline:

Same as the baseline in regression before:

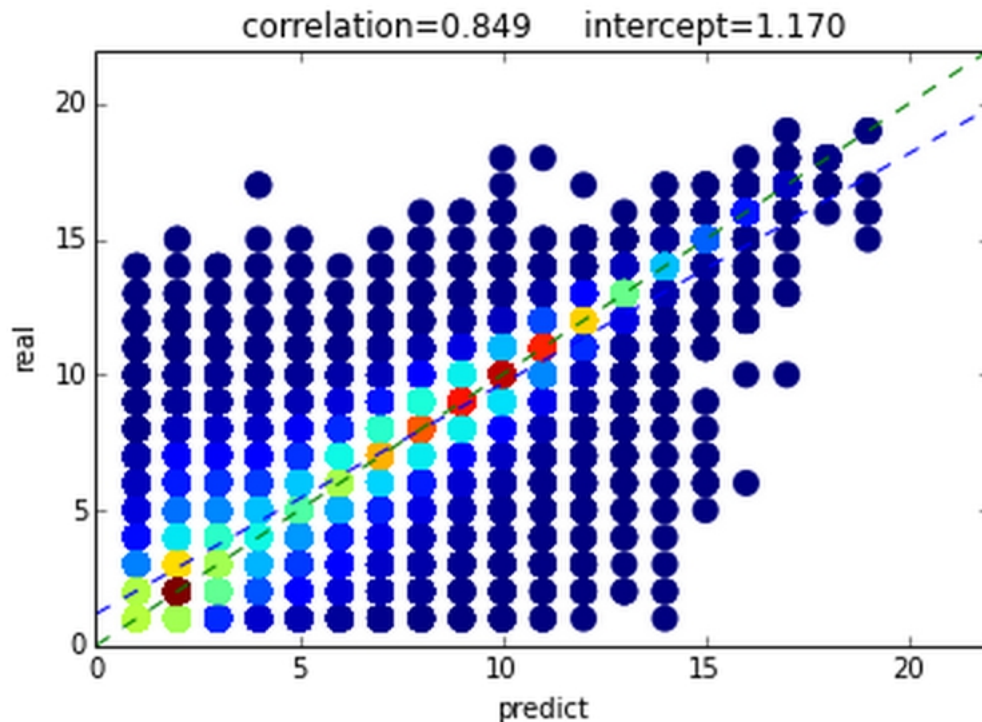


accuracy 0.0757215474648

Notice that the accuracy is 7.6%.

Decision tree with bagging (classifier version):

We use decision tree with bagging (classifier version) to do the classification:



accuracy: 0.382650641812

Notice the accuracy is 38.3% now.

Obviously, comparing to other problems which accuracy can always be more than 85%, 38.3% is really low. But the retweet count prediction is a relative hard problem because there are so much noise in it, so this accuracy may be reasonable?

5. Related work

[1]NRC-Canada: Building the State-of-the-Art in Sentiment Analysis of Tweets, by Mohammad, Saif and Kiritchenko, Svetlana and Zhu, Xiaodan, June, 2013

[2]<http://www.sentiment140.com/>

[3]<http://ciir-publications.cs.umass.edu/getpdf.php?id=1071> //also use image pixel as attributes

[4]http://research.srv.microsoft.com/pubs/141866/NIPS10_Twitter_final.pdf

[5]<https://gigaom.com/2013/04/26/predicting-twitter-popularity-is-all-about-probability/>

[6]<http://homepages.inf.ed.ac.uk/miles/papers/icwsm11.pdf> // use passive-aggressive algorithm

6. Conclusions and Future work

Conclusion:

- 1) Although not that accurate, it is possible to predict the retweet-count, at least the range of the retweet-count.
- 2) Can do the prediction of retweet-count by only using small number of attributes.
- 3) For this problem, decision tree with bagging is much better than linear regression
- 4) If you want to get more retweets: put images in it; say something positive; don't keep posting new tweets rapidly and don't mention people if you don't have to.

Future work:

- 1) Train the model base separately base on the UTC time hour it is post and see if this affect the accuracy.
- 2) Separate the tweets into few parts base on the time passed ([0, 10 min], (10 min, 1 hour], (1 hour, 1day], etc.) and fit the model on them separately.
- 3) Try other machine learning methods which may fit better with the attributes (for example: not (time_passed+follower_num) but (time_passed*follower_num)) to do the prediction.
- 4) Do more analysis / (add more parameters) base on the text and user description. Because now we just do the word-list-match sentiment analysis. There is more we can do with it.
- 5) Maybe even figure a new algorithm to match the retweet count prediction problem better.

7. Contribution

Jiada Tu: coding, analysis and refine the result, write report/ppt

Lv Zhang: write report/ppt, discuss the topic and the method used in the program