Predict Accept Result for Social Network Recommendation

Judy(Di) Zhu dzhu1

Abstract

Users on social networking systems are at risk of overwhelmed information—over-exposure to un-related information will decrease use experience. Recommendation algorithm takes important role in selecting thoughtful and useful information for target users. This paper aims to predict whether the user will follow an item recommended to users. The dataset is collected from real social network following records.

This paper is the final project for Applied Machine Learning, and follows and methodology of the discipline. The results may be useful for social network recommendation researchers.

1. Introduction

Logging into a social network, like Facebook, Twitter, Line, Wechat or Weibo, many recommended information will flush to you. Open them or not? Every user will have different acceptance behavior based on recommendations.

Recommendation system plays an important role in product marketing, it should effectively filter out key information appropriate for each user. Items have been tagged as movies, music, sports or fashion. Besides social networks, other vendors including Amazon, Netflix, Yelp and Jet.com have to recommend prior items for each user’s personalized preferences.

Recommended items can be persons, organizations or news, which has been extracted into different format based different systems. The prediction aims to tell whether an item will be accepted by a user.

In order to deliver the most efficient contents to user, many recommendation systems chose similar tags that user have followed before. Also, the user’s personalized setting and information play an important role in many recommendation systems. This paper aims to conclude previous recommendation methodologies and train recommendation model on collected data. Based on training and optimization process, evaluate and analyze the recommendation systems.

The rest of the paper is organizes as follows: Section 2 introduces the relating work and research background of recommendation algorithms. Section 3 descripts the dataset used in this project. Procedures for improving training model are outlined in section 4. Section 5 is mainly about feature extraction and selection based on collected data. Section 6 gives the baseline performance in initial iterations. There follows optimization process. Then the final result and discussion are delivered in section 8 and 9.

1. Related Work

The project uses data from one popular social network in China—Tecent Weibo[1]. Our dataset includes descriptions for different items, user profile information, user’s following relationships.

The work presents here is closely related to feature extraction—create and select the most useful features based on huge and complex logging information.

Jason Brownlee[2] has underlined that useful features come from several steps: 1.observe at raw state; 2. Devise features based on target problem; 3. Select features based correlations or other parameters; 4. Reevaluate model based on selected features. Refering to the methodology, user profile, user following history and item information have extracted to create features, for example, for each training item, the user’s gender is added as a feature, then the correlation between gender and accept result is calculated; after confirming the contribution of the feature to the final result, the feature is added into training set.

When it comes to detailed feature extraction, especially for content based system, [3] [Chia-Cheng Hsu](http://www.sciencedirect.com/science/article/pii/S0898122112003161) has came up with an artificial bee colony algorithm, which is based on similarity of item content and user preference tags. Inspired by his method, we utilize common tags to evaluate whether the user has overlapping interest with the provided items. Also, [4] in “A Social Network Caught in the Web”, Lada has estimated that user personal information like gender, birth year have great influence on their interest, which helps estimate acceptance for certain recommendation. Following Lada’s research, many personalized feature based on user profile have also been devised and evaluated.

Based on different available information and logging, many other industrial recommendation algorithms have also been applied. [5] Frank Edward Walter built a trust-based recommendation system based on dynamics among agents and network density. Utilizing content information, but calculating the weights carefully, Debnath[6] has devised a purely content-based recommendation systems. In order to avoid data sparsity problem, Pham[7] came up with clustering method to complete the collaborative filtering for social network data.

1. Data Collection

In order to understand user’s behavior, the project uses data from one popular social network—Tecent Weibo. Our dataset includes captured users’ interests and potential interesting items, descriptions and classifications on users.

The dataset includes many logging document, the adopted logging document and detailed introduction are as follows:

1. Training dataset:

*format:(UserId)\t(ItemId)\t(Result)\t(Unix-timestamp)*

Result values are 1 or -1, where 1 represents the user UserId accepts the recommendation of item ItemId and follows it (i.e., adds it to his/her social network), and -1 represents the user rejects the recommended item.

1. User Profile

*Format:(UserId)\t(Year-of-birth)\t(Gender)\t(Number-of-tweet)\t(Tag-Ids)*

Each line contains the following information of a user: the year of birth, the gender, the number of tweets and the tag-Ids.

Gender has an integer value of 0, 1, or 2, which represents “unknown”, “male”, or “female”, respectively. Number-of-tweet is an integer that represents the amount of tweets the user has posted. Tags are selected by users to represent their interests. If a user likes mountain climbing and swimming, he/she may select "mountain climbing" or "swimming" to be his/her tag. There are some users who select nothing. The original tags in natural languages are not used here, each unique tag is encoded as an unique integer. Tag-Ids are in the form “tag-id1;tag-id2;...;tag-idN”. If a user doesn’t have tags, Tag-Ids will be "0".

1. Item Information

*Format: (ItemId)\t(Item-Category)\t(Item-Keyword)*

Item-Category is a string “a.b.c.d”, where the categories in the hierarchy are delimited by the character “.”, ordered in top-down fashion (i.e., category ‘a’ is a parent category of ‘b’, and category ‘b’ is a parent category of ‘c’, and so on.

Item-Keyword contains the keywords extracted from the corresponding Weibo profile of the person, organization, or group. The format is a string “id1;id2;…;idN”, where each unique keyword is encoded as an unique integer such that no real term is revealed.

1. User action

*Format:(UserId)\t(Action-Destination-UserId)\t(Number-of-at-action)\t(Number-of-retweet )\t(Number-of-comment)*

This document contains the statistics about the ‘at’ (@) actions between the users in a certain number of recent days. If user A wants to notify another user about his/her tweet/retweet/comment, he/she would use an ‘at’ (@) action to notify the other user.

For example, user A has retweeted user B 5 times, has “at” B 3 times, and has commented user B 6 times, then there is one line “A   B     3     5     6”

1. User Key Word

*Format: (UserId)\t(Keywords)*

Keywords is in the form “kw1:weight1;kw2:weight2;…kw3:weight3”.

Keywords are extracted from the tweet/retweet/comment of a user, and can be used as features to better represent the user in your prediction model. The greater the weight, the more interested the user is with regards to the keyword.

Every keyword is encoded as a unique integer, and the keywords of the users are from the same vocabulary as the Item-Keyword.

1. User following History

*Format: (Follower-userid)\t(Followee-userid)*

The file user\_sns.txt contains each user’s follow history (i.e., the history of following another user). Note that the following relationship can be reciprocal.

To note, training data does not contain any useful information for training mode(UserId and ItemID can not provide useful information based on naive analysis), that means feature set have to be devised from scratch. However, we are provided by multi-dimensional data to build feature set for training data, which can create a lot of useful feature information for training data.

Another observation is that Training set is huge, which contains more than 73209277 in training data. In order to use Weka or train locally, after discussion with Professor Rose, I decide to use part of the data to build models.

Also, the training set have very imbalanced result(most item are negative-unfollow), which have to be preprocessed, detailed procedure is introduced in Section 5.

1. Procedure Outline

The goal of the project is to train a recommendation model-- given a userId and itemId and relating feature set, the model can predict whether this user will accept the item. This section outlines the whole process. Section 5 describes how to do the data preparation, which includes two parts: a> select balanced training set from huge dataset; b> extract features from other files. Section 6 introduces the baseline and its performance. In section 7, optimization strategies are applied to models, it focuses on selecting features and comparing models. Section 8 delivers an analysis based on previous steps.

1. Data Preparation and Feature Extraction
   1. Data Preparation

The original dataset contains 73209277 items, which is time-consuming to process in weka or run locally. After discussion with professor, I decide to select part of the project to evaluate the data.

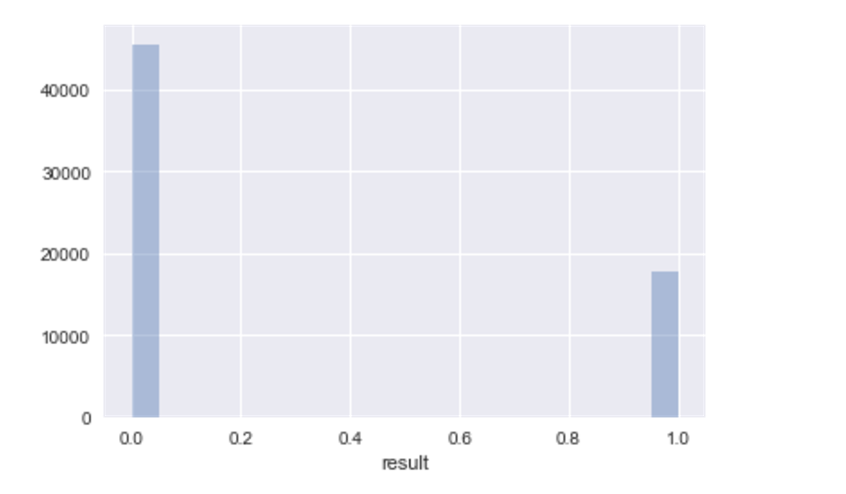
In the baseline, I select 0.1% of the totally data to do the training set, which is almost 10000 items. I just select the first 10000 items in order.

Also, after observing the raw training set, the dataset is imbalanced. Specifically, there are much more negative data than positive data. In the baseline training set, the ratio from positive data / negative data is 10.1. That is reasonable for normal life because most recommendation is abandoned, only a small number of recommended items will be followed.

In the optimization, I expand the training data into 63,263 lines. In order to balance the selected training set, I go through 200,000 items in raw data, then select part first 30000 lines of raw data and select just positive data in the following lines. After applying this strategy, the ratio from negative data to positive data has decreased less than 3, the graph is as follows:

Another problem is the sparcity of training data. For example, there are many users in the training set, some of them might just contain 1-2 recommendation record. It will decrease the model performance. In order to contain less users in a large training set, I use a set to record all the users with positive record, and just include users with positive record in the dataset. Note, even for user with following record, there are still many unfollowing records from the user, thus the unfollowing records are still more than following records, but the imbalanced condition can be eased.

After applying above two strategies, the training set statistics for positive and negative items are as follows.



Graph1. balanced data set

In the above graph, 0 represents negative(unfollow) record, 1 represents positive(follow) record.

5.2 Feature Extraction

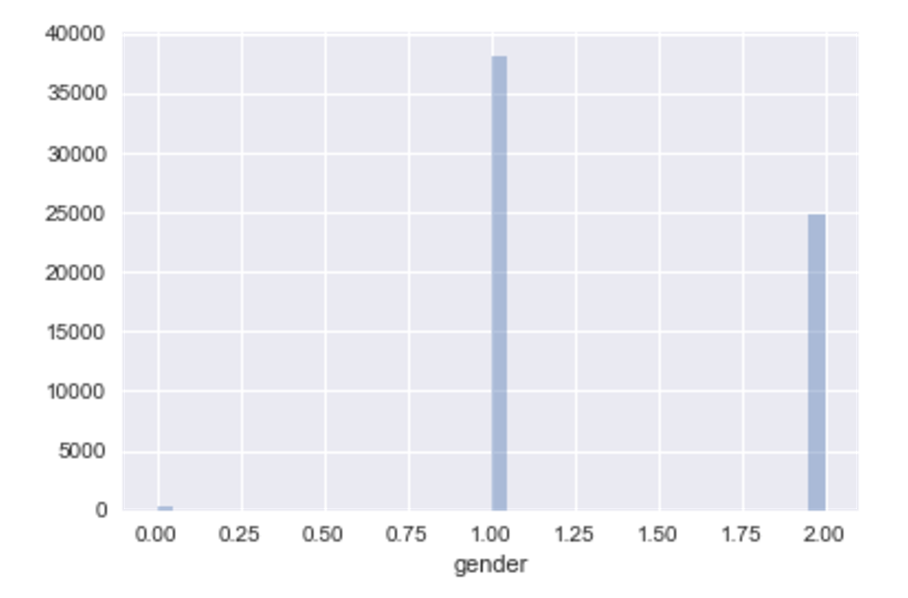
In order to get most out of the data for predictive model, devising accurate and contributive features are of great importance.

Following the strategy from Jason Brownlee[2], four steps are adopted: <a>brainstorm for features; <b>devise features; <c>select features; <d>evaluate models.

According to existing documents, there are two parts of data are contributive for prediction based on background knowledge: item information and user information.

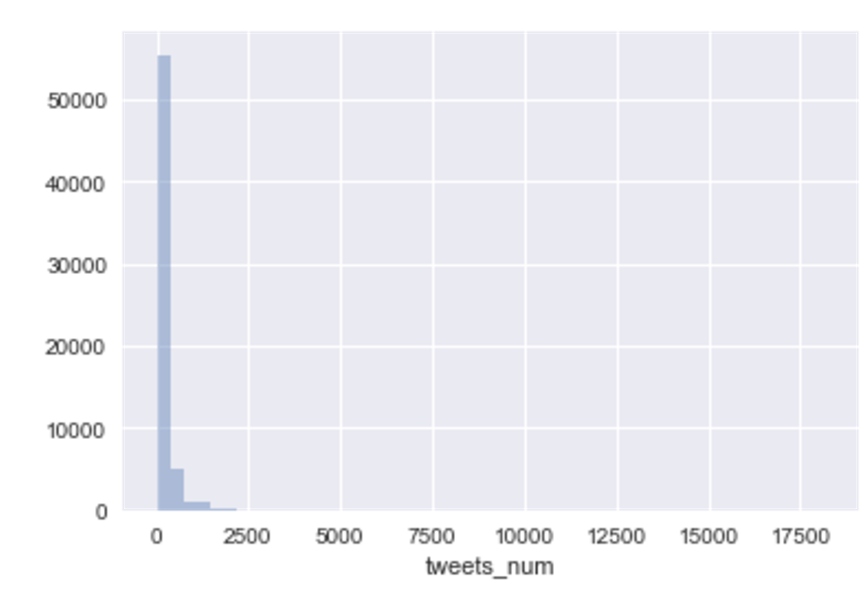
In user profile, we get the information of gender, birth year, number of tweets and tag Ids. Users background will exponentially influence interests and preferences. For example, males and females spend time on different items; young people and old people like different areas. Number of tweets represent activity of the user, an active user might have higher probability to accept recommendation, vice versa.

Firstly, I plot the feature of gender, birth year and tweets number distribution on the graphs.



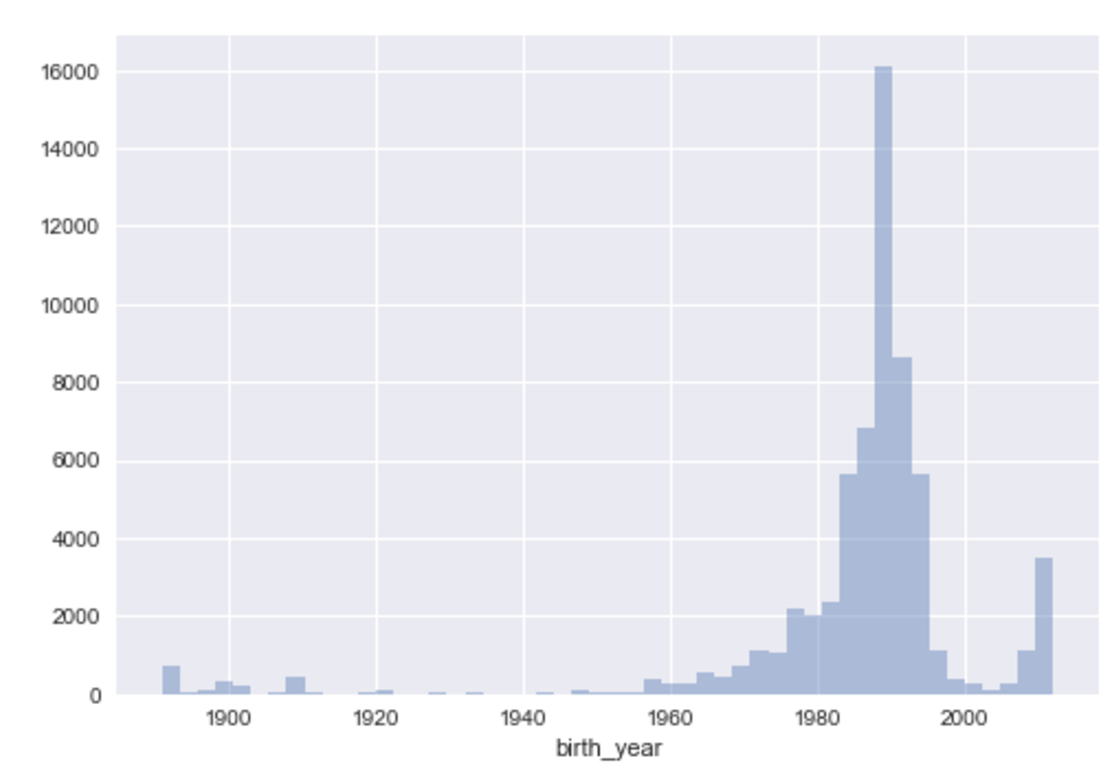
Graph2 gender distribution in training set

0, 1 and 2 represents “unknwon”, “male” and “female”. The ratio from male to female is nearly 2.52, which is acceptable for training set.



Graph 3. Tweets number distribution in training set

In the above graph, the x-axis represents tweets number, y-axis represent how many users have that number of tweets number. The number of use is decreased with the increase of tweets number. After calculation, the median of tweets number is 5. Thus user with less than 5 tweets are regarded as inactive user, user with more than 5 tweets are divided into active user group. The feature is named “tweets\_num\_high” in the training process. “True” and “False” represent “tweets number higher than 5” and “tweets number lower than 5”.



Graph 4. Birth year distribution in training set

In the above graph, the birth year distributes as Gaussian distribution. The median of birth year is 1990, thus we divide people into “born before 1990” and “born after 1990”. The binarization aims to classify group by age, this threshold divides group into “young” and “old”. The feature is named as “birth year late” in training process. “True” and “False” represents “born after 1990” and “born before 1990”.

For item information, tags have been obtained from the raw data. In evaluate whether user has interest in current item, adding the feature “common keywords”, if the user have common tags with the item, the feature is “True”, otherwise, the feature is “False”.

Original training set contains feature of itemId, UserId and timestamp.

After above feature construction, the feature set is as follows:

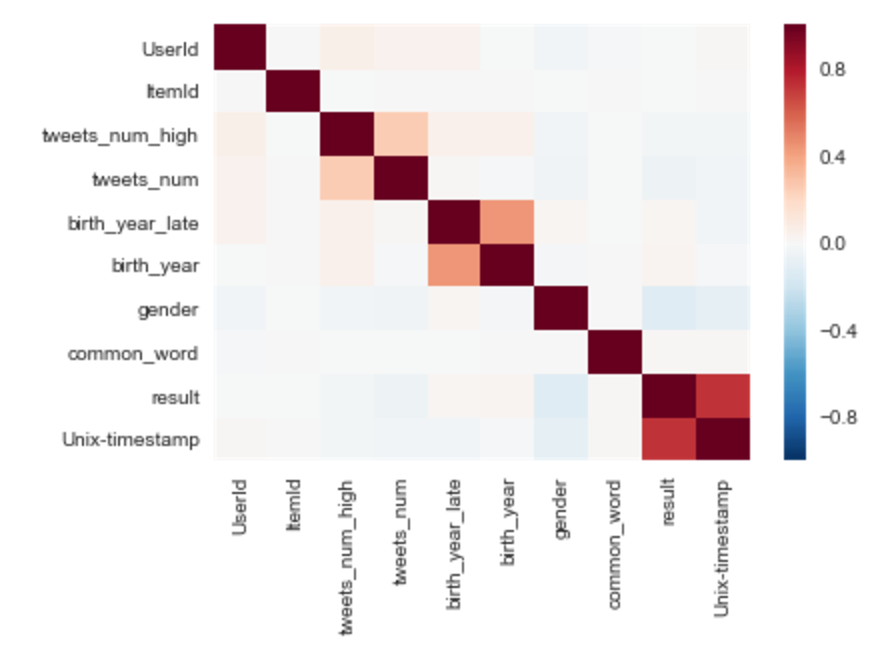
UserId/ ItemId/ tweets\_num\_high/ tweets\_num/ birth\_year\_late/ birth\_year/ gender/ result/ Unix-timestamp

Form1: Feature set example



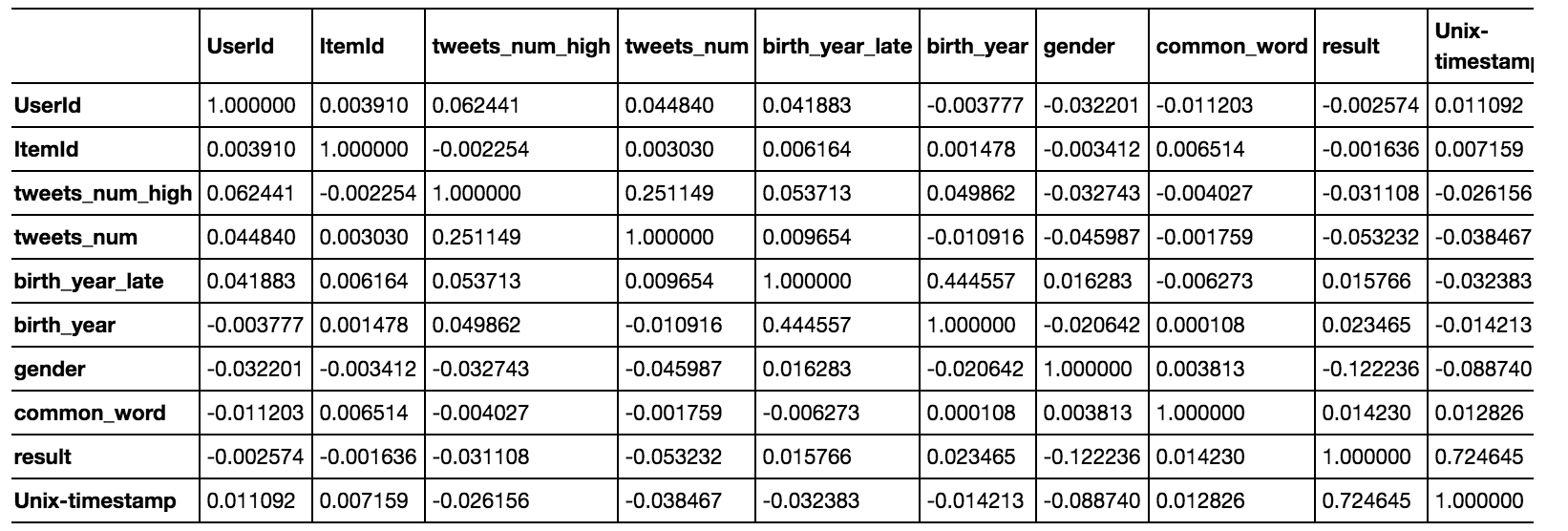
Irrelevant features to the problem need to be removed.  We use different importance scorings and background information to “view” the importance of each feature.

Correlation coefficients are used to “view” the importance for each feature.



Graph 5. Feature correlation relations

Form2. Feature correlation coefficient



From above correlation form and background knowledge, “userId”, “itemId” and “common\_word” have very weak correlation with final result, they are just dropped out of the feature set. “Gender” and “Unix-timestamp” have higher correlation with “result”, which should be kept to evaluate the result. For feature with birth year and tweets number, which can be reevaluated by adding them to the final model.

Form 3. Accuracy when select different feature set

|  |  |
| --- | --- |
| Feature set combinations(all with timestamp and gender) | Performance(Accuracy in Logistic Regression) |
| Birth\_year, birth\_year\_late, tweets\_num, tweets\_num\_high | 0.9222 |
| tweets\_num, tweets\_num\_high | 0.9222 |
| Birth\_year, birth\_year\_late |  |
| Birth\_year, birth\_year\_late, tweets\_num |  |
| Birth\_year, birth\_year\_late, tweets\_num\_high |  |
| Birth\_year, tweets\_num, tweets\_num\_high | 0.9222 |
| birth\_year\_late, tweets\_num, tweets\_num\_high | 0.9222 |

Based on comparison on different feature set, the combination with all features can achieve the best performance.

Thus selected feature set is as follows:

Form 6. Selected Feature Set

|  |  |  |
| --- | --- | --- |
| Selected Feature Set Introduction | | |
| Feature | Type | Description |
| gender | Nominal | Mark the gender for user |
| Unix-timestamp | Numetric | Recommendation time |
| Tweets\_num | Numetric | Posted tweets number of user in recent timeline |
| Tweets\_num\_high | Nominal | Posted tweets number higher than median in recent timeline |
| Birth\_year | Numetric | Birth year of user |
| Birth\_year\_late | Nominal | Birth year late than current user |

1. Baseline Performance

In the baseline, we choose 0.1% of full dataset to do cross-validation.

Lada has examined that gender influences personality and preferences. Thus, we just add user gender into feature set. After applying logistic regression algorithm, the accuracy reaches 51%. This step confirms the feasibility the methodology.

The next step is to apply different models on the full feature set

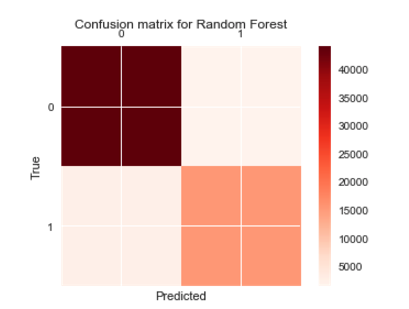
1. Model Selection

I performed different algorithms on existing training set. Use cross validation to calculate the accuracy.

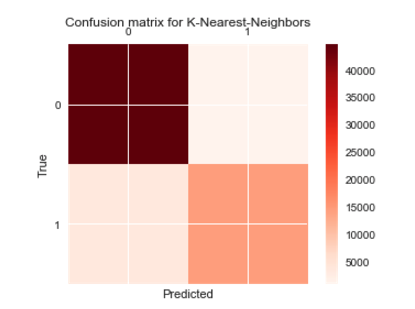
I selected user

Form 8: Performance under different algorithms

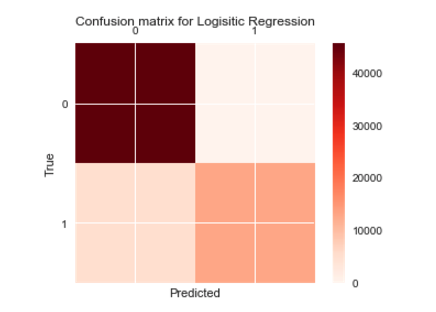
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Random forest | K nearest neighbors | Logistic regression | SVM |
| Accuracy | 0.9382 | 0.9329 | 0.9223 | 0.9146 |
| Precision | 0.9662 | 0.9808 | 1.0 | 0.9998 |
| Recall | 0.9488 | 0.9297 | 0.9025 | 0.8939 |
| kappa | 0.8448 | 0.8263 | 0.7899 | 0.7670 |



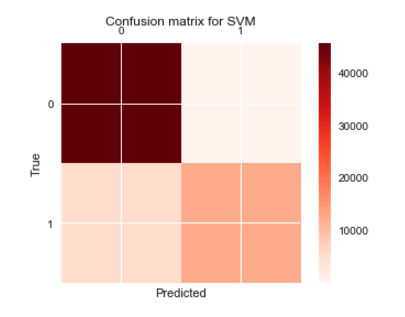
Graph 6. Confusion matrix using random forest



Graph 7. Confusion matric using K-nearest neighbors



Graph 8. Confusion matrix using logistic regression



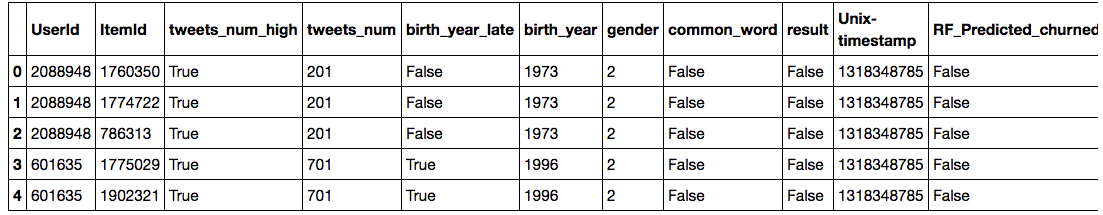
Graph 9. Confusion matrix using SVM

Confusion matrix can be viewed to compare different models. The color depth of each area represents number of positive or negative training data are classified as positive or negative. Viewed from above graph, confusion matrix from random forest have the shallowest color of wrong classified data, other algorithms have more positive data being classified as negative.

Based on the performance, random forest classifier achieves the best performance. This is because many features are mostly numeric and can help divide data into different branch accurately. The features influence the effectiveness of different models. SVM and logistic regression both belong to linear regression, it will only out perform when the function is truly linear. What is more, regularization is applied before building the model, which effectively avoids over-fitting when training random forest model.

Here is part of prediction results for some training items.

Form 9. Prediction result



Maybe because the training data still have the imbalance problem- much more negative data than positive data. That leads to many positive data are still being predicted as negative.

1. Optimization

Parameter tuning is an important step for model optimization.

Here we do parameter tuning on SVM, logistic regression and KNN.

I use the “grid search” function installed in sklearn python package(which is similar to weka tuning process, brutly try many parameter combinations).

I firstly tune parameters of random forest. Form 10 list important parameters to tune in random forest algorithm.

Form 10. Selected Parameters of Random Forest Algorithm

|  |  |
| --- | --- |
| Parameters | Meaning |
| N\_estimators | The number of trees in the forest |
| Max\_features | The number of features to consider when looking for the best aplit(set as “auto”) |
| Max\_depth | An integer represents maximum depth of the tree. |
| Min\_samples\_split | The minimum number of samples required to split an internal node. |
| Min\_samples\_leaf | The minimum number of samples required to be at a leaf node. |

In the tuning process, “N\_estimators” and “Max\_depth” are selected to tune.

Random forest is a ensemble method which average over many trees. Trees number determine how many bagging this algorithm will use. In general, more trees lead to better results; however, too much trees will make training process time consuming. “Max\_depth” determines the depth of the tree. Short tree will avoid over-fitting but might not give concise classification. Deep tree might have the risk of over-fitting but can give more concise classification.

In the tuning process, “n\_estimators” are set as 10, 20, 25, 30, 40, 60, 70; “max-depth” are set as 1, 5, 8, 9, 10.

The best tuning result is to set max\_depth as 1, n\_estimators as 25, which can achieves 0.795 as best score.

I then choose another non-linear algorithm-KNN, to tune parameters.

Form 11. Selected parameters of KNN

|  |  |
| --- | --- |
| Parameters | Meaning |
| N\_neighbors | Number of neighbors to use by default. |
| weights | Weight function used in prediction |
| metric | The distance metric use for the tree. |
| N\_jobs | The number of parallel jobs to run for neighbors search. |

I choose the “n\_neighbors” as the tuning parameter. The optimal number of neighbors usually depends on dataset features. Increasing n can reduce over-fitting but once n is too large will effect results in decreased variance, which lead to overall performance negatively.

After setting “n\_neighbors” as [3, 7, 10, 13, 20, 25, 30, 40, 50, 60, 70, 80, 100, 500, 1000], I found that 500 gives the best performance which gives the score as 0.874.

1. Final Result

Here we give a conclusion of the final result. This project aims to build a recommendation model based on raw data.

The project selects 63263 from original 73209277 Tecent Weibo raw data. After feature devise and evaluation, “user gender”, “birth year”, “young or old”, “tweets number in recent days”, “active or not based on tweets number” and “time of posting” have been selected into feature set.

When training the model, random forest, logistic regression, KNN and SVM have been tried. Random forest can achieve the best performance, the accuracy is 0.9382, kappa is 0.8348.

In the optimization, tuning process are applied on random forest and KNN algorithms. Fro random forest,

1. Discussion

This project follows the methodology taught in class and completes the basic classification task for existing items. Because of the data size, I write the python code to run locally, which also follows the weka’s exploration and training process.

The value of the work is to obtain an understanding that how the recommendation system can generate most efficient and personalized recommendation to each user. Moreover, the paper gives insights to how to do feature engineering based on complex logging, comparing different models and optimizes model by tuning parameters.

Another interesting finding is that, some features, look very important based on background knowledge, but achieves little contribution to final model. For example, we use common tags to predict whether user will have interest to the item, however, it confirms to have little correlation with result. One reason is that tags are very sparse information; another reason is that tags are more subjective compared with other features, which might influence the accuracy of the feature.

Final, the project is a class project, many further work still can be explored, for example, decompose the time stamp: maybe during certain time users have higher probability to accept the recommendations; many advanced algorithms can be tried[5][6][7]; training on large scale dataset can be applied using Amazon AWS or other cloud infrastructure, larger dataset decreases over-fitting of the training model.

Building a robust and efficient recommendation system will be an interesting industrial topic in the future.

References

[1] Tencent and K. Cup, "KDD cup 2012, track 1,". [Online]. Available:

https://www.kddcup2012.org/c/kddcup2012-track1. Accessed: Feb. 14, 2017 <https://www.kaggle.com/c/kddcup2012-track1#description>

[2]http://machinelearningmastery.com/discover-feature-engineering-how-to-engineer-features-and-how-to-get-good-at-it/

[3] <http://www.sciencedirect.com/science/article/pii/S0898122112003161>

[4] <http://www.ojphi.org/ojs/index.php/fm/article/view/1057/977>

[5] Walter F E, Battiston S, Schweitzer F. A model of a trust-based recommendation system on a social network[J]. Autonomous Agents and Multi-Agent Systems, 2008, 16(1): 57-74.

[6] Debnath S, Ganguly N, Mitra P. Feature weighting in content based recommendation system using social network analysis[C]//Proceedings of the 17th international conference on World Wide Web. ACM, 2008: 1041-1042.

[7] Pham M C, Cao Y, Klamma R, et al. A clustering approach for collaborative filtering recommendation using social network analysis[J]. J. UCS, 2011, 17(4): 583-604.