# On predicting net-increase followers of an WeChat official account

**Bin (Xiaobin) Li | Fall 2019**

## Abstract

In this project, I used online traffic data of an official account on WeChat, one of the biggest social media platforms in China, to predict the net-increase of follower count so as to have a better understanding of users' behavior.

## Introduction

I have been running a podcast, YinengFM, since May 2015. The goal of YinengFM is to share designers' stories and celebrate the value of design. We invite professional designers, design educators and students as guests on our podcast to share their experiences with Chinese-speaking designers globally.

Each week, 1-2 episodes will be released and posted on the official WeChat account of Yineng FM. Over the years, we have collected a lot of data on WeChat. As of now, we have more than 26,000 followers, and the number is still growing. To gain a better insight into the listeners' behavior, I planned to analyze the traffic data and predict the net-increase of follower count.

After a lot of research at the beginning, we chose podcasting as the medium to build connections with our audience. Chris Evans (2008) [1] has found that "podcasting can fill an important needs gap by allowing learners to continue the learning activities when it might not normally be possible." Through this effective, convenient and intimate medium, our audience could learn about design in a more flexible way.

On the other hand, WeChat was on the rise and poised to become the biggest social media platform in China. Studies have shown that people, especially the younger generation, spend more and more time on WeChat. "Almost all of the undergraduate students use WeChat every day, 84% of them use it more than half an hour per day," Mao (2014) [2] concluded. Since our target audience is college students who are interested in design, WeChat is the perfect platform for us to connect with them.

## Data Preparation

All data used in this project come from WeChat backend. There are 1584 instances in the dataset, each of them representing the traffic on that day. Table 1 is the feature list.

| Name | Type | Explanation |
|---|---|---|
| Traffic data | | |
| total_view_people | Numeric | Total number of people who visited our account. |
| total_view_times | Numeric | Total number of visitings we get on that day |
| from_official_account_people | Numeric | People reached our account through direct link. |
| from_official_account_times | Numeric | Traffic coming from direct link |
| from_post_people | Numeric | Number of people access from others' post sharing |
| from_post_times | Numeric | Traffic from others' post sharing |
| sharing_people | Numeric | How many people share our posts |
| sharing_times | Numeric | How many times our posts being shared on that day |
| people_save | Numeric | How many people save our posts |
| times_saved | Numeric | How many times our posts being saved |
| Follower data | | |
| new_followers | Numeric | How many new followers we got on that day |
| unfollowers | Numeric | How many people cancel their subscription on that day |

Table 1: Raw data from WeChat

I prepared the data before I dived into building the model. Here are the changes I made:

1. Changed date to weekdays: We release our episodes every Tuesday, therefore, weekday features will be more helpful.

2. Combine "new_followers" and "unfollowers" to one feature: The number of net-increase followers is more valuable to us.

3. Apply min-max normalization to every numeric attribute.

Table 2 is the new dataset with adjusted feature space.

| Name | Type | Explanation |
|---|---|---|
| Traffic data | | |
| weekday=Mon | Binary | If the instance came from that weekday, it would be 1, otherwise it would be 0. |
| weekday=Tue | Binary | |
| weekday=Wed | Binary | |
| weekday=Thy | Binary | |
| weekday=Fri | Binary | |
| weekday=Sat | Binary | |
| weekday=Sun | Binary | |
| total_view_people | Numeric | Total number of people who visited our account. |
| total_view_times | Numeric | Total number of visitings we get on that day |
| from_official_account_people | Numeric | People reached our account through direct link. |
| from_official_account_times | Numeric | Traffic coming from direct link |
| from_post_people | Numeric | Number of people access from others' post sharing |
| from_post_times | Numeric | Traffic from others' post sharing |
| sharing_people | Numeric | How many people share our posts |
| sharing_times | Numeric | How many times our posts being shared on that day |
| people_save | Numeric | How many people save our posts |
| times_saved | Numeric | How many times our posts being saved |
| Follower data | | |
| net-increase-in-followerrs | Numeric | New follower - Unfollwers |

Table2: Adjusted dataset

After preparing the raw data, I split it into three parts, Cross-Validation data(70%), Development data(20%), and Test data(10%). The Cross-Validation data was used to build models, the Development data to do error analysis to expand feature space, and Test Data to test the performance of the final model.

### Baseline Experiment

I ran the linear regression algorithm with default settings on the dataset to get the baseline performance (Table3).

| | |
|---|---|
| Correlation coefficient | 0.2841 |
| Mean absolute error | 0.0148 |
| Root mean squared error | 0.0376 |
| Relative absolute error | 81.0229% |
| Root relative squared error | 95.931% |

Table 3: Baseline performance

### Data Exploration — K Means Experiment

To make more sense of the data, I conducted the K means experiment. First of all, I discretized the data by dividing instances evenly into four classes based on the class value, "net-increase-in-followers." Each category has the same number of instances. From the smallest number to the largest, I labeled them as "very low," "low," "medium," and "high."

I found that weekday was the most critical feature that influenced clustering. For example, if an instance came from Wednesday, the model tended to cluster it into the "high" class. This may be because we usually release new episodes on Tuesdays, and most listeners would tune in the next morning.

The centroid of these clusters also verified this assumption. For example, the class "very high" was assigned to the cluster with a centroid that has "weekday=Wed" value. Meanwhile, "very low" was assigned to the cluster with a centroid that has "weekday=Friday" value.

### Feature Engineering — Adding Average Feature

With the discretized dataset, I expanded the feature space by doing an error analysis on LightSide.

First of all, I got my baseline classifier model with SVM on LightSide. Table 4 below shows the baseline performance.

Baseline Model Metrics:

| Metric | Value |
|---|---|
| Accuracy | 0.4221 |
| Kappa | 0.2076 |

Baseline Confusion Matrix:

| Act \ Pred | high | low | medium | very low |
|---|---|---|---|---|
| high | 135 | 1 | 90 | 29 |
| low | 14 | 2 | 117 | 105 |
| medium | 71 | 5 | 179 | 91 |
| very low | 12 | 2 | 105 | 153 |

Table 4: SVM baseline performance

I found no instance predicted as a "Low" class. After going through instances in "Medium" and "Low" classes, I realized that they mostly came from Mondays, Wednesdays, and Thursdays. However, the average values of all other features in the “Low” class are much smaller than those of the “Medium” class. Therefore, I speculated that the Weekday features may have confused the model.

Despite that, other features seemed to be able to differentiate these four classes. For one, the average value of most instances in the "Low" class was smaller than that of the "Medium" class. To emphasize this difference to the algorithm, I added to the dataset a new feature, Average (Table 5), which would contain the average value of all the numeric values from attributes. As a result, the model could make more accurate predictions by paying less attention to the Weekday features.

| Name | Type | Explanation |
|---|---|---|
| Average | Numeric | Contains the average value of all numeric values of an instance. |

Table 5: New average feature

The model with new feature space was proven to have significant improvement! (Table 6) However, it still had trouble predicting instances in the “Low” class. As a result, I kept analyzing the “Low” class instances that were predicted as “Medium” class.

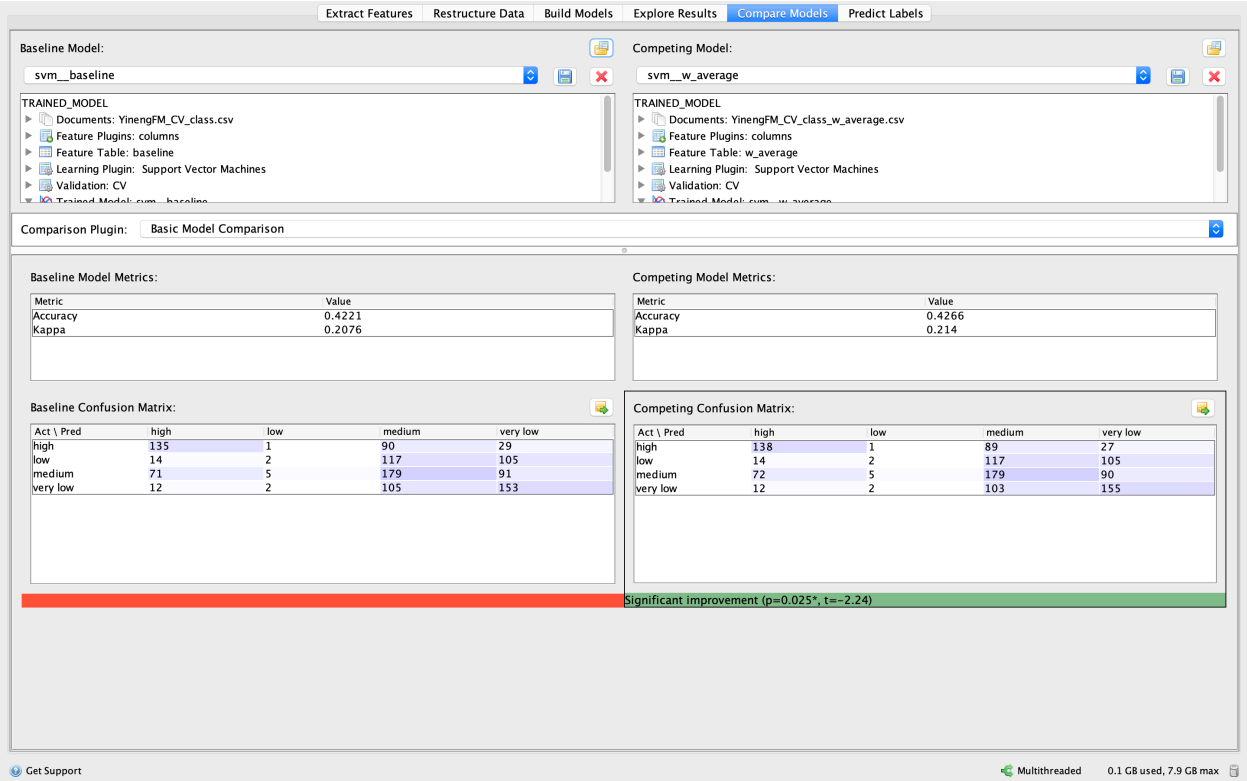


Table 6: Significant improvement after adding the Average feature.

## Feature Engineering — Features Showing Outside Traffic

As I continued comparing the incorrectly predicted “Low” class instances to those from “Medium” class, I found that the more total traffic there was in a day, the more it tended to come from outside the official account (Table 7). Only current followers can access to the official account; therefore, this outside traffic may result from being visited by those who were following our account. It is safe to conclude that these are the people who are interested in us and most likely to become our new followers.


Total View Times for each Class Prediction broken down by Class. Color shows average of total_view-account_times. The data is filtered on Exclusions (Class,Class Prediction,Sharing Times), which keeps 179 members. The view is filtered on Class and Class Prediction. The Class filter keeps low and medium. The Class Prediction filter keeps medium.

Table 7: The comparison between the incorrectly predicted "Low" class instances to those from "Medium" class. The red color indicated the number of outside traffic: the deeper the color is, the higher outside traffic is.

I believe that adding a feature that represents this outside traffic would help the model make more accurate predictions. The new feature, total_view-account, contains the value calculated by subtracting inside traffic from total traffic (total_view_times - from_official_account_times). Following this, I added another feature — total_people-account, which contains the value calculated by subtracting inside visitors from the total number of visitors (total_view_people - from_official_account_people) (Table 8).

| Name | Type | Explanation |
| --- | --- | --- |
| total_view-account | Numeric | Subtracting inside traffic from total traffic (total_view_times - from_official_account_times) |
| total_people-account | Numeric | subtracting inside visitors from total number of visitors (total_view_people - from_official_account_people) |

Table 8: New features

With the new features, there was a significant improvement comparing to the SVM baseline model. (Table 9)

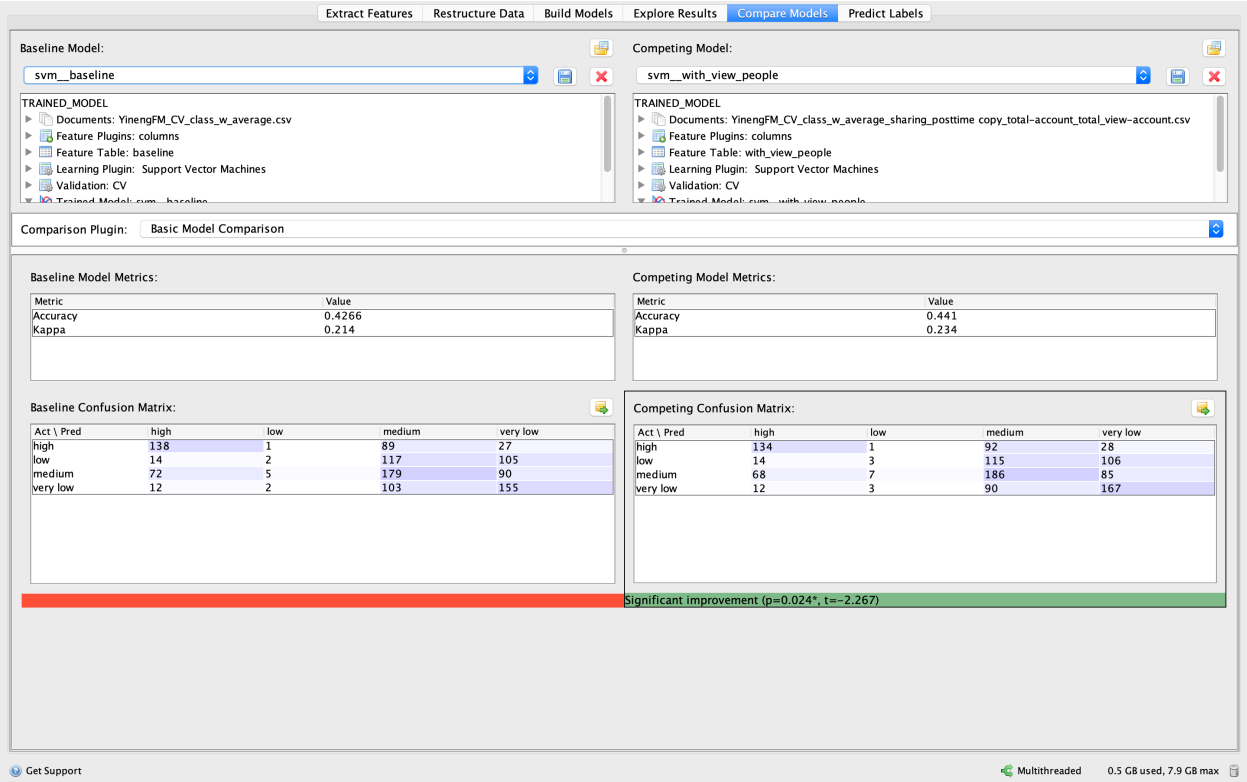


Table 9: Comparison between baseline model and the new model with added features

## Tuning

After feature engineering, I started algorithm tuning. I chose the SVM regression algorithm and tuned within the range of 1-3 the exponent setting of the poly kernel function. Table 10 shows the performance of the three different models in terms of

Root_relative_squared_error. As the default setting had the best performance, I chose it to run the final model.

| | With exponent=1 (default) | With exponent=2 | With exponent=3 |
|---|---|---|---|
| YinengFM_CV_final | 77.68 | 78.48 | 106.11 |

Table 10: Tuning result in terms of Root_relative_squared_error with different exponents of the SVM regression model Final evaluation

**Final Evaluation**

Table 11 indicates that the performance of the Testing dataset slightly outperformed that of the baseline model.

| | |
|---|---|
| Correlation coefficient | 0.3009 |
| Mean absolute error | 0.0143 |
| Root mean squared error | 0.0379 |
| Relative absolute error | 78.6965% |
| Root relative squared error | 96.6811% |

Table 11: Performance on testing dataset

**Discussion**

From this project, I have learned how to do error analysis and expand feature space step by step. I have also realized that Machine Learning is not a panacea, and only good data and meaningful features can lead to good results. Furthermore, knowing how big data generates values will benefit my career in the future.

**Reference**

[1] Chris Evans 2008, The effectiveness of m-learning in the form of podcast revision lectures in higher education. Computers & Education, 50 (2008), pp. 491-498. https://doi.org/10.1016/j.compedu.2007.09.016

[2] Mao, C. 2014,  Friends and Relaxation: Key Factors of Undergraduate Students' WeChat Using, Creative Education, 5, 636-640. http://dx.doi.org/10.4236/ce.2014.58075