Predicting Brand Advertisement Consumption on Facebook by Model Comparison

Zekai Chen

George Washington University

Shuqi Zhu

John Hopkins University

Reza Djavanshir

John Hopkins University

Abstract

This study presents a research approach by comparing machine learning models for predicting the lifetime consumption of posts published in brands' Facebook pages. It turns out the XGBoost model performs the best among all the single models and the ensemble one. Also, feature analysis is done to understand how each of the seven input features influenced the response (category, page total likes, type, month, hour, weekday, paid). The page total likes was considered the most relevant feature for the model, with the largest importance. Moreover, categories "Category" and "Type" are consistently important, which is consistent with the empirical study.

Keywords: lifetime consumption, machine learning, XGBoost, ensemble model, feature importance

1. Introduction

In recent years, social media has become an important advertising medium for businesses. As with any type of advertising, businesses need to understand what determines their advertisings impact to maximize the benefit they derive from such content. In other words, understanding what makes advertising

successful aides businesses in creating more effective advertising. Furthermore, establishing a predictive model with data mining approach for the success of advertising would allow a business to assess planned content quantitatively *(Moro et al., 2016)*. Beyond just aiding general advertisement design, this permits for fine-tuning of content.

It is well-known that one of the most important phases in the procedure of data mining is modeling. However, most studies focused on accessing the information provided by the model in terms of how input features affect the response while put less efforts on studying how to choose the most suitable models for specific cases or different data sets. A predictive model with a higher predicting accuracy will naturally provide more reasonable and convincing interpretations for the relationship between the predictors and the response. Thus, managers responsible for advertising could have a better idea on the receptiveness of the published posts and make more accurate decisions on modifying the strategies to attract more product consumptions.

A nature way to think about selecting the most appropriate models would be doing the model comparison. By comparing the performance of each "candidate" on the same data set with the same metric, we can easily and intuitively identify which are the best ones.

In this paper, we focused on finding out the most appropriate models to predict one of the available performance metrics of posts published on worldwide cosmetic companies' pages in Facebook. This data set contains nineteen features of five hundred posts of a worldwide renowned cosmetic brand on its Facebook homepage, including twelve performance metrics and seven features known before publication. All the posts were published between the 1^{st} of January and the 31^{th} of December of 2014 *(Moro et al., 2016)*.

Though the data set has more than nine performance metrics of posts, this paper will only discuss the impact of performance as Lifetime Post Consumptions, a variable which measures how many times anyone interacts with a given social media post, with some exemptions (see variable description). Therefore, this data set will be used as the experimental subject in this paper.

Table 1 includes the features known before publication used as predictors. The page total likes is the only numeric variable measuring the performance of the companys homepage. The remaining features are categorical variables.

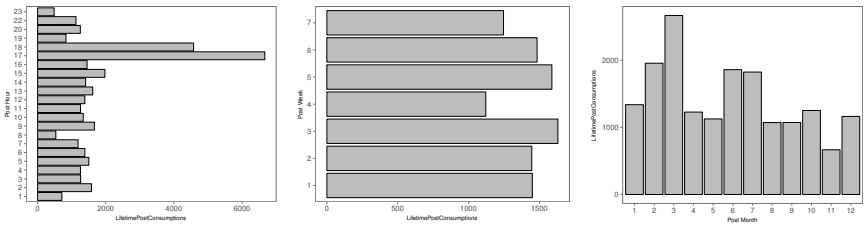
Feature	Description
Category	Manual content characterization: action (special offers and contests), product (direct advertisement, explicit brand content), and inspiration (non-explicit brand related content).
Page total likes	Number of people who have liked the company's page.
Type	Type of content (Link, Photo, Status, Video).
Post month	Month the post was published (January, February, March, $\cdots$, December).
Post hour	Hour the post was published (0, 1, 2, 3, 4, $\cdots$, 23).
Post weekday	Weekday the post was published (Sunday, Monday, $\cdots$, Saturday).
Paid	If the company paid to Facebook for advertising (yes, no).
Lifetime post consumptions	The number of clicks anywhere in a post.

Table 1: Features and description

2. Exploratory analysis

Once we know what the dataset is like, we need to do some exploratory data analysis to get a better understanding of the relationship between all the predictors and the response.

Here, for categorical variables, we choose to visualize the comparison in a histogram, each category stands for one column. For the one quantitative variable, we choose to visualize the relationship between predictor and the response in points with a smoothing curve which could indicate a trend behind the data.



(a) Influence of "Hour" on "Lifetime Post Consumptions."

(b) Influence of "Week" on "Lifetime Post Consumptions."

(c) Influence of "Month" on "Lifetime Post Consumptions."

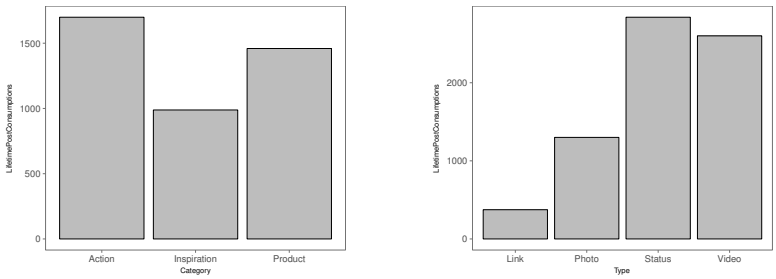
Figure 1: "Hour", "Week" and "Month" possible impact on the prediction

We have noticed from above that during the months from 2 to 3 and months from 6 to 7, the "Life time post consumptions" would get higher. This might indicate that spring and summer break period would get more consumptions. In addition, we can easily find that as page total likes number increases, the overall "Life time post consumptions" would go down in a long trend. More importantly, we can find that different hours and weeks could result in different consumptions. During the time range from 5pm to 6pm, the lifetime posts consumptions would get the highest value while other time periods perform without significant difference. Furthermore, it seems Wednesday and Friday have more consumptions while Thursday and Sunday have a lower value when compared with other days among a week.

From Fig.2b below, we can see that 'Status' type posts have the most LPC(stands for lifetime posts comsumption), then comes the 'vedio' type. We can also obviously see the trend between the 'Category' and the LPC.

It is also shown in Fig.3b that paid posts would get more consumptions as always assumed, which is also verified by the graphics above. As a result, all

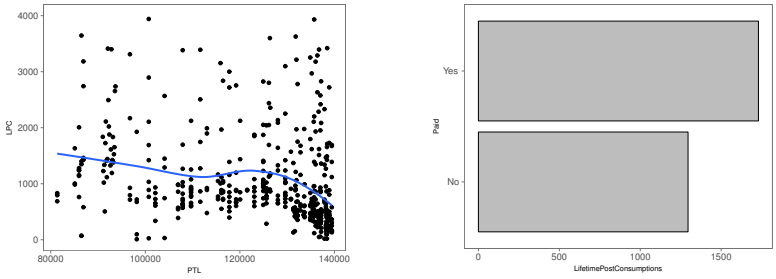
of these predictors might have various kinds of significant relationship with the response, though more analysis should be done to help us find a more clear pattern behind this dataset.



(a) Influence of "Category" on "Lifetime Post Consumptions."

(b) Influence of "Type" on "Lifetime Post Consumptions."

Figure 2: "Type" and "Category" impact on output



(a) Influence of "Page total likes" on "Lifetime Post Consumptions."

(b) Influence of "Paid" on "Lifetime Post Consumptions."

Figure 3: "Page total likes" and "Paid" impact on the output

Additionally, we have found from Fig.4 that Page Total Likes is strongly related to Post Month. This makes sense, as page likes are effectively cumulative if we assume the number of individuals un-liking pages is negligible. However, this relationship is even stronger than you might expect; more than 99% of the variance in Page Total Likes is explained by Post Month. Even though, it is likely more wise to use both two predictors in any model because the exact

relationship between them and the response is still not clear.

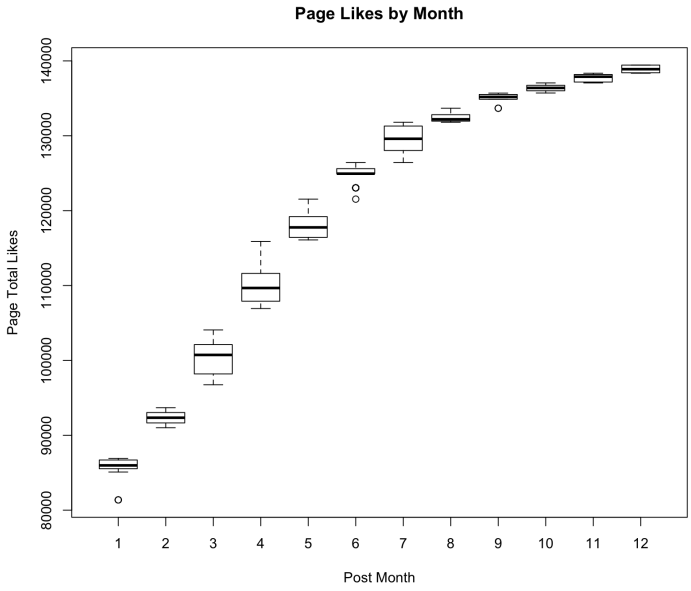


Figure 4: Page total likes by month

3. Methodology

From the exploratory analysis, we can see that all the variables are of some significant relationship with the response. Thus, we will choose all the variables as predictors for our models later. In this section, we briefly review the methodology of each model we will use next.

3.1. K-nearest neighbors regression

Here we consider one of the simplest and best-known non-parametric methods, *K-nearest neighbors regression* (KNN regression) *(Bac et al., 2016)*. The KNN regression method is closely related to the KNN classifier. Given a value for K and a prediction point x_0, KNN regression first identifies the K training observations that are closest to x_0, represented by N_0. It then estimates $f(x_0)$

using the average of all the training responses in N_0. In other words,

$$\hat{f}(x_0) = \frac{1}{K} \sum_{x_i \in N_0} y_i \tag{1}$$

We will use the cross validation to find the optimal value of K.

3.2. Supporter vector machine with radial basis function kernel

Here we use the radial basis SVM and minimize the upper bound of the training error, since it has the smallest MSE *(Chang et al., 2011)*. That is: Given the value of ϵ, we find $f(x)$ that has at most ϵ deviance from the real response y_i for all the training data, keeping the flatness of $f(x)$. Hence, with $f(x) = exp[\lambda(\beta - x)^2] + b$, it tries to

$$\text{minimize} \ \ \frac{1}{2} \|\beta\|^2 + C \sum_{i=1}^{n} (\xi_i + \xi_{i^*}) \tag{2}$$

$$\text{subject to} \begin{cases} y_i - f(x) - b & \leq \epsilon + \xi_i \\ f(x) + b - y_i & \leq \epsilon + \xi_{i^*} \end{cases} \text{for all i} \tag{3}$$

C and λ is automatically given by R by cross-validation, then ϵ is specified by cross validation with given C, λ and the tolerance ξ is the default value. Also, we will do cross-validation to pick the best parameter.

3.3. Random forest

Random forests provide an improvement over bagged trees by way of a small tweak that reduces correlation between the trees *(Leo et al., 2001)*. As in bagging, we build a number of decision trees on bootstrapped training samples. But when building these decision trees, each time a split in a tree is considered, a random sample of m predictors is chosen as split candidates from the full set of p predictors. The split is allowed to use only one of those m predictors. A fresh sample of m predictors is taken at each split, and typically we choose $m \approx \sqrt{p}$. That is, the number of predictors considered at each split is approximately equal to the square root of the total number of predictors *(James et al., 2013)*.

For here, we will tune the number of variables randomly sampled as candidates at each split and we will apply the best model to do the simulations.

3.4. Extreme Gradient Boosting

XGBoost is short for "Extreme Gradient Boosting", where the term "Gradient Boosting" is proposed in the paper Greedy Function Approximation: A Gradient Boosting Machine, by Friedman. XGBoost is based on this original model *(Chen et al., 2016)*. This model follows a common loss plus regularization pattern as,

$$\text{Objective} : \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k} \Omega(f_k), \text{ for } f_k \in \mathcal{F}$$

and replace the original loss function with a Taylor expansion approximation. Instead of spending excessive effort in minimizing the objective function, this model enhances itself by growing refined trees.

3.5. Neuron networks with one hidden-layer

Neural networks typically consist of multiple layers or a cube design, and the signal path traverses from the first (input), to the last (output) layer of neural units *(Goodfellow et al., 2017)*. Here, we only add one hidden layer for the NN model in case of over-fitting because the dataset size is not very large.

In fact, for our real data, the input size should be seven, and the output size should be one. We set four hidden units in the hidden layer. We also choose the rectified linear function as the activation function and we use mean square error(MSE) as our loss function. Last but not least, the optimization method is stochastic gradient descent which is very powerful when processing the regression problem.

3.6. Ensemble Models

We use ensemble modeling *(Dietterich et al., 2000)* to combine multiple of our previous models to get our final model. Essentially, given models $\hat{y}_1, \hat{y}_2, \ldots \hat{y}_k$, we determine the optimal weights $w_0, w_1, w_2, \ldots w_k$ for the combined predictor

$$\hat{y} = w_0 + \sum_{i} w_i \hat{y}_i.$$

Notably, we determined the optimal weights in two manners. The first we call the averaging method. We assumed that the weights in the ensemble add to

one; they represent a weighted average (the constant w_0 is zero). Then we search over all possible weights (allowing them to be negative, etc.) to find the one that minimizes the training root mean squared error. This works surprisingly well since the error curve is quadratic for a two-component model.

The second method we call the linear regression method. We consider the predictions of each component model to be predictors in a linear regression. Fitting this leads to weights, including a constant term w_0.

4. Simulation and results

In this simulation section, we will firstly separate the whole set into train and test sets with a ratio of 80%. Then we will train all the quick machine learning models with 10-fold cross validation to get the best parameters. Finally we determine the performance of the models on test set and use the results together to pick the best single model with the metric as root mean square error.

4.1. Model comparison

After a 10-fold cross validation, each model will have ten samples with its own prediction. We try to visualize them in a box plot and do the model comparison with verified and reliable data analysis.

Fig.5 has obviously told us the models performance on training set. XGBoost have the comparatively lowest median root mean square error while it is with a comparatively larger range of variability. Neural network has a high RMSE and also with large variability. Also, random forests has very stable median root mean square error.

In order to find out which is the best single model, we also plot the predicted value versus real value graph, from which we can easily see that XGBoost and Random forests do much better than other algorithms.

4.2. Model performance on test set

For our simulation on the test set, the prediction results are shown as following:

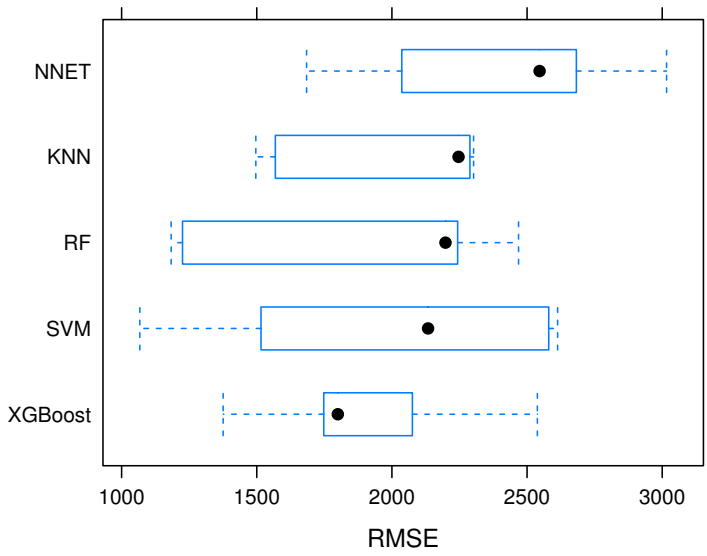


Figure 5: Model comparison on the training set

	KNN	SVM	RandomForest	XGBoost	NNET
RMSE	1923.84	2097.02	1865.28	1803.66	2493.82

Table 2: Model performance on test set

From this table, we find that XGBoost also has the best performance on test set with the lowest RMSE.

4.3. Ensemble model

We tried a few different ensemble models. The first involves combining two models, one capturing variance in the categorical predictors and the other in the single continuous predictor. The premise is twofold: first that some models work better with continuous predictors than categorical (and vice-versa), and second that the two models should capture different aspects of the data in using different variables, leading them to ensemble well. In a way, the idea is similar to that in random forests (albeit, without the random selection).

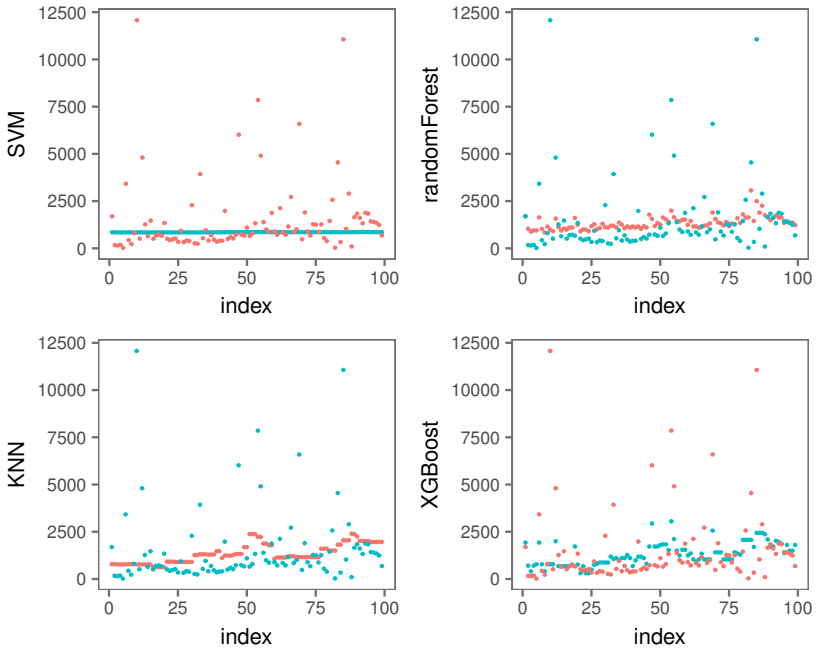


Figure 6: Predicted value "vs" true value

We ensembled a regression tree (using 'Category' and 'Type') with a linear model (using 'Page total likes'). It should be noted that we initially used a random forest model and polynomial regression model; switching from a random forest model to a regression tree substantially improved fit, and switching from a polynomial regression to a linear regression reduced model complexity with little loss in performance.

We fit this using the averaging method. That is, we found weights w_1, w_2 with $w_2 = 1 - w_1$ minimizing the training RMSE. In particular, you can see in Fig.7 that the appropriate weights are $w_1 = 0.8$ and $w_2 = 0.2$.The result is close to the tree model, and only performs slightly better in terms of RMSE. Note this is evaluated only on training data, as the results are clearly suboptimal.

Second, we combined "SVM with radial kernel", "XGBoost" and "Regression tree" together to create an ensemble model. For the fitting technique, we use a linear regression model in this case to combine predictions.

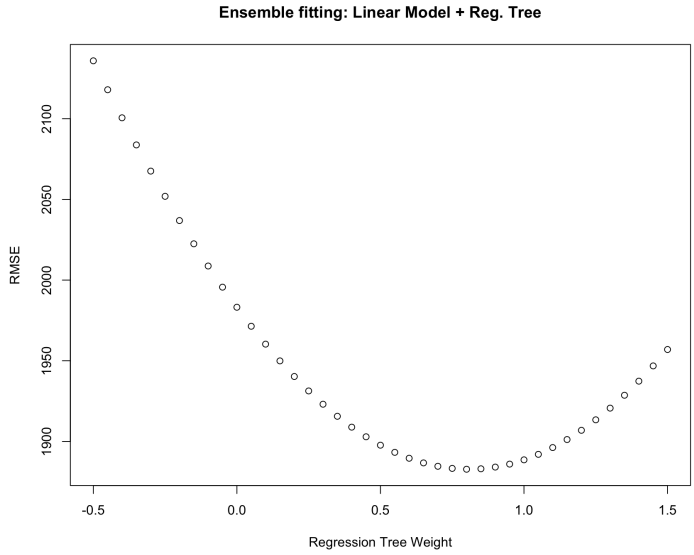


Figure 7: Searching for optimal weights in the first ensemble model.

The mean test RMSE values are then as follows:

	KNN	SVM	RandomForest	XGBoost	NNET	Ensemble model
RMSE	1923.84	2097.02	1865.28	1803.66	2493.82	1828.42

Table 3: Ensemble model performance on test set

Also, when we add the ensemble model into our comparison graph, we will see that,

From this gram, we can easily see that this ensemble model has the smallest variability and even lowest median RMSE. Combining model performance on both training and test sets, we can recognize the ensemble model as the best one.

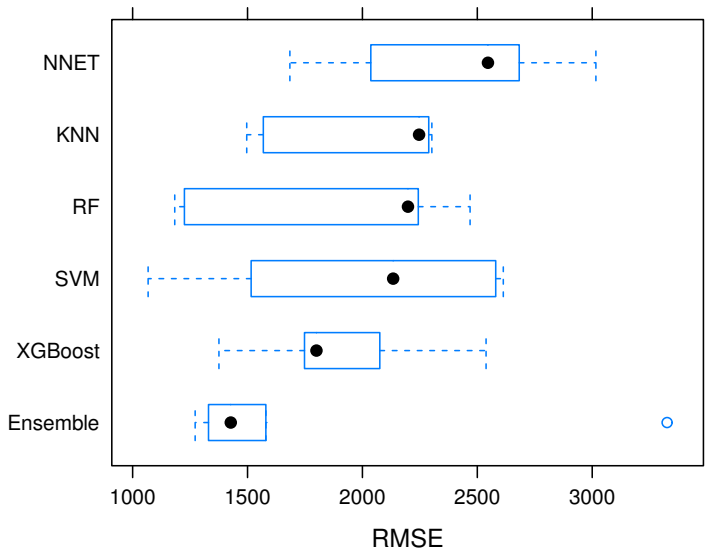


Figure 8: Models comparison including ensemble model

5. Feature analysis and Interpretation

In this section, we will do some feature analysis to find any valuable information behind this database. Fig.9 is a graph which shows the importance of each predictor in the data set. The importance stands for the total amount of residual square is decreased every time when boosting trees split on each given predictor *(Trevor et al., 2001)*.

Obviously, from Fig.9, this variable "Page total likes" is the most important variate among all predictors, then comes the Type and Category. More precisely, Status posts and Action kind posts are seemed more attractive to consumers which can also be verified by the explorotary data analysis in our report.

As a result, it would be much better for business to design posts which are more attractive to consumers. More importantly, Status and action posts tend to result in more consumptions than other types or categories.Furthermore, note that a number of post month categories are in the top ten, reflecting their

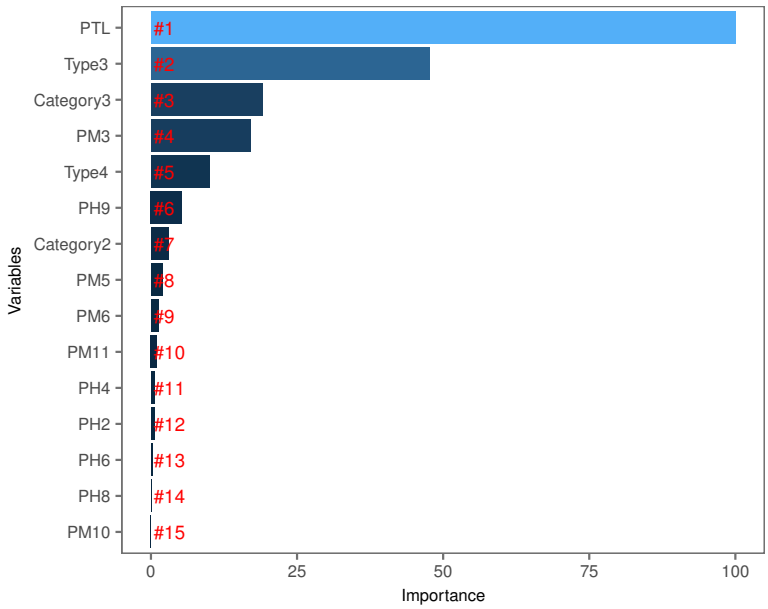


Figure 9: Importance of each feature from XGBoost

relationship to page total likes.

Note that these results are relatively consistent with our other models. Certainly, we found in fitting our regression tree models that category and type were consistently the most predictive of the categorical variables.

6. Discussion

6.1. Model Selection

In our analysis, it became apparent that simple regression trees were more appropriate than random forests for this problem. This is likely the case for two reasons. The first is that there are not very many variables, which reduces the primary benefit of using random forests in the first place: that it is robust against signal dilution, which is the idea that if one source of variance (i.e. one signal) is dominant in the data, the effect of the others on the fit model will be substantially reduced, regardless of their size. Random forests avoids this

by fitting tree models to subsets of the predictions, and thus can pick up each signal separately.

The second reason that random forests does poorly is the sheer variance in the regression tree models. Results can vary from having 50% or more variance explained by the model, and having close to non explained. The precise reason for this is not clear, but it is clear that aggregating many such models should produce a poor result.

Another issue we identified was with fitting regression models to our continuous predictor. As noted in our exploratory analysis, the co-distribution of 'Page total likes' and 'Lifetime post consumptions' is quite unusual (see Fig.3). In particular, there are many data points with high page likes and low post consumptions, while the rest of the data is sparsely distributed around the space (very roughly). This means we both fail to have homoscedasticity, required for linear regression, but furthermore have potential for more exotic issues with our data analysis. Admittedly, this is purely speculative, as we have not particular issues in mind. Nevertheless, it suggests that fitting a model to just the continuous predictor was not ideal, since more sophisticated methods were needed than linear or polynomial regression; in particular, these sophisticated methods tend to work better with more than one variable (e.g. SVM).

In ensemble modeling, we used two methods to fit our models, one based on the idea of averaging component models, and the other on the idea of fitting a linear regression with component model predictions as inputs. These two methods appeared to produce very different results. The linear regression method gave large coefficient estimates, and a particularly large intercept coefficient, but this was impossible for the averaging method. This reflects that the linear regression method is more flexible, but it did not seem to benefit substantially from this, suggesting that the averaging method is better. Admittedly, neither ensemble technique did very well, so it is hard to use the results for comparison.

6.2. Interpretation of Training Error

Early in the analysis, we were bothered by the apparently large training RMSE that each of our models exhibited. However, upon further inspection we realized two things. The first is that this is relative to our response variance, and in fact this is only roughly half the variance in the response (which we consider acceptable). The second is that it is not apparent what proportion of this error is irreducible or not, as we inherently are limited in estimating the irreducible error (we only know it is not more than total error in any case). Therefore, it is only reasonable to use this error as a comparison between models, and not as an absolute measure of our success in modeling.

7. Conclusions

In conclusion, our best model is the ensemble model with XGBoost, SVM and Regression tree and the XGBoost model among single models. From Fig.8, you can see that typically it does best with an RMSE of around 1400.

In terms of features, this study showed that the categories 'Category' and 'Type' are consistently important, measured relative to noise, which is consistent with the conclusion provided by *(Moro et al., 2016)*. To be more precisely, a "status type" of posts will be much more likely to result in a big success than other types. Also, special offers and contests are obviously much more attractive to consumptions than any other "post category". The factor Page Total Likes also appears important, but its relation to the response is more complicated, and the evidence for its significance less compelling.

More importantly, this research presents a general approach for choosing machine learning models when applied to building predictive systems especially for advertising through social media, which is doing model comparison. A model able to provide a higher accuracy prediction will own more value and reflect the information inside more explicitly. Though the conclusions from XGBoost is very similar to the ones from Support Vector Machine, it can not prove this process is dispensable at any time.

As noted before, the particular distributions of the predictors (e.g. page total likes) may cause issues with modeling. Therefore, we might consider subsampling the data to get a different (perhaps uniform) distribution such predictors in order to avoid such problems.

In inspecting feature importance, we noted that many categorical predictors had disparities among the importance of their categories. For example, out of the post hours, only posts at 5 pm seemed to do significantly different than at other hours. It is possible that splitting these predictors into separate variables before model training might improve performance.

References

[1] S. Moro, P. Rita and B. Vala, "Predicting social media performance metrics and evaluation of the impact on brand building: A data mining approach", Journal of Business Research, Volume 69, Pages 3341-3351, 2016

[2] G. James, D. Witten, T. Hastie and R. Tibshirani, Introduction to Statistical Leaning with Applications in R, Springer-Verlag, New York, 2013

[3] T. G. Dietterich, "Ensemble methods in machine learning", Multiple Classifier Systems Lecture Notes in Computer Science, Pages 1-15, 2000

[4] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system", Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Pages 785-794, 2016

[5] C.-C. Chang and C.-J. Lin, "Libsvm: A library for support vector machines", ACM Transactions on Intelligent Systems and Technology, Volume 2, Issue 3, 2011

[6] H. Trevor, T. Robert and F. Jerome, The Elements of Statistical Learning, Springer New York Inc., New York, 2001

[7] N. Bac, M. Carlos and B. B. De, "Large-scale distance metric learning for k-nearest neighbors regression", Neurocomputing, Volume 214, Pages 805-814, 2016

[8] B. Leo, "Random forests", Machine Learning, Volume 45, Pages 5-32, 2001

[9] G. Ian, B. Yoshua and C. Aaron, Deep Learning, MIT Press, 2016

Author Profile

Zekai Chen earned his bachelor degree at Shanghai University with major in Applied Mathematics in 2016. He is currently a master candidate with major in Statistics at the George Washington University, Washington, D.C.

Shuqi Zhu earned her master degree at Johns Hopkins University with double major in Enterprise Risk Management and Information System in 2017. Currently, she is a data analyst at Baker Botts LLP.

Reza Djavanshir, Doctor of Science in System Engineering and Engineering Management, joined the Johns Hopkins Carey Business School in 2002. He is an Associate Professor in the practice track with expertise in the areas of Global Sourcing and Supply Chains and System Integration Strategies.