Project Part 1 for MS&E 226, Foundamentals of Data Science, Winter 2022-23

Exploring BlogFeedback Dataset

Abstract Our goal is to understand the BlogFeedback Data Set from the UCI Machine Learning repository[1] to help set us up for the next parts of the project.

Authors Andrew Hong, Annie Zhu

1 Dataset Description

Our dataset is a collection of over 500,000 blogposts in processed HTML files posted online from 2010-2012 from UCI's Machine Learning Repository. It includes several different useful covariates: the previous number of comments at varying points before a basetime (the basetime is the arbitrary time and date used as the marker for when we start predicting the number of comments 24 hours later), day of the original post, 200 frequent bag of words used in the post text, length of the post, number of parent pages (blogs linked from the given blog) and their comment statistics, and lengths of time between its post date and the basetime. The intention behind this dataset is to build models that predict the number of new comments a given post will receive in the following 24 hours after the basetime.

The author of this data set does not provide any more information on its collection process than this, which raises some questions about its rigor. For instance, was the data scraped from a few popular blog sites, or all over the internet? The (unknown) answer to that question has implications for what population(s) can this dataset and any associated data models be generalized to.

Another issue is about the completeness of the data set was that there were no column names. That meant anytime we were going to refer to a particular column in the dataset we would have to use the index number. Therefore, we created code to add column headers based on the description of the attributes online.

2 Train - Test Data Split

Our data set was already been divided into the training and test sets for us by the creator. After downloading the data folder from the repository, we had one "training set" cvs file with 52,397 rows as well as sixty different "testing set" cvs files with 7,624 rows in total. However, this setup leads to around a a 87% / 13% split between the training and testing data, rather than the 80% / 20% split typically used.

Because the smaller testing ratio could effect our confidence in the final error calculations, we decided to ignore a random portion of the training set (21,901 rows) so that we could formulate a 80% / 20% split from the data. The reason why we chose to throw out that portion of training data rather than repurpose it for testing data was because the training data was exclusively from 2010 to 2011 and the testing from 2012. We did not want to mess with the time series.

3 Why This Dataset and Potential Predictions

We chose this dataset because an analysis of it offers answers about human social networks, linguistics, and media studies. It also has practical applications to social media companies who want to more efficiently manage their content and bloggers who want to increase their content interaction. We were also excited about this dataset because of meaningful applications in activism and journalism. For instance, how can activists use insights about timing of posts and linkages to ensure crucial messages are engage more people?

Because this dataset only focuses on blog posts specifically, a pitfall is that our findings may be limited to blog posts. However, our analysis could still inspire deeper investigations into similar datasets across different media like journalism articles.

Potential Application. By measuring how different variables predict with number of comments in the future, our dataset could inform what leads to more content interactions, which could inform content creators and platforms on how to best increase their content interaction efficiently.

A pitfall for using our dataset to predict what leads to more content interaction is that we do not identify the sentiment of comments. It could be that what leads to more interaction is actually negative, not positive, comments. Our dataset has no way to discern this, thus may inadvertently lead to content strategies that incur hate comments.

Potential Binary Predictions. There are two potential binary predictions that stand out: 1) We could set a **threshold** and predict whether the number of comments in the next 24 hours is above or below that threshold. 2) Or we could have the outcome variable represent whether the number of comments in the next 24 hours is an increase or decrease from the number of comments from the 24 hours right beforehand. This version shed light on the direction of the **trend**.

The **trend** option as a binary response variable would likely be harder to model than the **threshold** option because that is an easy adaption of the continuous response variable. Our data set is also limited because it does not track one blog post over many days so there would be a rather small number of reference points for change over time.

4 Ideal Changes

Building off of the pitfalls, it would have been ideal to have data on on the number of comments per day for a week leading up to the basetime. Currently, the dataset only has the number of comments for two days before the basetime.

We also believe that data about the author of each post (i.e. the number of followers or friends they have) would have been useful. The number of comments a post will receive in the future could correlate with its author.

Additionally, we wish the dataset could have included the number of views, shares, likes, or dislikes a post has. This would help us gauge sentiment and non-comment content engagement.

5 Selected Outcome Variables

Here we will discuss our selected continuous and a binary outcome variable for the prediction task.

5.1 Continuous Variable

Our continuous outcome variable is the number of comments received in the 24 hours after the basetime. We chose this variable firstly because it was the dataset's creator's intention, but we also found personal interest because it may reveal correlations between specific variables and the resulting content interaction a post will receive immediately after an arbitrary time of interest. Measuring this variable has applications for content creators motivated to boost their short-term 24-hour content interaction and platforms to anticipate interaction surges for specific posts in short, 24-hour increments.

5.2 Binary Variable

Our binary variable is measure of directional trend: does the number of comments increase or decrease be-

tween the 24 hours before and after the basetime? We determined that this option is favorable because it has interesting applications in understanding change over time of online activity. As more social media companies are trying to predict what will become viral on their platforms for harm reduction purposes, being able to identify the overall trend is increasingly important.[2] In addition, this dataset has the capabilities of predicting this variable because there is ample data about the popularity of the site the blogpost is on and the trend over two days before the basetime.

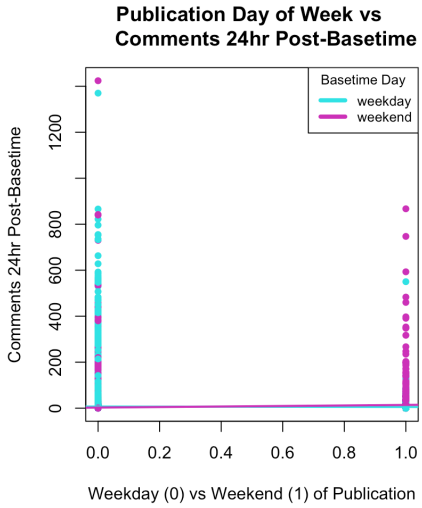
6 Engineered Features

We engineered a feature binary variable to measure whether a blog post (row) had any of the 200 most frequently used bag of words. This helped us gauge the significant of prevalence of bag of 200 words on the number of new comments 24 hours after the basetime. We also engineered a feature continuous variable to measure the sum of a blog post's (row) prevalence of any or all of the 200 most frequently used bag of words.

In addition, we engineered two features pulling from the 14 days of week binary variables for both the publication day and the basetime day. Those 2 new variables are binary variables of whether the publication date and basetime was on a weekend.

However, we found no correlation between either basetime or publication day being on a weekend or not compared to number of comments 24 hours post-basetime: both r-squared values-and that of their interaction regression-were less than 0.01.

In the graphic of the interaction regression we interestingly see visually when the basetime was on a weekend, there were more high-comment posts that had publication dates on a weekend; whereas when the basetime was on a weekday, there were more high-comment posts when the publication date was also a weekday. The correlation value is so low (0.003), however, that the outliers we see visually are not statistically significant when taking the entire population of blog posts. However these outliers may shed light on future analyses on these high-comment outliers and what best predicts the success of very-high-comment blog posts, so it's a useful insight in that perspective.



7 Irrelevant Columns

There were multiple instances of NA values in the binary columns dedicated to measuring whether each of the 200 most frequent words used in the blog post. Because those NA values do not represent instances where a given word was used, we imputed 0's in place of each of those values so to run complete regression analyses and summaries of the 200-frequent-word data.

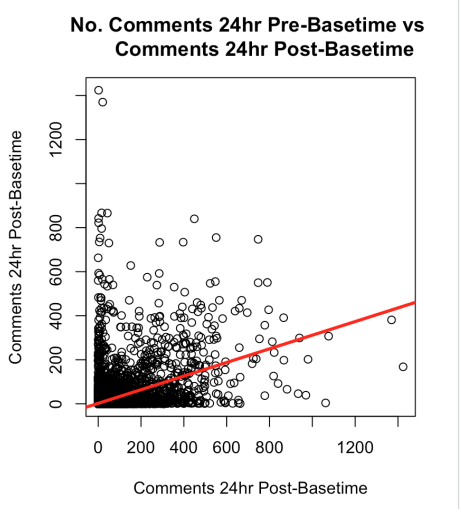
A regression comparing the binary instance of any of the 200 frequent bag of words being used in a post to the number of comments received 24 hours after the basetime did, indeed, have a very low r-squared value of 0.002. Similarly a regression comparing the continuous sum of any of the 200 frequent bag of words in a blog post compared to the number of comments 24 hours post-basetime found a low 0.006 r-squared value. Thus there is no obvious relationship between these 200 bag-

of-words variables and our outcome variables related to comments post-basetime, so they're most likely irrelevant to researching what predicts comments in the future.

8 Mutual Correlations

8.1 For Continuous Outcome Variable

The greatest positive correlation we could find to predict the outcome variable of number of comments 24 hours after the basetime was the number of comments 24 hours before the basetime. This has an r-squared value of 0.213.



8.2 For Binary Outcome Variable

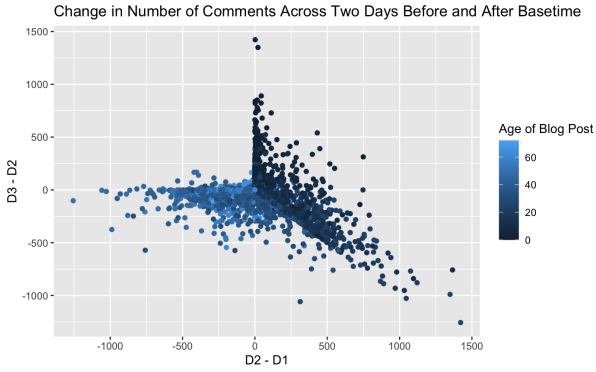
Let's narrow in on three covariates:

· D1: The number of comments posted between 48 hours and 24 hours before the basetime.

· D2: The number of comments posted 24 hours before the basetime.

· D3: The number of comments posted 24 hours before the basetime.

So, [D1, D2, D3] happened in sequential order. We can examine the relationship between the change between D1 and D2 with the change between D2 and D3.



This graph suggests that there is some nuanced correlation between D2 - D1 and D3 - D2. Almost all the [D3 - D2] values that are above zero correspond to a [D2 - D1] value that is above zero as well. Similarly, almost all the [D2 - D1] values that are below zero correspond to [D3 - D2] values that are below zero, too. These suggest that the direction of growth is correlated with previous direction of growth.

Additionally, there is an pattern happening with the time length factor, which can be seen with the color. Generally, the older posts have negative growth while the newer posts have positive growth, corresponding to how lighter points are more so below zero along both axes, while the darker points are more so above zero.

References

[1] Buza, K. (2014). Feedback Prediction for Blogs. In Data Analysis, Machine Learning and Knowledge Discovery (pp. 145-152). Springer International Publishing.

[2] Bak-Coleman, J.B., Kennedy, I., Wack, M. et al. Combining interventions to reduce the spread of viral misinformation. Nat Hum Behav 6, 1372-1380 (2022).

Project Part 2 for MS&E 226, Fundamentals of Data Science, Winter 2022-23

Predicting BlogFeedback Dataset

Authors Andrew Hong, Annie Zhu **TA Meeting** Met with Yueyang 2/2, 5:15pm and with Han Wu 1/30, 9:15am

1 Evaluation Strategy

One of our biggest challenges was determining our evaluation strategy since our data has some dependency on time and the train set didn't have timestamps so we couldn't know which rows overlapped in time. As a refresher, our dataset's intended goal is to predict the number of comments a blog will receive in the next 24hrs. Some covariates include the number of comments received in the previous 24-48hrs.

Hence, we couldn't use cross validation since the time series causes correlations across certain times, meaning K-fold CV wouldn't give an unbiased estimate of the generalization error. A similar issue arises with randomly sampling a validation set from the train set.

So we created the validation set from the "test set" we held out in Part 1. This method works because the dataset's author labeled the test sets with timestamps so we could ensure to assign the earlier dates to the validation set and the later dates to the real test set.

This was the only way we could confidently reduce bias in our estimated generalization error. This also helps prevent us from creating an overly optimistic estimate of test error. Moreover, we have enough data to confidently split the "test set" (which as 7624 rows) into a test and validation set. Still, our predicted error may be optimistic because we are selecting a model that performs best on one validation set.

To execute this method, we created a roughly 80-10-10 split for train (30496 rows), validate (3641), and test (3983), respectively. The train set was pulled from data recorded from 2010-11, the validate set from 2/2/12-3/1/12, and test set from 3/2/12-3/31/12.

2 Continuous Regression

We predicted the number of comments on a blogpost 24hrs after the basetime.

2.1 Baseline Model and Objective Function

Underestimating the raw number of future comments is worse than overestimating because a blog platform needs to allocate its bandwidth efficiently to posts with future high traffic. If a post gets higher than expected traffic, it could lead to more detrimental impacts like site-crashes or excess unmonitored spam comments. It's better to overprepare than underprepare. So we measured error in 2 ways. (1) unweighted root-mean-square error (RMSE). (2) Weighted root-mean-square error (w-RMSE) with an added 30% increase in error for positive residuals (underestimates). When selecting a best model strategy, we placed more emphasis on lower weighted errors, using unweighted-errors as tie-breaker.

The baseline was a simple OLS regression using all covariates.

2.2 Modeling Strategies

We opted for interpretable models instead of black box models for two reasons. (1) As we optimize our model, it's easier to pinpoint improvements if we can interpret how covariates impact the model. (2) It fosters data transparency for potential clients. We even created a baseline model with a transformation to center covariates so the intercept's interpretable as the predicted avg. no. comments on a blog for avg. values of each covariate: Intercept=6.493 (6-7 comments).

1) Baseline. This is our baseline of a linear model of all covariates equally. This had a w-RMSE of 27.83.

2) Lasso Regularization. Because we have 280 covariates, we wanted to prune insignificant covariates to center the most important ones. So we used a Lasso regression on the baseline with optimal lambda, but it didn't change much. w-RMSE: 27.78.

3) Higher Ordered Terms. We predicted the data of post engagement in the 24hrs before the basetime was more predictive of future comments, so we added higher order terms for those 12 covariates. w-RMSE: 27.83.

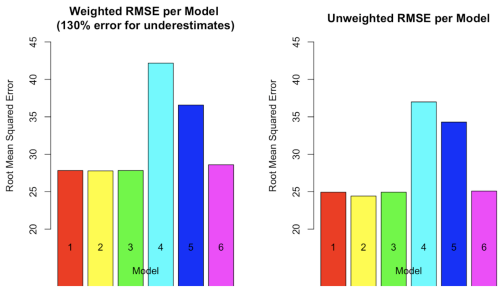
4) Logarithmic Transformation. We wondered if a model with logarithmic transformations would improve performance because the data is all positive (cannot be negative number of comments) and might be logarithmically proportional (% change) instead of linearly. So we did a logarithmic transformation on the same 12 covariates as Model 3. w-RMSE: 42.17.

5) Interaction Terms. We added 13 interactions of covariates that we saw in Part 1 better predicted our outcome, including interactions between parent pages, post activity preceding basetime, and covariates that represent categorical data like day of the week of basetime. w-RMSE: 34.30.

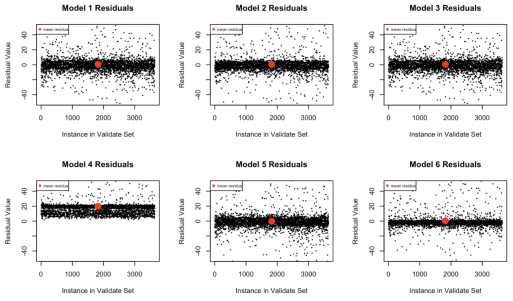
6) Lasso with Transformations. We wanted to add our higher ordered and interaction terms transformations to a Lasso regression to fully optimize covariates used across our added transformations, excluding logarithmic transformations that drastically worsened the model. w-RMSE: 28.59.

2.3 The Best Model and Its Estimated Error

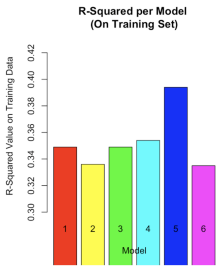
Model 2 had the lowest predicted w-RMSE and unweighted RMSE on the validation set. This suggests a simple model worked best with lasso regularization to optimally prune our 250+ covariates in this dataset.



Examining residuals, we see that every model has a relatively balanced number of positive and negative residuals except for Model 4, which skew positive (underestimates, what we're avoiding). Model 4 utilizes logarithmic transformations, implying the relationships within the data are more linear than logarithmic.



Bias-Variance Decomposition. Between variance and bias, Model 2 likely has a higher bias but lower variance because it fits the training data comparatively worse (2nd-least r-squared value) but has the smallest prediction error on the validation set. So while Model 2 has greater bias, its lower variance overcompensates to lead to the lowest prediction error model.



Conversely, Model 5 fit the training set comparatively well with the highest r-squared but had the 2nd-highest w-RMSE. This suggests Model 5's interaction terms overfitted the train set, so while its bias was lower, variance was comparatively much higher.

Overall, our baseline was already a good model that Lasso regularization slightly improved.

3 Binary Classification

The next subsections will be about binary classification: can we predict whether the rate of change in a blog's number of comments is increasing or decreasing?

3.1 Baseline Model and Objective Function

Our chosen objective function is to minimize 0-1 loss. The magnitude of the effect of a false positive (FP) and false negative (FN) are roughly equivalent. Directional trends serve more as long-term projections and meta-analysis of collective trends. As a result, the direction of the trend does not say anything about the exact magnitude of change. Thus, neither FN nor FP stands out as worse than the other so we aim to reduce the overall error with 0-1 loss.

The baseline model is to predict whichever label occurs most frequently in the data. This baseline makes sense because in practice people will either be tracking blog comment or not and there isn't much of a heuristic for it. The 0-1 loss for the baseline model is 0.255.

3.2 Modeling Strategies

The errors for all the strategies described are in the table at the end of this section.

A) Simple Logistic Regression. The first improvement is to build a logistic regression model using all the covariates.

B) Interaction Terms. We add 8 interaction terms between the covariates we believe make the biggest contributions. We selected covariates with p-values under 0.001 calculated from a t-test on all covariates, because that means they were statistically significant.

C) Logarithmic Transformations. We keep the interaction terms and add in logarithmic transformations.

D) Regularization. We also try out regularization in the form of lasso. We are motivated by the fact that our data set starts off with 281 columns and we add more with the transformations, so there seems to be a potential for too many covariates. However, using lasso generally appears to increase 0-1 loss. For example, the 0-1 loss became listed in the table is created with alpha as 0.01 and lambda as 0.5.

Here is a table comparing these different models.

	Base	A	B	C	D
0-1 Loss	0.255	0.119	0.105	0.111	0.194
Bias	$>$	$*$	$<$	$<$	$>$
Var	$<$	$*$	$>$	$>$	$<$

This table also offers some analysis on the bias-variance decomposition. In the table, $>$ means we think that the bias/variance of that model is greater than the bias/variance of model A. The symbol $<$ denotes the opposite. We leave $*$ for model A because that is what we are comparing the other models to. The base model and model D have higher bias (and lower variance) because they are simpler models so would systematically have higher error. Models B and C have higher variance (and lower bias) because they are more sensitive to the training data since they are more complex models.

3.3 The Best Model and It's Estimated Error

Our best model is the model with all covariates used and eight interaction terms added (this was model B from the previous section). Here was the breakdown of the performance:

		Actual	
		0	1
Prediction	0	612	63
	1	318	2648

Our estimated test error is 0.105. We expect the true generalization error from using the held out test set to be slightly higher. This is because we picked our model's covariates and transformations based on the validation set so we introduced bias into our error.

4 Appendix and Code for Best Models

4.1 Binary Prediction Code

For binary prediction, we have the code for model B: adding in interaction terms.

```
new_model <- glm(direction_of_growth
    ~ . + time_length*total_before_comments +
    time_length*before_24hr_comments +
    time_length*from_48hr_to_24hr_comments +
    time_length*first_24hr_comments +
    med_before_24hr_comments*before_24hr_comments +
    med_before_24hr_comments*from_48hr_to_24hr_comments +
    med_first_24hr_comments*before_24hr_comments+
    med_first_24hr_comments*from_48hr_to_24hr_comments,
    data=bin_data, family = "binomial")
```

where the dataframe bin_data is defined with the code below:

```
train_data <- read_csv("./trainingData.csv") %>%
    mutate(difference_in_ROG = (after_24hr_comments - before_24hr_comments)
    - change_between_two_days_comments) %>%
    mutate(direction_of_growth = ifelse(difference_in_ROG<0, 0, 1)) %>%
    select(-c(difference_in_ROG))

bin_data <- select(train_data, -c(after_24hr_comments,
    min_before_24hr_comments,
    min_before_24hr_links,
    min_first_24hr_comments,
    min_first_24hr_links,
    min_change_between_two_days_comments,
    change_between_two_days_comments,
    change_between_two_days_links))
bin_data$direction_of_growth = as.factor(bin_data$direction_of_growth)
```

Lastly, here is the code for the evaluation / test error prediction method:

```
evaluation <- function(model, data) {
  # Prediction
  p1<-predict(model, data, type = 'response')

  # Confusion Matrix
  pre1<-ifelse(p1 > 0.5, 1, 0)
  table<-table(Prediction = pre1, Actual = data$direction_of_growth)
  print(table)
```

```
  # getting the metrics
  TN = table[1]
  FP = table[2]
  FN = table[3]
  TP = table[4]
  accuracy = (TP + TN) / nrow(data)
  loss = 1 - accuracy
  TPR = TP / (TP + FN)
  FPR = FP / (FP + TN)

  print(paste("The 0-1 loss is", loss))
  print(paste("The accuracy is", accuracy))
  print(paste("The TPR is", TPR))
  print(paste("The FPR is", FPR))
}
```

4.2 Continuous Prediction Code

For continuous prediction we have the code of model 2: taking away some covariates bag of 200 words, day of week of basetime, and day of week of publication date

```
Xtrain2 = select(traindata, -c(after_24hr_comments)) %>% data.matrix()
Ytrain2 = traindata$after_24hr_comments
model2 <- glmnet(x = Xtrain2, y = Ytrain2, alpha = 1, lambda = bestLambda)
# lambda generated from modelling wRMSE's across dozens of different...
#... lambda values to select the lambda that yielded the lowest test...
#... error on lasso regression. See code to find best lambda below.
model2$dev.ratio #r2=0.3363457
```

And here is a version of the code we used to determine the best lambda value (using RMSE and w-RMSE:

```
# Generate the values of lambda
lambda_values <- rep(NA, 100)
for (i in 1:100) {
  lambda_values[i] <- i / 10
}
lambda_values

# Modify datasets
Xtrain = select(train_data, -c(after_24hr_comments)) %>% data.matrix()
Xtest = select(val_data, -c(after_24hr_comments)) %>% data.matrix()
Ytrain = train_data$after_24hr_comments
Ytest = val_data$after_24hr_comments
```

```
# Store the RMSE results
train_RMSE <- c()
train_RMSEw <- c()
test_RMSE <- c()
test_RMSEw <- c()

# Loop over n
for (lam in lambda_values) {
  print(lam)
  # Fit the lasso model
  fit <- glmnet(x = Xtrain, y = Ytrain, alpha = 1, lambda = lam)

  # Make train set predictions
  predictions <- predict(fit, newx = Xtrain)

  # Calculate the train RMSE
  RMSE <- sqrt(mean((predictions - Ytrain)^2))
  # Store the RMSE
  train_RMSE <- c(train_RMSE, RMSE)

  # Calculate the train RMSE-w
  RMSEw <- do_rmsew_lasso(fit, Xtrain, Ytrain)
  # Store the RMSEw
  train_RMSEw <- c(train_RMSEw, RMSEw)

  # Make test set predictions
  predictions <- predict(fit, newx = Xtest)

  # Calculate the test RMSE
  RMSE <- sqrt(mean((predictions - Ytest)^2))
  # Store the RMSE
  test_RMSE <- c(test_RMSE, RMSE)

  # Calculate the test RMSE-w
  RMSEw <- do_rmsew_lasso(fit, Xtest, Ytest)
  # Store the RMSEw
  test_RMSEw <- c(test_RMSEw, RMSEw)
}

RMSE = c(train_RMSE, test_RMSE)
RMSEw = c(train_RMSEw, test_RMSEw)
Error = factor(c(rep("Train", length(lambda_values)),
                 rep("Test", length(lambda_values))))
lambda_values = c(lambda_values, lambda_values)
df_RMSE = data.frame(lambda_values, RMSE, Error)
df_RMSEw = data.frame(lambda_values, RMSEw, Error)
print(df_RMSE)
print(df_RMSEw)
min(RMSE)
min(RMSEw)
which.min(RMSE)
which.min(RMSEw)
```

and here is the function definition for w-RMSE that we used to evaluate models to estimate generalization error with 1.3 weighting on underestimate predictions:

```
evaluation_cont_w <- function(model, data){
  sum_squared <- 0
  for (i in 1:nrow(data)) {
    resid <- data$after_24hr_comments[i] - predict(model, data)[i]
    if(resid > 0){
      sum_squared <- sum_squared + (1.3 * (resid)^2)
    }
    else{
      sum_squared <- sum_squared + (resid)^2
    }
  }
  rmse <- sqrt(sum_squared / nrow(data))
  return(rmse)
}

# Same logic as above, just in data parameter format needed for...
#...finding best lambda to use in lasso regression#
do_rmsew_lasso <- function(model, xdata, ydata){
  sum_squared <- 0
  xval <- predict(model, xdata)
  for (i in 1:nrow(xdata)) {
    resid <- ydata[i] - xval[i]
    if(resid > 0){
      sum_squared <- sum_squared + (1.3 * (resid)^2)
    }
    else{
      sum_squared <- sum_squared + (resid)^2
    }
  }
  rmsew <- sqrt(sum_squared / nrow(xdata))
  return(rmsew)
}
```

Project Part 3 for MS&E 226, Andrew Hong and Annie Zhu, Winter 2022-23

Test, Infer, and Summarize BlogFeedback Dataset

1 Prediction on the test set

Regression Task. Using the best regression model on our test set resulted in a prediction error (based on weighted RSME from part 2) of 27.85, only slightly higher than our estimated 27.78 from part 2.

Classification Task. Applying the best model for the classification task to the test set resulted in a prediction error (based on 0-1 loss) of 0.107. In comparison, our predicted generalization error from part 2 was 0.105 so we were quite close with that estimation.

Discussion. In both cases, our predictions were slightly underestimating error because we had used the validation set to both pick the best model and predict error, which introduces bias since we are using the same dataset for training and testing purposes. But, overall our predicted test errors were very close to true test error as the validation set wasn't used to fit the model.

2 Inference

We are choosing our baseline linear regression model (OLS w/all covariates) for inference analysis. We select this model as opposed to the chosen model from part 2 because inference analysis on lasso models is very difficult and the difference in the predicted test error between these two models was negligible (27.78 vs 27.83).

Throughout our inferential analysis we assume a few conditions: One, that our population model is linear. Two, that sample data consists of all independent draws from the population where all covariates in the population model are present. In particular, errors are independent and identically distributed. Three, the errors have 0 mean conditional on the sample X covariates: $E[\varepsilon|X] = 0$; they are also homoskedastic. And four, the errors are normally distributed around a mean of 0.

It is important to note that some of these assumptions may be broken in reality by our data set, which could impede on the reliability of the inferences we will make in this report. Specifically, the second assumption is likely broken because we do not know how the data was scraped (as we mention in part 1) and if the posts were scraped from a few popular sites instead of from all over the internet, we may not have independently-drawn observations.

2.1 Significant Coefficients

We define a statistically significant coefficient to be a coefficient where the likelihood that, assuming its value is equal to zero, we obtain the data in our observed data set is less than 0.1%. We chose this threshold instead of 5% due to our uniquely high number of p covariates (280). We are more likely to obtain more statistically significant covariates, so we lowered our threshold to 0.001 to extract a concise number of the most significant covariates. Using the build it lm function in R, we get that the statistically significant coefficients are:

Variable	P-value
before_24hr_comments	0
time_length	2.77e-58
from_48hr_to_24hr_comments	1.74e-19
total_before_comments	8.72e-09
med_from_48hr_to_24hr_comments	2.04e-07
freq_word_feature_179	1.43e-05
freq_word_feature_134	4.36e-05
freq_word_feature_15	1.85e-04
freq_word_feature_23	2.60e-04
avg_from_48hr_to_24hr_comments	2.73e-04
avg_before_24hr_comments	2.73e-04
avg_total_before_links	2.73e-04
avg_total_before_comments	5.67e-04
freq_word_feature_34	9.29e-04

Moreover, the p-value of a coefficient represents the probability that we see data at least as extreme as our training data given that coefficient equals zero. As a result, having a p-value under 0.001 means it is very unlikely that we would get our data under the null hypothesis that the coefficient is equal to zero. As a result we can conclude that these coefficients are most likely not equal to zero and are therefore significant.

Across our model's covariates, there is a logical pattern between each of our statistically significant covariates. We have several significant covariates measuring average number of comments on both specific posts and their platforms in total and across certain timestamps before the basetime, which makes sense as a significant covariate since a larger number of comments on blog sites and posts before intuitively suggests larger number of comments on respective posts in the immediate

future. We also have significant covariates relating to the length of time the post has been up for, which makes sense because newer posts tend to have higher comment activity. Finally we have 4 200-bag-of-words significant covariates, which make sense given that blog posts that use commonly-used words are likely more popular (more comments) since those words (and related topics) are more popular and have higher demand among blog consumers.

2.2 Fit to test data

When we fit our chosen model on the test set, we get six less significant coefficients. This could be explained by the fact that our test set is 13% of the size of our train set, and the decreased dataset size increases the standard error of our coefficients (from higher variance), which in turn decreases the number of coefficients that are statistically significant by our selected p-value threshold compared to the train-fitted model.

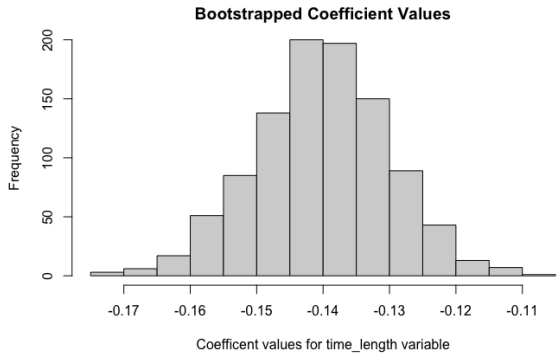
Additionally 4/8 of our significant coefficients are different from the train-set-fitted model's significant coefficients. This can be explained by a few reasons. First, because our test and train sets are time series with disjunctive time ranges, the distributions of our train and test sets may be different as time series could be a factor in our distributions that we do not control for. This is especially relevant as internet blogs rapidly change in traffic and consumer behaviors in short time spans as digital and social media exploded in the early 2010's-where our dataset is from. The differences in data distributions between the train and test sets from time series very well may impact which coefficients are significant between the two (differently) fitted models.

Secondly, because we have so many covariates in our model (280), our model is very prone to overfitting. This would explain why fitting the model on different datasets would result in different significant coefficients as each fitted model would overfit its significant coefficients to the data it's trained on.

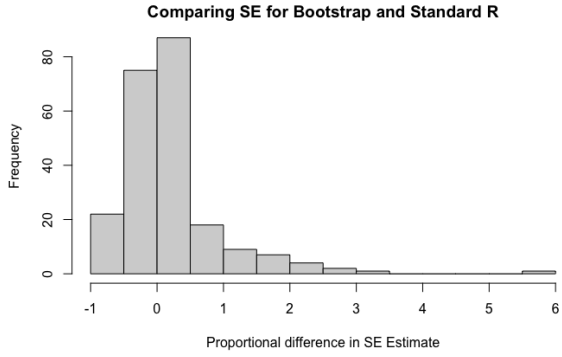
2.3 Bootstrap

Our bootstrap procedure involves simulating 1,000 "parallel universes." At each iteration, the program randomly samples 30,496 rows with replacement from the training set (since 30,496 is the number of rows from our training set), and then we train a linear regression model on this newly sampled data set. We record the coefficients outputted by this model. By the end of the procedure, we have a data frame with 1,000 rows of simulated coefficients.

To calculate the confidence intervals, we calculate the standard deviation among the 1,000 simulated coefficients for each covariate. Those standard deviations become our estimated standard errors (SE_{BS}). Then, our 95% confidence interval is $[T-1.96SE_{BS}, T+1.96SE_{BS}]$, where T is the coefficient value from our original non-bootstrapped model. Here is an example of the bootstrapped output for time_length's coefficient:



We get that the 95% confidence interval for the time_length variable is [-0.1606, -0.1210] meaning that in 95% of possible simulations, this interval captures the true coefficient value. Next, to summarize the observed difference between the standard regression and the bootstrapped confidence intervals across all 280+ coefficients, we created a histogram.



Since confidence intervals are built around estimated standard errors, this histogram showcases the frequencies of the proportional difference between the SE estimates for each coefficient. From the histogram, it can be observed that the large majority of coefficients had bootstrapped SE values within ± 1 of the standard regression SE values. However, there are some

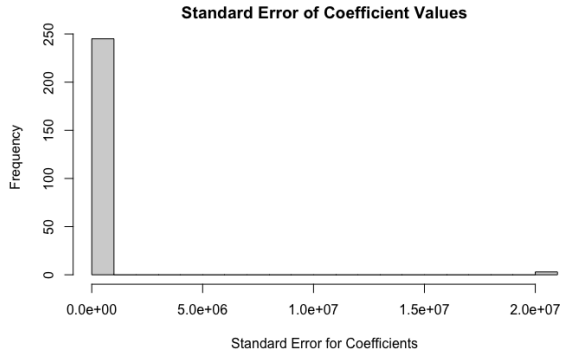
notable outliers with over 5x larger bootstrapped SE values compared to the standard regression SE values.

We believe that these large differences come violations to the assumptions necessary for standard regression. In particular, the covariates for which the standard regression model vastly underestimated its standard error are covariates representing average statistics and we think these covariates could be linearly dependent on other covariates, violating assumption A2 from our checklist. See Section 3.1 Collinearity for a more in-depth discussion on this topic.

3 Additional Inference (Recommended Work)

3.1 Collinearity

There is evidence of potential collinearity within our model. The largest standard errors came from avg_first_24hr_comments and avg_first_24hr_links and avg_change_between_two_days_links. These covariates have a very high SEs over x10,000 larger than the median standard error of 1.457. (As observed from the histogram below, these SE values are clear outliers.)



These big standard errors may suggest these covariates have collinearity with other (or a combination of other) covariates, which makes sense as these covariates representing averages of X-statistic per site (ex: avg_before_comments) are a computation (average) of statistics (ex: before_comments) of individual posts in sites, which are present covariates in our model already.

Because there may be collinearity in our model, our model's statistically significant coefficients may not be the actual statistically significant coefficients since we initially assume we have no collinearity in our linear model when deriving our coefficients and p-values.

3.2 Multiple Hypothesis Testing

With over 200 covariates, we certainly think multiple hypothesis testing is artificially inflating how many significant coefficients we see. Specifically, by nature of using a 0.001 cutoff in multiple hypothesis testing, we can already expect $280 * 0.001 \approx 3$ covariates to be significant due to random chance alone from multiple hypothesis testing. To ensure we're rigorously identifying our significant coefficients (not from chance), we implemented the Benjamini-Hochberg procedure (BHP). Using BHP, we optimized our "significant" coefficients from 14 to 6. Specifically, these six: 1) med from 48hr to 24hr comments, 2) total before comments, 3) before 24hr comments, 4) from 48hr to 24hr comments, 5) time length, and 6) freq word feature 179.

Because BHP ensures that the false discovery rate is less than or equal to α (which is 0.001), that means out of these six identified coefficients, we will have made mistakes at most 0.1% of them.

To compare, we also implemented the Bonferroni correction. This method yields 5 significant coefficients; all the same ones as from the BHP except for the freq word feature 179 coefficient. This is to be expected since the Bonferroni correction controls for the family-wise error rate which is more conservative than BHP.

3.3 Causality

Despite identifying significant coefficients in our model that may have strong associations with our outcome variable, we cannot prove causal relationships. This is because we are unable to control for confounding variables since we cannot randomly assign treatments (covariate values) to each observation in a controlled experiment that is needed to reliably determine causality. Thus we warn our clients not to assume causal relationships between any significant covariates (ex: time length of post) to the outcome variable (# of comments 24hr post-basetime) as we can't reliably prove this.

3.4 Subset of Covariates

We tested whether narrowing the number of covariates in our model would impact the significant coefficients by running a LASSO regression using all covariates without transformation (equivalent to our chosen model in this report) and removing covariates from our OLS model here that LASSO eliminated (set coefficient to 0). It's important to note that this is post-selection

inference, so we would be introducing some error to our parameter estimations of this updated OLS model with removed covariates (from LASSO results).

We get half as many significant coefficients likely from removing covariates from our model strategy that LASSO canceled out. Otherwise, 6/7 of the significant coefficients from this LASSO-defined model are the same as our selected model with all covariates. This makes sense because LASSO shrinks the number of covariates in the model by penalizing large coefficient values. Thus removing covariates with smaller coefficients usually does not remove statistically significant coefficients we observed from our initial OLS model.

4 Discussion

4.1 Practical Applications

In part 1, we defined a few possible practical applications of our model. First, we wanted to build a model that could predict the number of comments a blogpost would receive 24 hours after a point (basetime) given data on our X covariates. We accomplished this as we constructed and fitted a model that predicted this outcome with the lowest estimated error after trying a dozen of model strategies. This model has extremely useful practical applications for blogsites who want to efficiently allocate resources to blogposts with greater predicted future comment activity and bloggers (from activists to journalists) who want to increase their post activity to engage their message and stories to greater audiences; these were outlined goals of our project.

We also wanted to extract which variables (ex: key words or timeliness of posts) were best indicators of an increase or magnitude of future comments on a given post. However we were unable to accomplish this goal as we could not identify causal relationships between covariates and our outcome variable given the constraints of our dataset as described in Section 3.3 Causality.

4.2 Usability Over Time

We do not believe our model would hold up over time because of the nature of our domain: internet blog posts. The internet and social media platforms are rapidly changing. Platforms, users, authors, features, social norms, laws, and the technology sector generally is constantly changing (especially in the time since this data was collected in the early 2010's) so our models need to be refitted regularly as technology and society's relationship with the technology evolves exponentially.

4.3 Model Use

There are a few things we believe our clients should be aware of. First, the sheer number of covariates (280) has some unique ramifications on our model's interpretability and efficacy compared to models with smaller numbers of covariates. While the larger number of covariates generally means we can optimize our model to better predict our outcome on a dataset, it makes our model more prone to overfitting the dataset it was trained on. This may cause increased error on the intended predicted outcome when this model is applied to new data, especially as new data has a time series component where future data may be very differently distributed compared to the early 2010's-era training dataset we fitted the model to. Thus, we warn our clients to regularly re-fit the model and be aware of its vulnerability to overfitting.

Indeed, we tested several different regression models with complex transformations and engineered features, but we found adding transformations to the baseline model strategy increased estimated test error on our validation set (new data). Thus, we selected this simple model and advise against adding complex transformations even if it decreases error on one individual dataset.

Finally, we have very large confidence intervals for coefficients representing average statistics in blogsites (ex: avg. no. of comments or linked pages) that we derived from bootstrapping, so those coefficients may not be accurately portrayed in our model. Those coefficients also had wildly larger standard errors. When interpreting our model, we remind you of those very large SEs and CIs to take those coefficient values with a grain of salt.

4.4 Reflection

If we were to redo our project, we would have done deeper analysis in part 1 on relationships between covariates and our predicted outcome Y (# comments post-24hr) so we could have more intuition about which covariates to emphasize and interact with in our modeling stages. We had such a vast amount of covariates that we would have benefited from spending more time than we had to think and test out different relationships within our data to gain better intuition ahead of constructing model strategies, and subsequently potentially construct better models.

5 Appendix and Code

5.1 Bootstrap and Confidence Interval Code

We implement bootrap procedure.

```
coef.boot = function(data, indices) {
  fm = lm(data = data[indices,], after_24hr_comments ~ .)
  return(coef(fm))
}
boot.out = boot(regression_train_data, coef.boot, 1000)
```

and we compute the confidence interval using the code below.

```
compute_confidence_interval <- function(vec, name) {
  # Set values
  est <- model1$coefficients[name]  # Point estimate
  se <- sd(vec)   # Standard error
  alpha <- 0.05  # Significance level

  # Compute critical value
  cv <- qt(1 - alpha/2, df = Inf)

  # Compute confidence interval
  ci_lower <- est - cv * se
  ci_upper <- est + cv * se

  # Print results
  cat("Confidence interval for", name, "is : [", ci_lower, ", ", ci_upper, "]\n")
}
```

5.2 Multiple Testing Code

We implement the Benjamini-Hochberg procedure.

```
bh.adjust <- function(pvals, alpha=0.001) {
  # Get number of hypotheses tested
  m <- length(pvals)

  # Calculate Benjamini–Hochberg critical values
  k <- 1:m
  crit <- alpha * k / m

  # Sort p–values in ascending order
  sorted_pvals <- sort(pvals)
```

```
  # Find the largest index k such that p[k] <= crit[k]
  index <- max(which(sorted_pvals <= crit))

  # Adjust p-values using the largest index
  adj_pvals <- pvals * m / index

  # Return adjusted p-values
  return(adj_pvals)
}

# Calculate p-values for coefficients
pvals <- summary(model1)$coefficients[,4]

# Perform Benjamini-Hochberg procedure on p-values
adj_pvals <- bh.adjust(pvals)

# Filter significant coefficients
coeffs <- as.data.frame(summary(model1)$coefficients)
sig_coeffs <- cbind(coeffs, adj_pvals) %>% filter(adj_pvals <= 0.001)

# Print significant coefficients
print(sig_coeffs)
```