ASSIGNMENT1
[CSE 574]

1. **Rajiv Nagesh – UBIT: rajivnag**
2. **Ishansh Sahni – UBIT: ishanshm**
3. **Shreya Joshi – UBIT: shreyajo**

Part 1.1 Understanding API's

1.1.1 How many API calls were required to collect the submissions?

Reddit limits 100 submissions per API call, thereby PRAW will be able to scrape only 100 submissions each call with a delay of 2 seconds after a call. Meaning, if one was supposed to request 250 items, it will require 3 API calls and can take at least 2*2=4 seconds. Ultimately, to answer the main question in focus, we were asked to get 1000 items from each subreddit which accounts up to 3000 items. **Doing the same math, API calls that were required for requesting 3000 such items from 3 different subreddits accounts to 30 API calls.**

Citing a link that was used to understand PRAW and Reddit API –
https://www.jcchouinard.com/reddit-api/
https://praw.readthedocs.io/en/v3.6.2/

1.1.2 Why did we set the submission limit at 1000?

As stated before, the Reddit API allows at most 1000 items to be scrapped from a certain listing. That includes subreddit submission listings, user submission listings, and user comment listings. **So, one request, Reddit will give 1000 submissions which will take 10 API calls. This is a limitation set by Reddit.** Even if someone had to set the limit to 1001, they would receive only 1000 submissions in that request cycle.

1.1.3 How long, in minutes, would it take you to collect 1000 posts from 25 different subreddits? What about 500 different subreddits?

Case 1 – 25 different subreddits and 1000 posts:

As from above, 1 API call gives 100 submissions. So, for 25000 submissions API calls required will be 25000/100 = 250 API calls.

Now to calculate the time required, there is a 2s delay after every API call. But we won't be considering the 250th API call in the calculation as there is no succeeding API call after the 250th. Hence, only 249 API calls will be required to calculate the time required. Doing the math, it comes to 249*2 = 498s. Therefore, 498/60 = 8.3min.

So, 8.3 minutes will be required to collect 1000 posts from 25 different subreddits.

Case 2 – 500 different subreddits and 1000 posts:

500000 submissions, API calls required will be 5000.

2s delay after every API call, using the same logic above; 4999 API calls will be used to calculate the time required which is equal to 9998s.

Therefore, in minutes: 9998/60 = 166.63min.

So, 166.63 minutes will be required to collect 1000 posts from 500 different subreddits.

Part 1.2 Thinking about your sample

1.2.1 Do you think these posts are representative of all the posts on that subreddit?
To answer this question, I can put forward an example of a subreddit that is very nascent (maybe a day old) and scrape posts of that subreddit and claim that the posts I received for that subreddit is representative of all the posts. So, factors deciding this are age of the subreddit, checking for total posts on the subreddit and not just "top" posts as we were asked to in the given assignment. **So, no, these posts are not representative of all the posts on that subreddit.** People can upvote any random post that is trending and that can be at the topmost of that subreddit but that doesn't quantify or explain anything about the other posts on that subreddit. It is a herd trend in a nutshell.

1.2.2 Why or why not? That is, if you think so, why do you think there's not much sampling bias here? If not, what do you think might be different about these top posts than other posts?
There is a sampling bias in this capturing of submissions. Why? -
I think there is a great sampling bias here as we are just asked to capture the "top" posts of a given subreddit. This filter itself is a self-explanatory classic example of ML bias. We cannot infer anything about a subreddit just by looking at the top posts. There are other factors like age of a subreddit, subscribers, etc. Difference in the top posts and other posts are that the other posts might not have that many upvotes as the top posts, they might be not as cross posted as the other top posts, etc. These differences are valid, but they cannot be overlooked, and the top posts cannot be justified as generalizing a subreddit.

Part 2.1 Univariate descriptive analyses

2.1.1 What are the names (subreddit_name_prefixed) of the 24 different subreddits that are in part2_data.csv?
Names of the 24 different subreddits are –
array(['r/Jokes', 'r/news', 'r/science', 'r/WritingPrompts', 'r/Showerthoughts', 'r/worldnews', 'r/todayilearned', 'r/learnprogramming', 'r/announcements', 'r/funny', 'r/food', 'r/sports', 'r/gadgets', 'r/aww', 'r/mildlyinteresting', 'r/memes', 'r/technology', 'r/travel', 'r/books', 'r/gaming', 'r/cats', 'r/conspiracy', 'r/PoliticalHumor', 'r/hockey'], dtype=object)

2.1.2 How many reddit authors (author_name) have a post in more than one unique subreddit in part2_data.csv (e.g., they have a top post in both r/news and r/hockey)?
There are 569 reddit authors having a post in more than one unique subreddit in part2_data.csv

2.1.3 What is the mean number of upvotes (ups) for posts in r/Jokes?
41057.7813440321 is the mean number of ups for posts in r/Jokes.

2.1.4 What is the variance of the number of upvotes in r/news?
600707867.6203133 is the variance of the number of ups in r/news.

2.1.5 What is the standard deviation of the number of upvotes received across the entire dataset?
43102.4844737104 is the standard deviation of the number of ups received across the entire dataset.

2.1.6 Mathematically, what is the relationship between the standard deviation of the number of upvotes and the variance of upvotes?
Standard deviation is the spread of a group of numbers from the mean. The variance measures the average degree to which each point differs from the mean. The standard deviation of upvotes for the entire dataset is **43102.4844737104** and the variance of upvotes for the entire dataset is **1857824167.806446**. This means that the ups of the entire dataset is centric from the mean as the SD is not very huge.

2.1.7 Which subreddit had the third highest median number of up-votes?
r/aww with 109811.0 has the third highest median number of up-votes.

2.1.8 What is the conditional probability of an author having a top post in r/news, given that they have a top post in r/worldnews?
The conditional probability computes out to 0.067

PART 2.2 NEXT PAGE

Part 2.2 Plotting

2.2.1 Submit your histogram in the assignment

2.2.2 Based on your histogram, which subreddit would you say is the *least* popular? (Note, there is more than one reasonable answer here. We are looking mostly for how you justify your response using the histogram)

We can infer that the 'r/announcements' subreddit is the least popular as that particular subreddit does not have much distribution of ups as comparatively to the other subreddits and the submissions scraped in the par2_data.csv to the former is also lesser than the other subreddits.

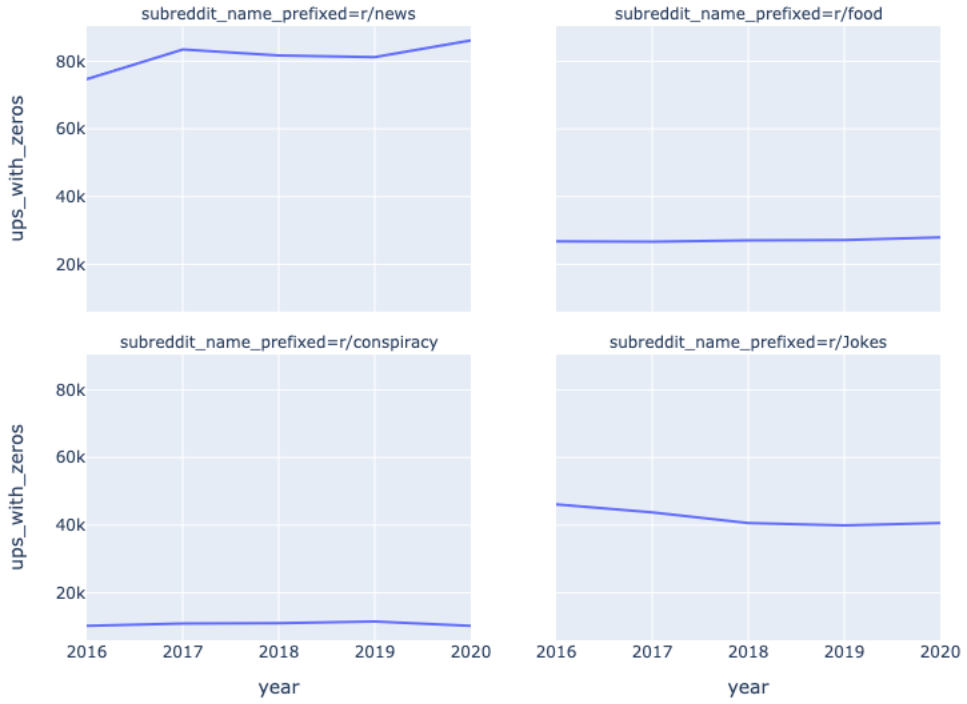
2.2.3 **Approximately (within 1-2 percentage points)** what percent of top posts for each of the three subreddits plotted below have less than 100,000 upvotes? (Give answers for each subreddit)
For the subreddit **r/news**, percent of top posts having less than 100,00 ups is **83.9**
For the subreddit **r/science**, percent of top posts having less than 100,000 ups is **98.4**
For the subreddit **r/wroldnews**, percent of top posts having less than 100,000 ups is **78.9**

2.2.4 **Approximately (within 1-2 percentage points)** what is the probability that a post on each of the three subreddits plotted below has more than 70,000 upvotes? (Give answers for each subreddit)
The probability of subreddit **r/news** having ups more than 70,000 is **0.74**
The probability of subreddit **r/science** having ups more than 70,000 is **0.13**
The probability of subreddit **r/worldnews** having ups more than 70,000 is **0.97**

2.2.5 How many posts in the dataset were sent in 2010?
35 posts in the dataset were sent in 2010.

2.2.6

2.2.7



2.2.8 Using what you have plotted, make an argument for which of the four subreddits is the most "up and coming" - i.e., the one that seems to be getting more popular over time. NOTE: There is more than one reasonable answer here. We are looking for how you justify your answer using the (plotted) data.

According to the four plots, we can see that the **r/news** subreddit is the most popular since 2016 till present by looking at the ups and comparing the latter with the other 3 subreddits. Having said that, upvotes are not the only deciding factor of how popular a subreddit is, many factors like cross posts, subscribers, age of that subreddit also matters.

Part 2.3 Data Cleaning & Regression-related Analyses

2.3.1 There are two continuous variables that are very clearly not going to be useful for our analysis. Identify them and explain why they are not useful (**note: you do NOT need to know why these variables take on the values they do in our data. You just need to know why we don't want to use them!**)

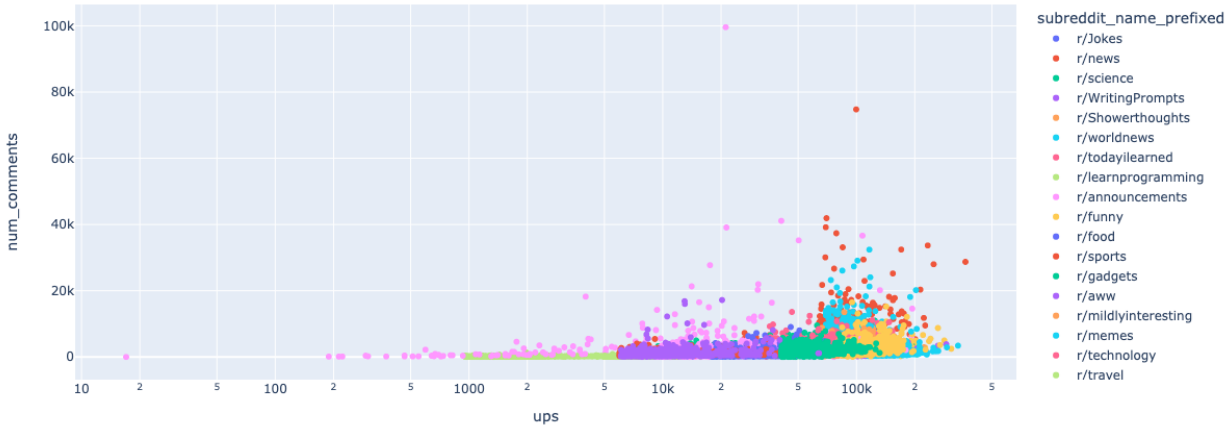
According to my understanding, downs and num_reports are the two continuous variables that are clearly not going to be useful for the analysis. This is because these two variables do not provide on the popularity of a subreddit and do not contribute any significant relevance related to ups. If these two are considered for modeling the ups, they would be outliers to the target variable.

2.3.2 There are two (supposedly) binary variables that are very clearly not going to be useful for our analysis. Identify them and explain why they are not useful.
Binary variables in our dataset are 1's and 0's or True and False. According to the given list of binary variables, the is_self and locked variables. The is_self variable is dropped because it is only a text post and cannot be shared outside of reddit and this variable won't provide any importance to the prediction of ups. The other binary variable that is dropped is the locked variable. This variable just disables comments and people can still up vote a post and it won't be useful for the prediction of the ups for a model.

2.3.3 Explain why it is not useful to use *both* subreddit_id and subreddit_name_prefixed in any predictive analysis of per-post upvotes.
It is a rule of thumb to always try and use numerical values for a model and it will be easier to handle the data. As we see, the subreddit_name_prefixed field is a character variable. The PRAW library allows us to convert the subreddit_id into objects and thereby getting the names of the same. (reddit.info)
Using this feature, we can convert a subreddit_id into an object and then return it's subreddit_name. Hence, it is advisable to drop the subreddit_name column and only keep the id's.

2.3.4 Explain why it is not useful to use permalink in any predictive analysis of per-post upvotes.
Permalink will just provide the URL of the post of the particular subreddit. This is not useful for predicting the ups for a model as it does not provide any relevance.

2.3.5 Plot the relationship between num_comments and upvotes as a scatterplot with log-scaled axes, with the posts from different subreddits as different color points. Paste this plot into your PDF writeup



2.3.6 Describe, briefly (a sentence) the relationship between num_comments and upvotes.
The relationship for ups and comments doesn't seem statistically significant. Consequently, we can see that the subreddit r/sports has a higher number of comments for the same corresponding high number of ups.

2.3.7 Which of these has the strongest positive correlation with ups?

```
# Code for 2.3.7-8 here
part2_data[['ups', 'total_awards_received', 'gilded', 'num_comments', 'num_crossposts', 'created_utc', 'subreddit_subscribers']].corr(meth
```

[9] ✓ 0.5s Python

	ups	total_awards_received	gilded	num_comments	num_crossposts	created_utc	subreddit_subscribers
ups	1.000000	0.388154	0.228103	0.330700	0.537982	0.165471	0.410248
total_awards_received	0.388154	1.000000	0.484246	0.146805	0.234456	0.263338	0.076032
gilded	0.228103	0.484246	1.000000	0.276973	0.210057	0.039404	0.099206
num_comments	0.330700	0.146805	0.276973	1.000000	0.210075	0.008073	0.284377
num_crossposts	0.537982	0.234456	0.210057	0.210075	1.000000	0.206429	0.271484
created_utc	0.165471	0.263338	0.039404	0.008073	0.206429	1.000000	-0.191099
subreddit_subscribers	0.410248	0.076032	0.099206	0.284377	0.271484	-0.191099	1.000000

According to the Pearson Correlation Table, we can see that the subreddit **'num_crossposts'** has the **strongest (0.53)** positive correlation with ups.

2.3.8 Which of these has the weakest positive correlation with ups?
According to the Pearson Correlation Table above, we can see that the subreddit **'created_utc'** has the **weakest (0.16)** positive correlation with ups.

Part 3.1 Regression Basics

3.1.1 Report your error on the test data, in RMSE. State what this metric means for the expected error in terms of the number of upvotes (not log upvotes!) you should expect to be off on any given prediction
The **RMSE** computed for our model comes out to **0.10** after splitting the dataset(part2_data) into an 80-20 split and keeping the ups as the target variable. RMSE is root mean squared error for a model, lower the error, higher the accuracy of that model. The model was trained on the continuous and binary variables as discussed earlier and the following were used to predict the ups. Logarithmic transformation helps convert a highly skewed data to a more normalized data.

3.1.2 What did the whole one-hot encoding thing on subreddit_name_prefixed actually do?
One hot encoding basically converts the categorical variables into various features with labeling them as "1' as "hot' or "on". It simply creates additional features based on the number of unique values in the categorical feature. Every unique value in the category will be added as a feature. The one-hot encoder converted the subreddit_name_prefixed into separate features so it will assign a 1/0 to each and thereby improved the prediction of the model.

Type
AA
AB
CD
AA

Onehot encoding →

Type	AA_Onehot	AB_Onehot	CD_Onehot
AA	1	0	0
AB	0	1	0
CD	0	0	1
AA	0	0	0

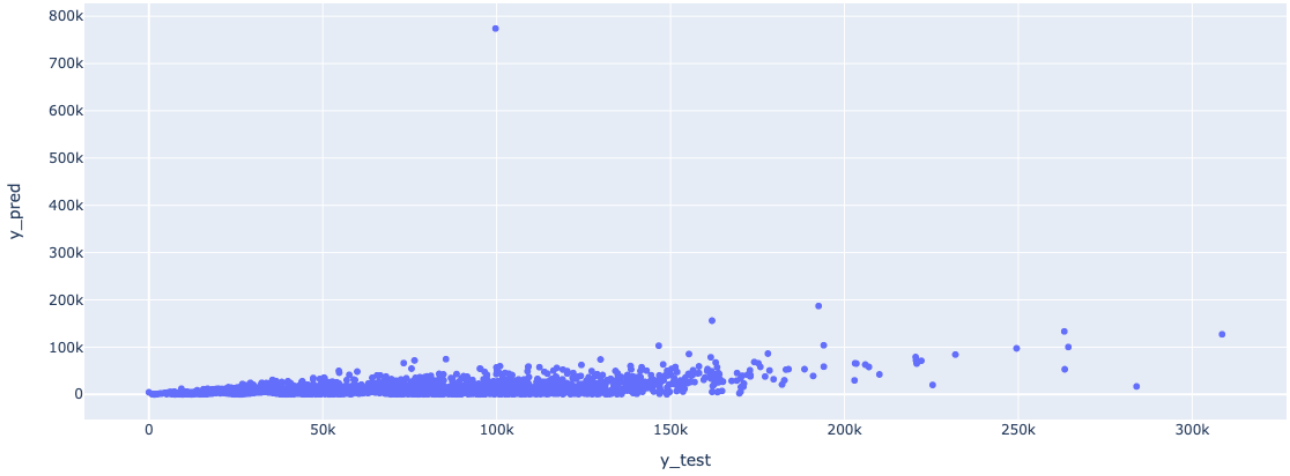
img src: https://www.educative.io/blog/one-hot-encoding

3.1.3 What does the argument drop = "first" do for us when we are doing that to subreddit_name_prefixed?
By dropping one of the one-hot encoded columns from each categorical feature, we ensure there are no "reference" columns, **the remaining columns become linearly independent**. Therefore, when using the normal equation to create a model, you must drop one of the one-hot encoded columns from each categorical feature. It will drop the first category for the subreddit_name_prefixed in each feature.

3.1.3 Why did we need to add one to the outcome variable before using log?
We had to add one because logarithmic transformations of values will lead to cases where there will be 0 as an answer and the inverse or log of the same will lead to infinite. So, for ease of computation, we have added 1 to the outcome variable before using log.

3.1.4 What does the StandardScaler do? Why do we want to do that?
StandardScaler **removes the mean and scales each feature/variable to unit variance**. This operation is performed feature-wise in an independent way. It involves in the estimation of the empirical mean and standard deviation of each feature and can be influenced by outliers. We technically want to do this because it standardizes the feature by subtracting the man and then scaling to unit variance. The latter means dividing all the values by the standard deviation.

3.1.5 Provide a scatterplot that compares the true values in y_test to the absolute value of the difference between y_test and your predictions. **The axes should be on the original scale** (i.e., not the log scale you're predicting on.)

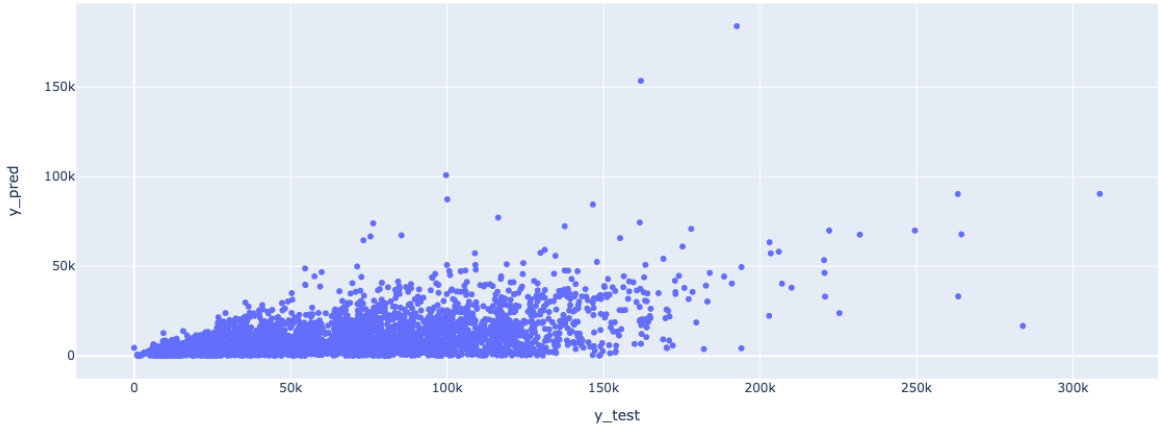


3.1.6 What does this plot suggest about how well your model fits the data as the true number of upvotes changes?
The scatter plot seen above is for the model without logging the independent variables. The RMSE is 0.105 for the model, which means there will be some generalization error by the times we train our model on the same dataset and the prediction of ups for this model will

have a lesser accuracy as the number of ups changes. More we train our model on the same data, the accuracy won't be substantially great due to the generalization error. To answer the question, the plot shows a variance as the number of ups for y_test increases, the y_pred is not as accurate as it is supposed to be.

3.1.7 What is the new RMSE with the logged independent variables?
After logging the independent variables, **the RMSE comes to – 0.091**
This is lower than the earlier (without logging independent variables) RMSE – 0.105

3.1.8 How did this compare to the old RMSE? Why do you think that is? Hint: It may help to re-plot the same figure as you did in 3.1.5, but with the new model, to answer this question.
The comparison of the two RMSE are –
OLD RMSE: 0.105 NEW RMSE: 0.091
When our original continuous data do not follow the bell curve, we can log transform this data to make it as "normal" as possible so that the statistical analysis results from this data become more valid. This is a probable reason why the RMSE of the new model is better than the previous model. Logging a data makes our original skewed data more normal thereby improving the linearity between the dependent and independent variables.



Part 3.2 Interpreting Regression Coefficients

3.2.1 What is the strongest positive predictor of upvotes? How many more log(upvotes+1) does a one standard deviation increase in the feature correspond to?
The strongest positive predictor of upvotes is the subreddit **'r/memes'** with a **regression coefficient of 0.792.** This means that a factor of 0.79 in the positive x direction for the subreddit 'r/memes' will lead to a one unit increase in the positive y direction for upvotes. Consequently, we can say that the regression coefficient of 0.792 of r/memes is directly proportional to the upvotes.

3.2.2 What is the strongest negative predictor of upvotes? How many fewer log(upvotes+1) does a one standard deviation increase in the feature correspond to?

The strongest negative predictor of upvotes is the subreddit **'r/learnprogramming'** with a **regression coefficient of -2.86.** This means that, on average, each post gained in r/learnprogramming is associated with a decrease of -2.86 upvotes since the subreddit r/learnprogramming is continuous.

Part 3.3 574 Only. Attempting to Improve Your Predictions

3.3.1 Describe at least two changes you made – at least one to the feature set, and at least one different model – to try to improve prediction. Explain *why* you think that these changes make sense, given the Exploratory analyses above, or any other exploratory analysis you choose to do.

Changes Made:

1) Features – Dropped features from the previous model viz. The Binary Vars (created_utc, is_self, locked, is_video)
2) Model - Random Forest
3) Train Test Split – 75:25

Starting with the features, we have chosen to drop these features mentioned above as they do not contribute positively to our target variable as seen in the regression coefficients. They are negatively correlated and dropping these features was a wiser option as they also were just categorical variables with no such major significance to the target variable in the prediction. Furthermore, we switched from the linear regression model to the Random Forest model as this seemed to be the most widely used model for any ML approach as it uses the decision tree algorithm approach thereby producing a higher accuracy and less error. The Random Forest model ensures that there is no/less bias in the model and there is no segment of data or a set of features that dominates the prediction results. Consequently, we were doing an 80-20 Train-Test split for the linear model; we changed these numbers to a 75-25 Train-Test split to see if this positively or negatively affected our Random Forest model. To our surprise, it positively affected and reduced the RMSE of the model. **The RMSE of the Random Forest model with the feature reduction and split sizes mentioned above came to 0.079.** This RMSE was the lowest recorded of the three models that we built for this assignment.

3.3.2 By how much did your RMSE improve? Which change that you made improved it the most? How do you know?

RMSE Recorded for all Models:

1) Linear Regression Model without logging independent variables – 0.105
2) Linear Regression Model with logging independent variables – 0.091
3) Random Forest Model with feature reduction and 75:25 split – 0.079

The RMSE has improved by 12 percent from the Linear Regression model with logging independent variables to the Random Forest model. The change that seems to be the most impactful is the new type of pf model from Linear to Random Forest. As answered above in 3.3.1, the Decision Tree Algorithm and nbags concept of Random Forest model handles huge data easily and improves the accuracy of the model. Also, reducing the features from the list like the Binary vars viz. is_self, created_utc, is_video,

locked that negatively correlated to the target variable led to the increase in the accuracy and the decrease in the RMSE. Consequently, Random Forest gave the best RMSE of all the three models built in this assignment and feature reduction played a vital role in the prediction too.