Machine Learning Project Report

Index

Introduction

In the realm of machine learning, the quest for optimal solutions to complex problems often involves exploring diverse methodologies and techniques. This pursuit is driven by the recognition that different approaches may offer unique insights, trade-offs, and performance characteristics. In this paper, we present our investigation into predicting the number of likes that a review will obtain based on a set of textual features using two distinct approaches, plus a third attempt, within the machine learning paradigm.

This regression problem poses significant challenges, demanding sophisticated algorithms capable of extracting meaningful patterns from data to make accurate predictions or decisions. Recognizing the multifaceted nature of this challenge, we have adopted a multifaceted strategy, leveraging the strengths of two approaches: the first, more conventional, which we'll refer to it as "*Standard Approach*" and the second, more experimental, we'll refer to it as "*Local Variance Minimization Approach".*
Since we weren't satisfied with the previous results, we gave a try to a third methodology that we will call "*Frequent Itemsets Approach".*

Approach N°1 – Standard Approach

In this first approach, we started the process by training models on all of the available features, using a variety of machine learning algorithms for regression – such as Linear, Lasso and Ridge Regression, Random Forest, Adaboost and Gradient Boosting – and a variety of parameters using GridSearchCV, we found ourselves making very little progress, meaning R-squared of 0.2 or less. We decided to perform some Exploratory Data Analysis, in order to help us reduce the feature space, and allow the algorithm to more easily focus on features that provide value to the model.

Our first course of action was to examine the target variable, shares. We found the data to be heavily skewed (as can be seen in the figure below), and that an appropriate course would be to take the log of the shares, in order to normalize the distribution and make the relationship between the target variable and the predictors more linear, and also to reduce the variance to have more stable models.

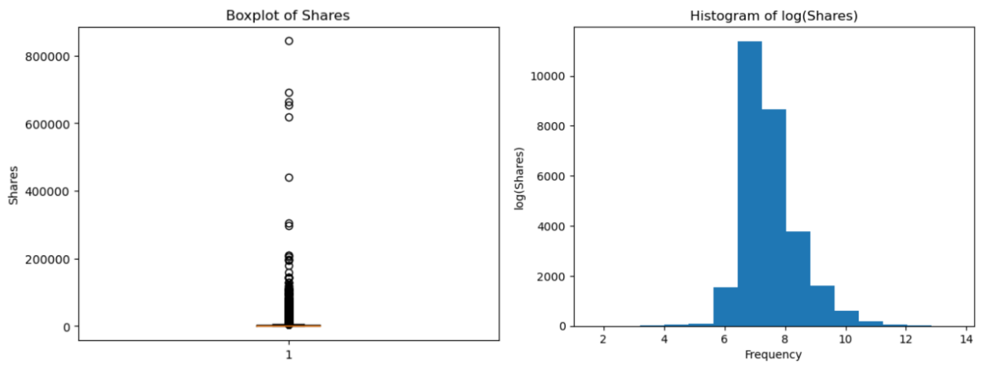


Figure 1: Boxplot of shares, and histogram of log of shares.

As can be seen in the figure next to the boxplot, taking the log revealed a fairly normal distribution of the target variable. Showing to us that the target variable had a very heavily skewed, log-normal distribution. In order to make the problem of regression simpler, we would choose to use the log of our target variable instead, so that larger datapoints would not be too spread out for a regression algorithm to be able to make valuable inferences on them.

After examining our target variable, we started with mapping the relationship between shares and our explanatory variables. What we found was very unhelpful, with the following figures exemplifying what we generally saw. No visible correlation could generally be understood from the shares value to any of the explanatory variables. Figures like the ones we see below can be shown to characterize the relationship, as mostly a cloud of points with nothing discernible to tell us whether this variable will have predictive power.

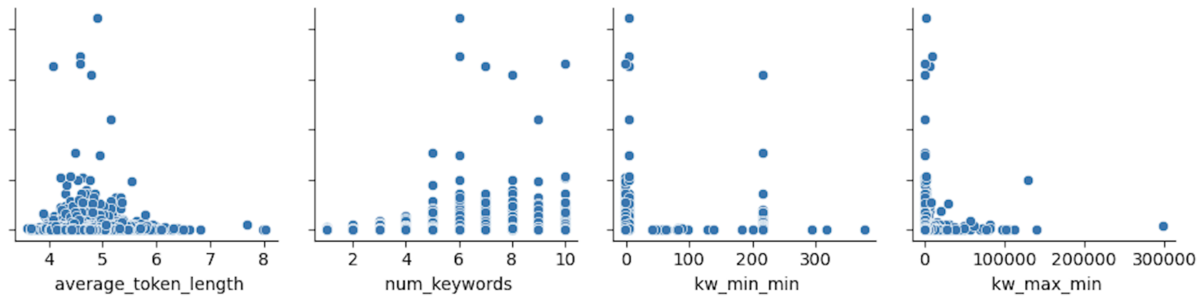


Figure 2 Log of shares plotted against some of our explanatory variables.

In addition to looking at plots of the mapping of our target variable to each of our explanatory variables. We ran a correlation analysis, and outputted a heat map. What we found regarding our relationship the log of shares, is what we had already seen from our visual process. With this information we are not able to remove any features because from this analysis we can see they are all equally unhelpful.

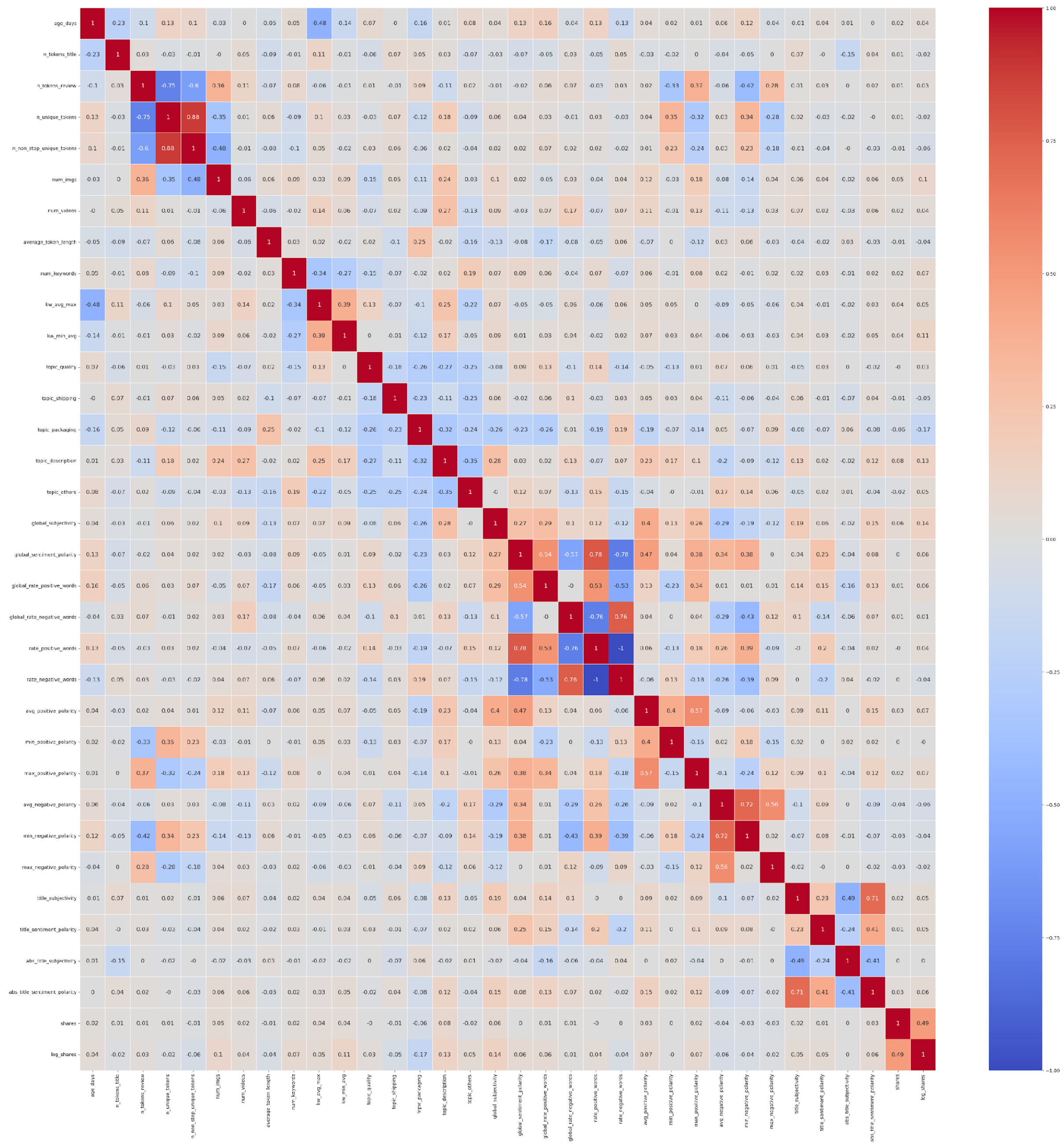


Figure 3 Covaraince matrix of our dataset

From here we decided to investigate the distributions of the explanatory features themselves. Based on this analysis we were able to eliminate around 13 features. We got rid of them because their distribution was extremely lopsided, we did not expect them to be able to contribute to a regression problem, if so many of the values are almost exactly the same. Most of them were part of the "kw group", meaning a lot of variables regarding keywords.

This is because regression requires a relationship between the explanatory variable to the target variable, if that explanatory variable is so skewed, then our intuitions is that it's likely not able to map well onto the highly varied domain of the target variable. This would prove false later.

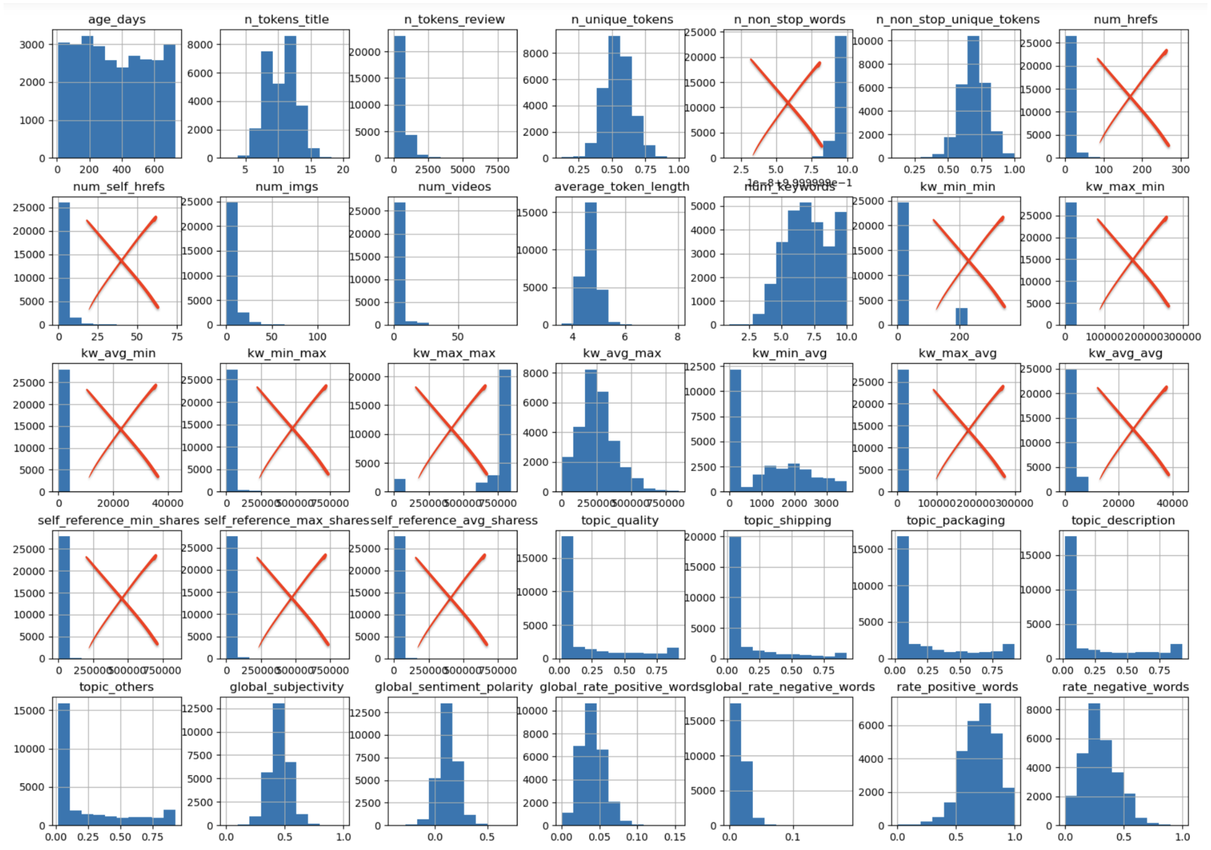


Figure 4 Histograms of all of our explanatory variables.

We decided to keep the categorical variables, because we did witness some interesting patterns in the difference of distributions in our target variable across both days and product categories. As we can see in the figure below, we have noticeable differences in both the means and the interquartile ranges of different product categories (which are greater than they appear, because we have taken the log of this data). We also see a slight, but noticeable differences across days as well. Most noticeable, weekend vs weekdays.

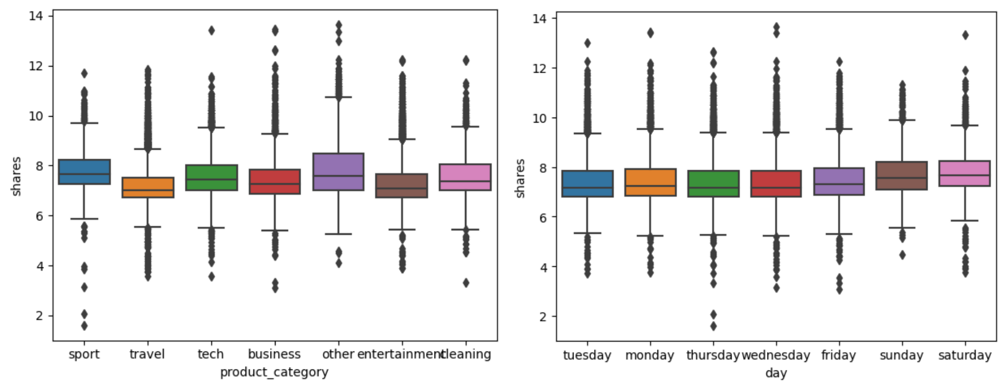


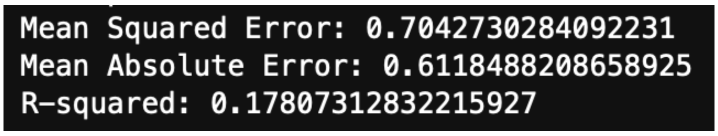
Figure 6 Boxplots of log of shares, split up by our dummy variables.

We ran the models and found out that the metrics were actually worse than what we got from using all the original columns. So we went back and ran our models on the original data.

After this we separated train and test set, performed a grid search for hyperparameter tuning on a given regression model using the GridSearchCV class from scikit-learn and then implemented the following models and gave us these results:

- Linear Regression
  - MAE train: 0.638, MAE test: 0.634
  - MSE train: 0.740, MSE test: 0.746
  - R-Squared train: 0.139, R-Squared test: 0.129
- Ridge Regression
  - MAE train: 0.638, MAE test: 0.634
  - MSE train: 0.740, MSE test: 0.746
  - R-Squared train: 0.139, R-Squared test: 0.129
- Lasso Regression
  - MAE train: 0.638, MAE test: 0.634
  - MSE train: 0.741, MSE test: 0.746
  - R-Squared train: 0.139, R-Squared test: 0.129
- KNN Regressor
  - MAE train: 0.614, MAE test: 0.624
  - MSE train: 0.716, MSE test: 0.758
  - R-Squared train: 0.168, R-Squared test: 0.115
- Decision Tree Regressor
  - MAE train: 0.644, MAE test: 0.653
  - MSE train: 0.744, MSE test: 0.779
  - R-Squared train: 0.136, R-Squared test: 0.091
- Random Forest Regressor
  - MAE train: 0.450, MAE test: 0.623
  - MSE train: 0.390, MSE test: 0.717
  - R-Squared train: 0.547, R-Squared test: 0.163
- MLP Regressor
  - MAE train: 0.605, MAE test: 0.622
  - MSE train: 0.678, MSE test: 0.753
  - R-Squared train: 0.212, R-Squared test: 0.143
- Adaboost
  - Best score: 0.107
  - Gradient Boosting
  - Best score: 0.161
- XGBoost
  - MAE test: 0.704
  - MSE test: 0.612
  - R-Squared test: 0.178

Unfortunately, none of them produced good metrics. The one that performed best was XGBoost, but it's noticeable that it's still nothing helpful, thse are the test values observed.



```
Mean Squared Error: 0.7042730284092231
Mean Absolute Error: 0.6118488208658925
R-squared: 0.17807312832215927
```

Figure 7 Results from XGboost

We then decided to pull out the shapely of the model and these were the results:

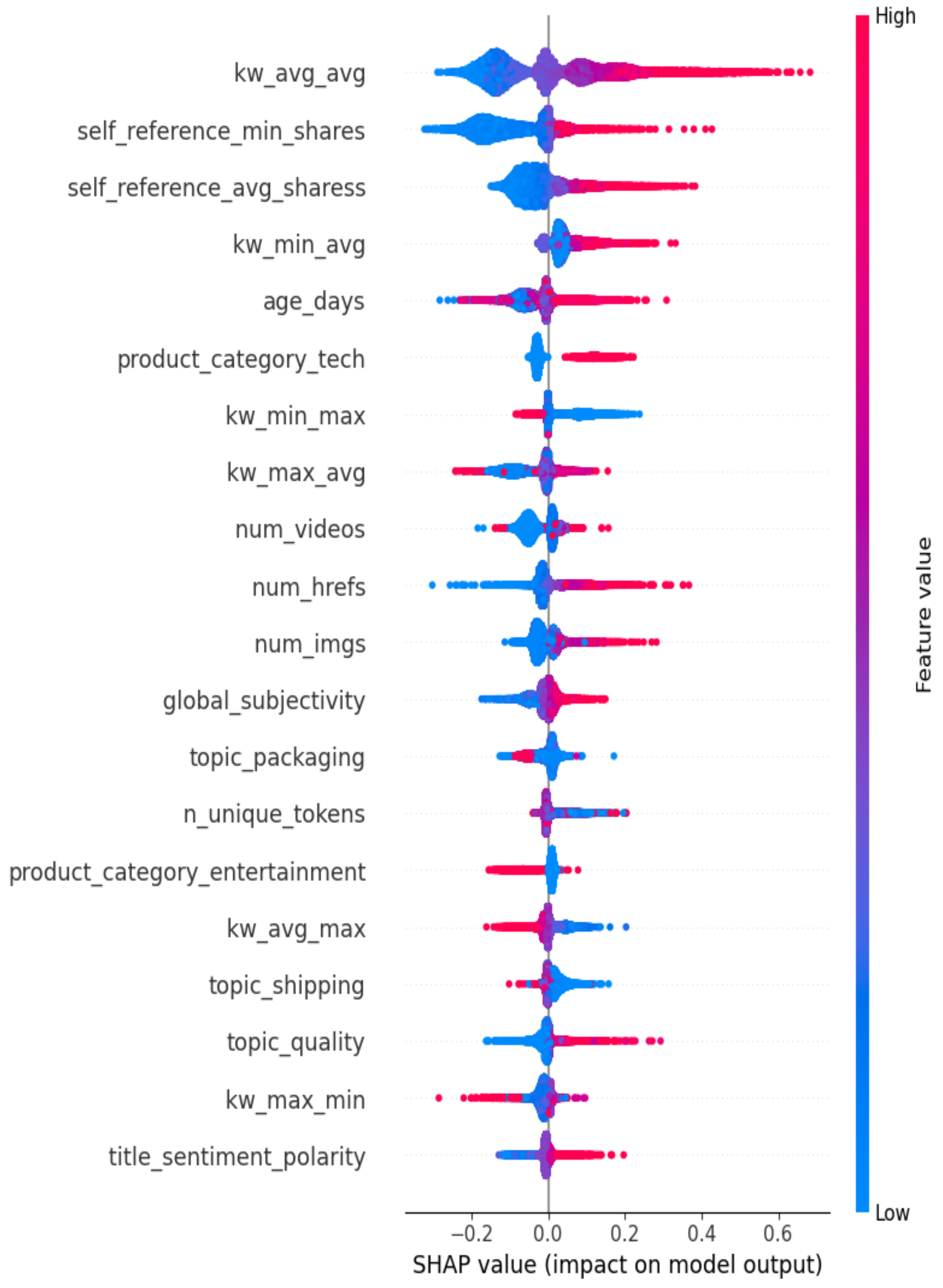


Figure 8 Shapley values plot

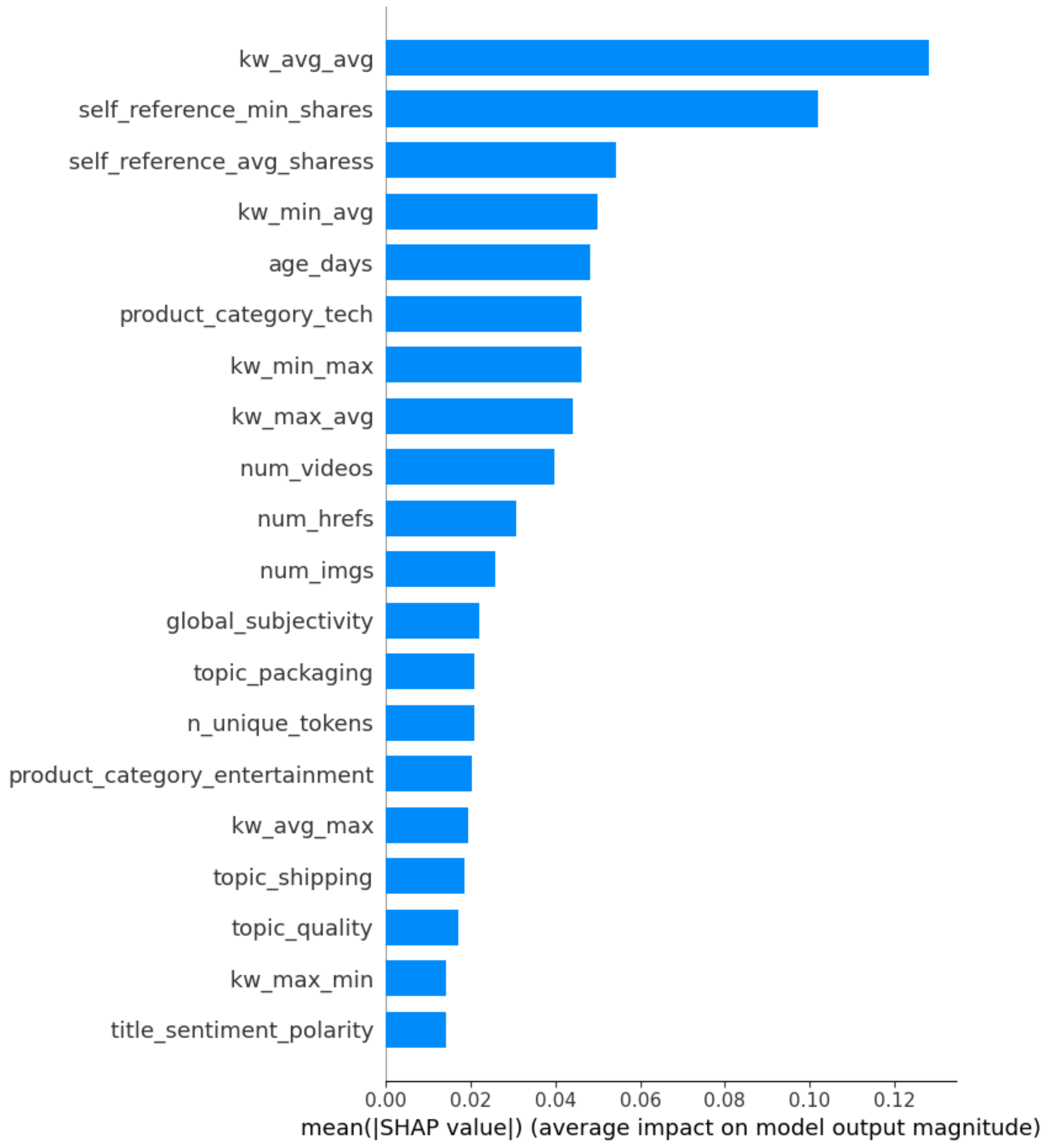


Figure 9 Shapley values plot

We can see that "kw_avg_avg" represents by far the most important variable in this dataset. All the other variables have a lower average impact on the model output, but some like "self_reference_min_shares" have a good impact as well (0.10). It should be noted though that except 6 variables all the other columns have an average impact always lower than 0.05. This would suggest us to drop some variables and only consider a few of them, with kw_avg_avg being the most important, but when we tried we got worse results.

Approach N°2 – Local Variance Minimization

In another attempt to perform our feature selection, we chose to propose a new algorithm that we will refer to as Iterative Local Variance Minimization. The basic premise relies on the idea that the neighbors to a data point will have similar target variable values in the optimal feature space, and as features that do not provide enough explanatory value are added to the space, the target variable values of a data point's nearest neighbors become farther away and more random. This assumption is clearly not always true, most clearly in segments of our space where the relationship between the explanatory and target variable changes. We will discuss how we introduce a heuristic measure added to the algorithm to remain more robust against this possibility later.

The algorithm works as such. In our first start step, we start by iterating through every feature, and finding the k-nearest neighbors for each data point based on the Manhattan distance. Then for each datapoint, we evaluate the variance of the target variable for the neighbors of each point. We then throw out the 100 highest variances that we find in this assessment (in an understanding that in places where model behavior changes, we can expect higher variances). After throwing out these values, we take the average of the all the remaining variances for each feature and keep the feature that has the minimum average variance. In the next step, we calculate the nearest neighbors, but in this case, we are calculating it with the previously picked columns as our first dimension, and then testing every remaining feature, to find the one that minimizes the variance of the target variable in the local neighborhood of each of the points. This process is continued. We can then plot the collected average variances for each of the added dimensions. We then look for the greatest number of dimensions we can observe before the average variance starts to increase. We can actually see this in the plot below.

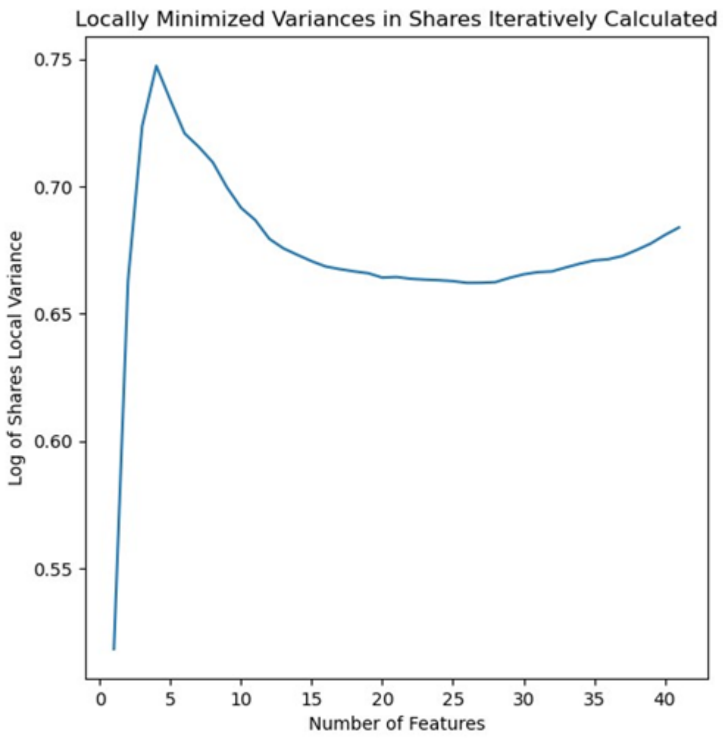


Figure 10 Plot of iteratively calculated local variances in target variable

The previous image shows the iteratively calculated local variance of the log of the target variable, plotted against the number of features.

Using the plot we have above, we decided 24 features to be an appropriate measure. In this case we are interested in maximizing the number of features that could provide explanatory value, because with around 30 thousand data points, we did not see our curse of dimensionality as large of an issue as it might be in other scenarios. We also added the dummy variables, for each of the days and the categories. The features picked were kw_max_max, kw_min_min, n_tokens_title, kw_avg_avg, num_hrefs, kw_max_avg, kw_avg_max, global_sentiment_polarity, num_imgs, n_non_stop_unique_tokens, topic_packaging, topic_shipping, max_positive_polarity, kw_max_min, num_videos, max_negative_polarity, min_negative_polarity, average_token_length, kw_min_avg, rate_positive_words, kw_avg_min, num_keywords, global_subjectivity, title_sentiment_polarity (in order of how they were picked).

Our results for this method yielded us mediocre results, in general it does not seem the explanatory variables are able to provide a good explanation of our target variable. The results we get are generally fairly agnostic of feature selection or model choice. After running GridsearchCV on Gradient Boosting Regression, Random Forest Regression, and Neural Network Regressor, we all generally got an MAE that hovered around 2500 for our test set, the same results that we got for a LASSO regression. An exemplary version of our results can be shown below. Effectively, no model created (in this and many other cases), was able to capture the behavior of the model as the shares value rises significantly.

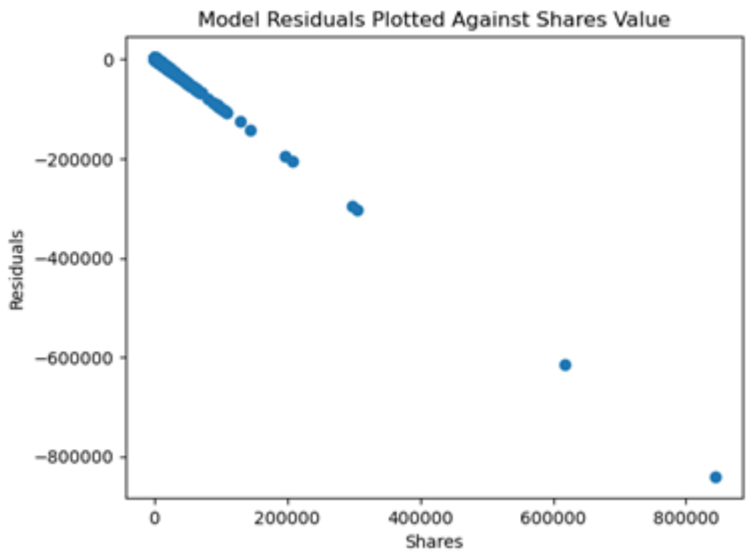
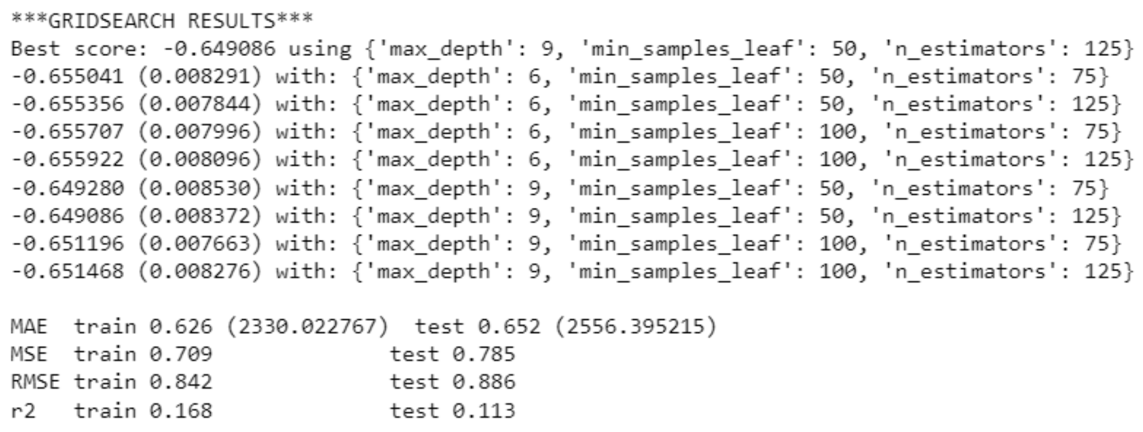


Figure 11 Residuals plotted against actual values.

The previous image shows the Random Forest residuals plotted against their actual shares value. Showing that the model is completely unable to have predictive value for larger target variable values.

There are several criticisms that we have of this approach we tried developing, and we hope to give some future directions for how this can be improved. Firstly, is the approach to start with the algorithm running on only one dimension, a first feature could possibly picked that does not necessarily provide the most value in understanding local behavior. A possible fix to this could involve feature bagging, or using a backward selection process instead (however this was not computationally possible for us in the group). Another issue, is how we have chosen to remove some of the worst values under the assumption, that a single variable's relationship to the target variable likely changes throughout the feature space. A better possible way to pull out variances that reflect regions where the explanatory variable is no longer helpful, could involve Otsu Thresholding(a method pulled from image processing, where we could separate the two categories of good and bad variances into two separate piles). In addition, we used Manhattan distance because we felt like it better reflects how distances should be measured when we expect linear relations between explanatory and target variables, and eased our computational load. However, with possible polynomial relationships, this might be a less accurate methodology.

These are the exact results we got with this approach, a GridSearchCV on RandomForest for the picked features.



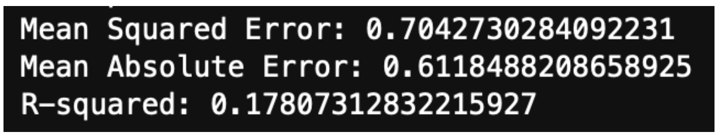
```
***GRIDSEARCH RESULTS***
Best score: -0.649086 using {'max_depth': 9, 'min_samples_leaf': 50, 'n_estimators': 125}
-0.655041 (0.008291) with: {'max_depth': 6, 'min_samples_leaf': 50, 'n_estimators': 75}
-0.655356 (0.007844) with: {'max_depth': 6, 'min_samples_leaf': 50, 'n_estimators': 125}
-0.655707 (0.007996) with: {'max_depth': 6, 'min_samples_leaf': 100, 'n_estimators': 75}
-0.655922 (0.008096) with: {'max_depth': 6, 'min_samples_leaf': 100, 'n_estimators': 125}
-0.649280 (0.008530) with: {'max_depth': 9, 'min_samples_leaf': 50, 'n_estimators': 75}
-0.649086 (0.008372) with: {'max_depth': 9, 'min_samples_leaf': 50, 'n_estimators': 125}
-0.651196 (0.007663) with: {'max_depth': 9, 'min_samples_leaf': 100, 'n_estimators': 75}
-0.651468 (0.008276) with: {'max_depth': 9, 'min_samples_leaf': 100, 'n_estimators': 125}

MAE  train 0.626 (2330.022767)  test 0.652 (2556.395215)
MSE  train 0.709                test 0.785
RMSE train 0.842                test 0.886
r2   train 0.168                test 0.113
```

Figure 12 Gridsearchcv results

Conclusion

Ultimately what we found after our several approaches, was that including all of the available information was most appropriate. We chose it because it clearly had the most consistently better values in terms of R-squared and Mean Absolute Error. Our results from this model can be seen here, in the test set:



```
Mean Squared Error: 0.7042730284092231
Mean Absolute Error: 0.6118488208658925
R-squared: 0.17807312832215927
```

Appendix

Approach N°3 – Frequent Itemsets Approach

This section endeavors to present an alternative methodology employed to address our problem.
In order to identify the most significant explanatory features, we leveraged the Apriori algorithm, a well-established technique within the domain of association rule learning, used for extracting frequent itemsets from large dataset. An itemset is considered as "frequent" if it meets a user-specified support threshold. For instance, if the support threshold is set to 0.5 (50%), a frequent itemset is defined as a set of items that occur together in at least 50% of all transactions in the database.

The idea was to find frequent itemsets between the discrete chunks of our explanatory variables, within split up discretized chunks of our target variables. Once we found the most important itemsets among the most frequent ones of our explanatory variables, which are basically the high explanatory variables, for each region of our target variable, we wanted to use these findings to look for statistically significant rules that link regions of our explanatory variables to regions of our target variable.
Finally, we wanted to use these identified association rules to select important features for modeling, based on having a small region that is highly associated to a small region of the target variable.

Discretization can make the model more interpretable, especially in this case where the target variable has a complex relationship with the predictors (categorical variables are often easier to interpret than continuous ones). In addition, we thought that discretization could help us capture non-linear relationships between the predictors and the target variable by allowing the model to treat different ranges of the target variable differently. Moreover, discretization would have helped us work with models, where the relationship between target variable and the explanatory variable changes based on which part of the feature-space you reside.

To streamline the modeling process and address specific data characteristics, we conducted a discretization procedure on our dataset.
Specifically, we performed a quantile-based discretization of the following:

- Explanatory variables into 3 equally-sized buckets.
- Target variable into 5 equally-sized buckets.

Following the execution of the Apriori algorithm, our objective was to filter itemsets within each of the 5 segments based on specific criteria, including:

- A length equal to 1.
- Support greater than or equal to 1/3.

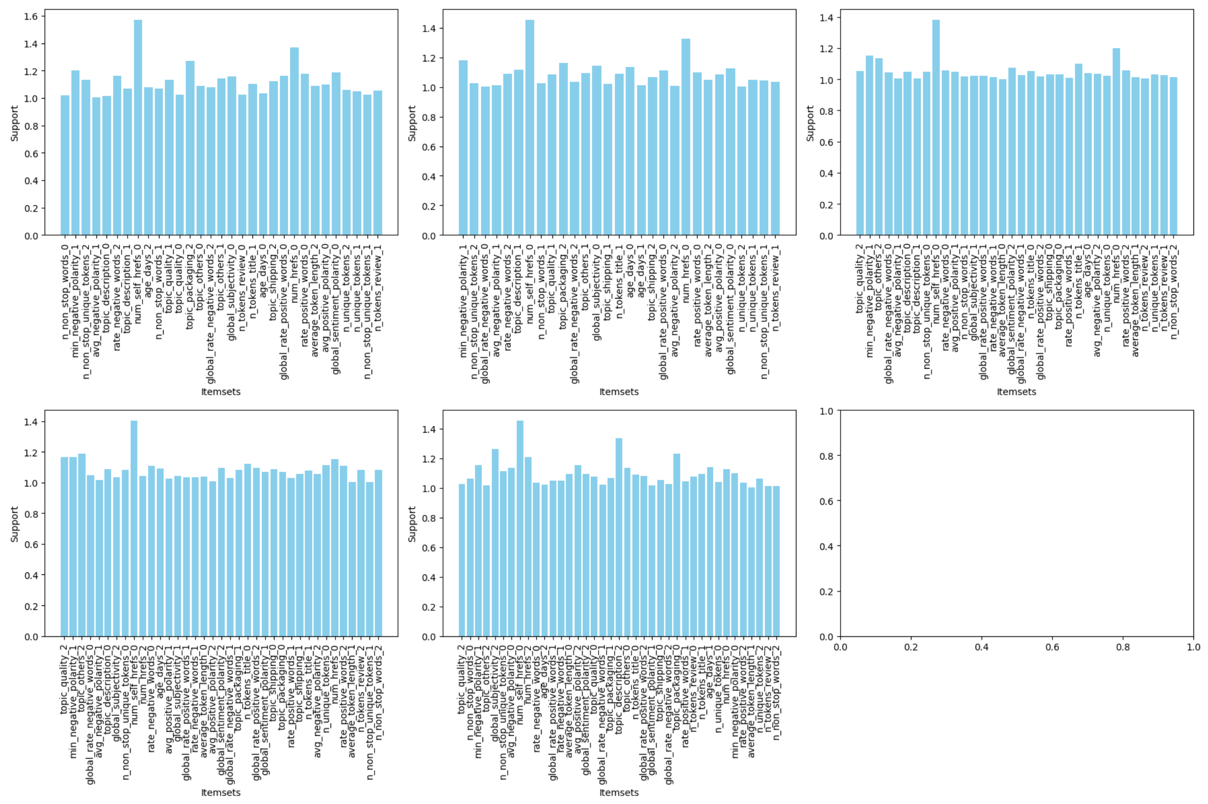


Figure 13

The previous image shows the support values for each itemsets for each of the five chunks.

We opted to filter itemsets with a support of at least 1/3 based on a strategic reasoning.
Our dataset was divided into five distinct parts, each representing a significant portion of the whole. Consequently, the support values derived from the Apriori algorithm for each itemset were merely 1/5 of the support observed in the original dataset.
If we look at the lift equation, when we divided the support by 5, and then divide again by 1/5 for the probability of the discrete chunk of our target variable, they canceled each other out.
As a result, for itemsets of length one, we determined their probability to be 1/3 due to our data splitting approach. Consequently, we observed that all itemsets of length one with support exceeding 1/3 exhibited a lift value greater than one.
Note that we decided not to deal with itemsets greater than 1 for computational reasons.

$$lift(A \rightarrow B) = \frac{support(A \cup B)}{support(A) \times support(B)}$$

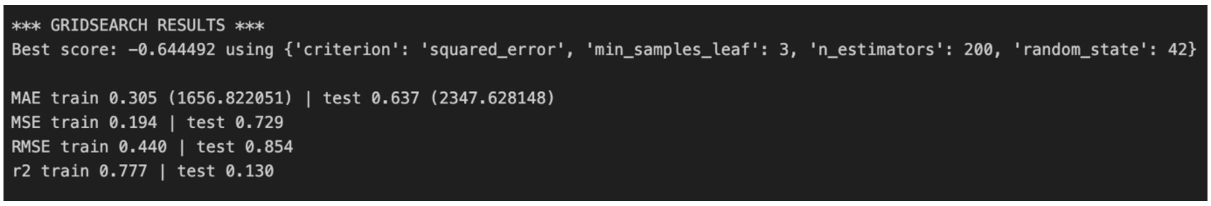
Where:
- A was the itemsets for the explanatory variables
- B was the itemsets for the target variables
- support(A) is the support (frequency) of itemset A (1/3).
- support(B) is the support of itemset B (1/5).

- support(A∪B) is the support of the combined itemset A∪B (1/5).

As a result of this procedure, the following features emerged as the most salient within our dataset:

'age_days','average_token_length','avg_negative_polarity','avg_positive_polarity','global_rate_negative_words','global_rate_positive_words','global_sentiment_polarity','global_subjectivity','min_negative_polarity','n_non_stop_unique_tokens','n_non_stop_words','n_tokens_review','n_tokens_title','n_unique_tokens','num_hrefs','num_self_hrefs','rate_negative_words','rate_positive_words','topic_description','topic_others','topic_packaging','topic_quality','topic_shipping'.

After identifying the salient features through this approach, as for the other approaches, we proceeded to implement several machine learning models utilizing these features. Regrettably, none of the models yielded satisfactory performance metrics. Among them, the Random Forest algorithm still exhibited the most promising results, although it had no statistical relevance, as you can see from the image:



```
*** GRIDSEARCH RESULTS ***
Best score: -0.644492 using {'criterion': 'squared_error', 'min_samples_leaf': 3, 'n_estimators': 200, 'random_state': 42}

MAE train 0.305 (1656.822051) | test 0.637 (2347.628148)
MSE train 0.194 | test 0.729
RMSE train 0.440 | test 0.854
r2 train 0.777 | test 0.130
```

Figure 14

Sadly, we had to abandon this idea because each lift value for the discrete chunks of our explanatory variables, within split up discretized chunks of our target variables, was higher than 1. Despite our efforts, we couldn't find a logical explanation for these results, and they did not improve our ability to explain the models. As a result, we decided to remove this strategy and include it in the appendix.