Sentiment Analysis of Amazon Fine Food Reviews

Abstract

The purpose of this project was to accurately predict positivity of sentiments of Amazon fine foods reviews with secondary importance attached to the correct classification of negative reviews. The analysis combined structured data (average score) with features extracted from unstructured data (average polarity, review text and summary) to predict the positivity of reviews. Logistic regression, random forest and gradient boosting machine were used to learn patterns from the features after tuning relevant hyper parameters. Gradient boosting machine was identified as the (single) best performing and applied on the test set, leading to accuracy of 0.835 (NIR = 0.775, $p - value \sim 0$ and F1 score of 0.90. Stacking was also explored, which led to a more stable model with similar performance on test set - accuracy = 0.835, F1 = 0.90.

Keywords: text information extraction, text encoding, logistic regression, random forest, gradient boosting

1 Introduction

Amazon fine foods reviews were in free text format. Amazon's business model revolves around customer loyalty, for which identification of positivity of reviews plays a central role. However, manual tagging of reviews consumes time. Statistical learning can be used to learn patterns from a small tagged set and can be extended to the remaining data set. The objective of this project was to use statistical learning to classify positive reviews accurately without misclassifying too many negative reviews.

The analysis combined text mining with product history to accurately predict the positivity of reviews. Statistical learning algorithms such as logistic regression, random forest and gradient boosting machine were used to learn patterns from relevant features.

2 Data Description

2.1 Background of Data Set

The dataset consisted of reviews of fine foods from Amazon. The data spanned a period of more than 10 years, including all 568,454 reviews up to October 2012. Data set included product and user information, ratings, and plain text review. It also included reviews from all other Amazon categories.

Source: Kaggle

2.2 Useful Variables

- ProductId: Unique identifier for the product (integer).
- Score: Rating between 1 and 5 (integer).
- Time: Timestamp by the exact seconds gap from 1970/01/01 00:00:00.
- Summary: Brief summary of the review provided by the user.
- Text: Review text.

2.2.1 Use of Variables

- Product score was rolled-up using `ProductId` and `Times` and the average rating of product before the current review was calculated.
- Text and summary were processed to obtain useful words (parts-of-speech and dependencies). The useful words were used to generate tf-idf.

2.3 Business Problem / Application

Amazon fine foods reviews includes product and user information, ratings, and a plain text review, which were of free text format. The main goal was to predict the sentiments of reviews on Amazon fine foods, either positive or negative, by focusing on bag of words features extracted from the text data. Importance was given to predict the positive sentiments correctly. A suitable use-case would be to reward people for providing positive reviews - therefore false positives were not penalized heavily.

2.4 Scientific Goal

Hypothesis: Sentiment of a review can be estimated as a (random) function of structured product information (past average rating of the product) and words used in the review. The true underlying model was assued as unknown.

Business Goal: To identify customers who have positive outlook on the food products sold at Amazon and to enroll them in loyalty program by providing coupons / discounts.

Technical Goal: To predict positivity of reviews correctly without misclassifying too many negative reviews as positive.

3 Literature Review

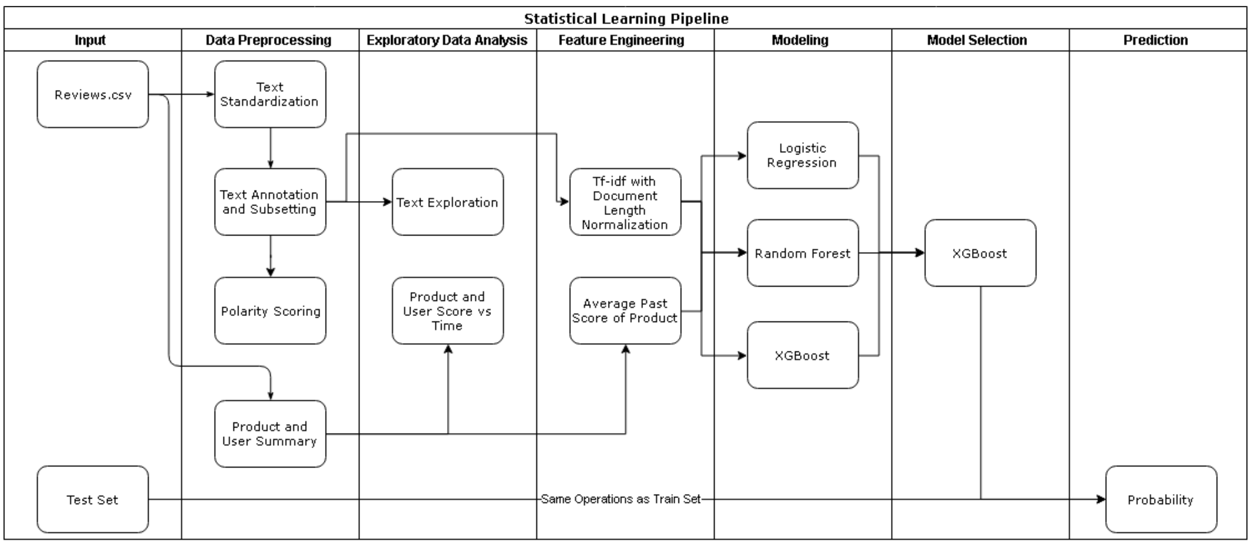
The data set was a part of Stanford Network Analysis Project. Several kernels in Kaggle have performed different types of analyses on the data set. The kernel by Shashank explained several basic text cleaning steps and techniques for encoding text data. Bag of words, bigram/n-gram, tf-idf and word2vec were mentioned in the kernel.

The kernel by Anghel (2017) which was forked from Guillaume Payen focused on predicting helpfulness of the review using words used in the review and summary. The kernel selected Score, Summary and engineered other features from the dataset. A review was considered helpful if $\frac{\text{HelpfulnessNumerator}}{\text{HelpfulnessDenominator}} > 0.8$. Features were extracted from text by creating term document matrix. Three learning algorithms - Multinomial naive Bayes, bernoulli naive Bayes and logistic regression - were used for modeling Helpfulness. Logistic regression outperformed the other 2 models with the highest AUC (0.91).

The kernel by Shashank (2018) performed data preprocessing, but did not perform statistical learning. The kernel by Ligade (2018) used spaCy to clean the text and performed feature extraction using sense2vec for predicting the sentiment of the review. The kernel used spaCy's internal neural network based text classification model to train and predict the sentiment of the review, but did not evaluate or compare models. The model achieved recall of 0.859 and F-1 score of 0.87.

The kernel by DLao (2018) applied logistic regression to analyze the sentiment. The kernel used upvote model for predicting sentiment, which computed percentage of upvote in terms of helpfulness. Unigram and bigram bag of words model were used for text representation. The model achieved 94% accuracy on training set using unigram-bigram features.

4 Statistical Learning Pipeline

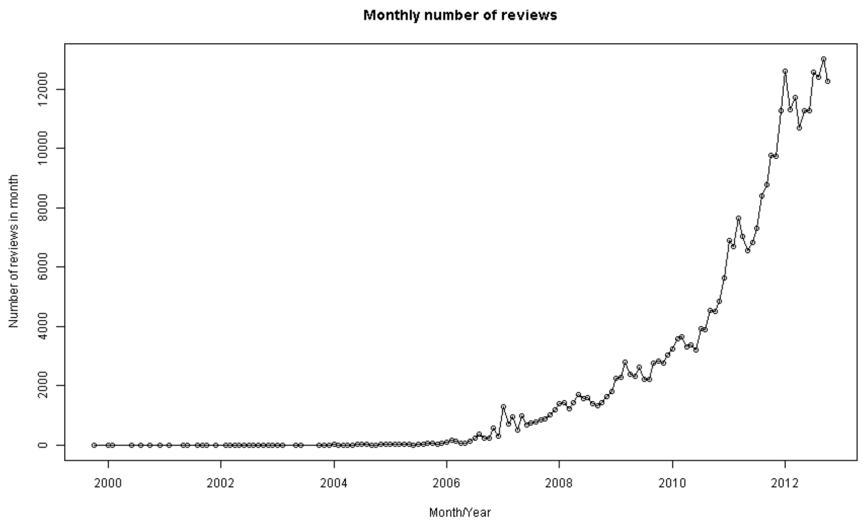


5 Exploratory Data Analysis

5.1 Exploration of Product Rating

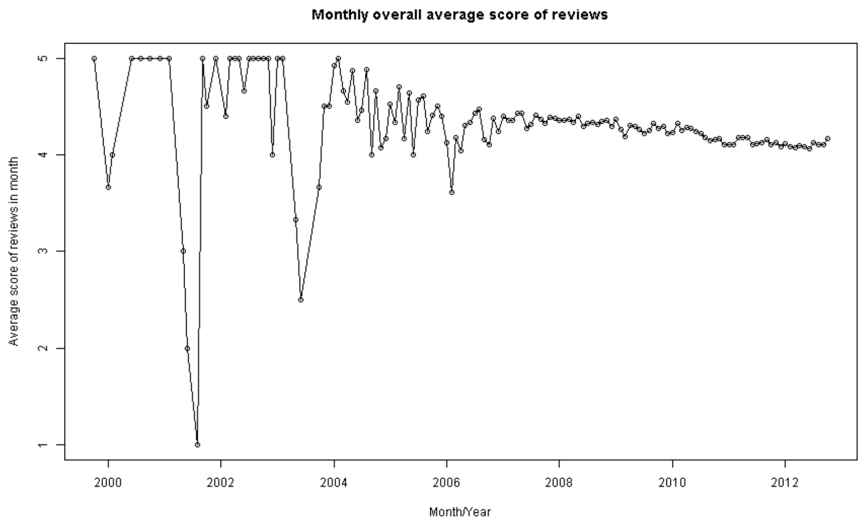
5.1.1 Temporal Evaluation of Review Count

The earliest recorded product rating was given in 1999 and the latest recorded rating was given in October 2012. The number of reviews has increased over time as shown below:



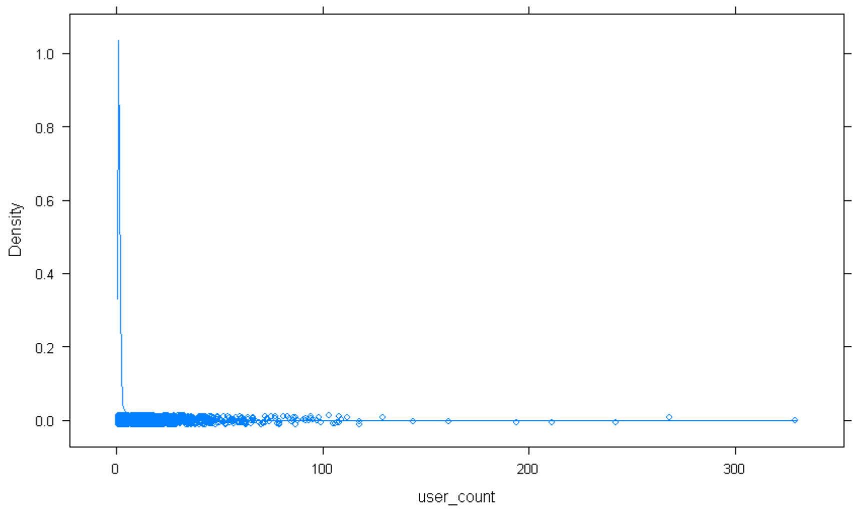
There was a decreasing pattern in the overall average score. There was no clear seasonal pattern in the average score. Therefore, time can be used as a predictor. However, to introduce within-product variations, the past average score of product was used instead of time.

5.1.2 Temporal Evaluation of Average Score



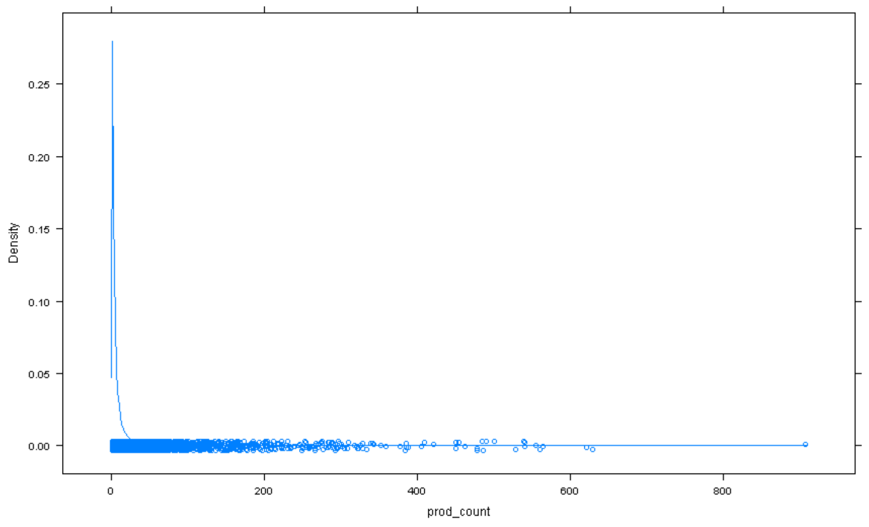
5.1.3 Distribution of User Review Count

It was observed that the distribution of number of ratings by a user was extremely skewed. Also, since the number of distinct users ∼ number of reviews, the idea of using user history for predicting sentiment was dropped.



5.1.4 Distribution of Product Review Count

It was observed that the distribution of number of ratings for a product was also skewed, but not as much as that of user. Therefore, the variable was not dropped from the analysis.



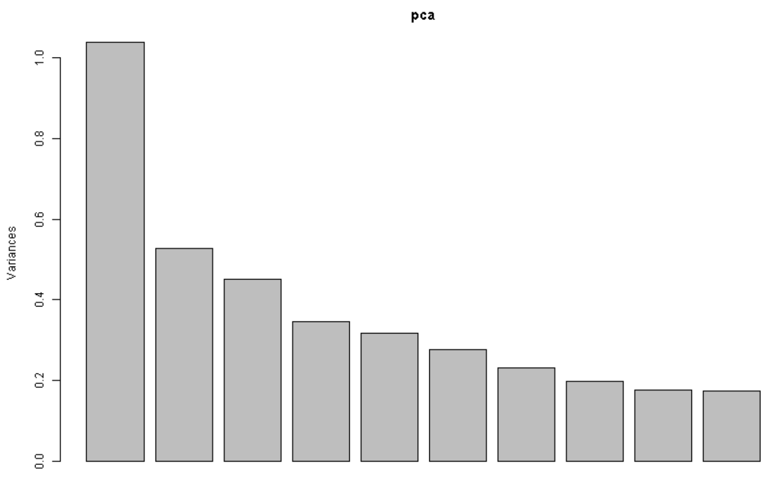
5.2 Correlation

It was observed that the correlation between features of `Summary` such as average, minimum and maximum polarity was very high. There were no other groups of that showed similar behavior:

	summary_avg_polarity	summary_max_polarity
summary_avg_polarity	1	0.9489
summary_max_polarity	0.9489	1
summary_min_polarity	0.9472	0.8015
summary_sd_polarity	0.001521	0.3124

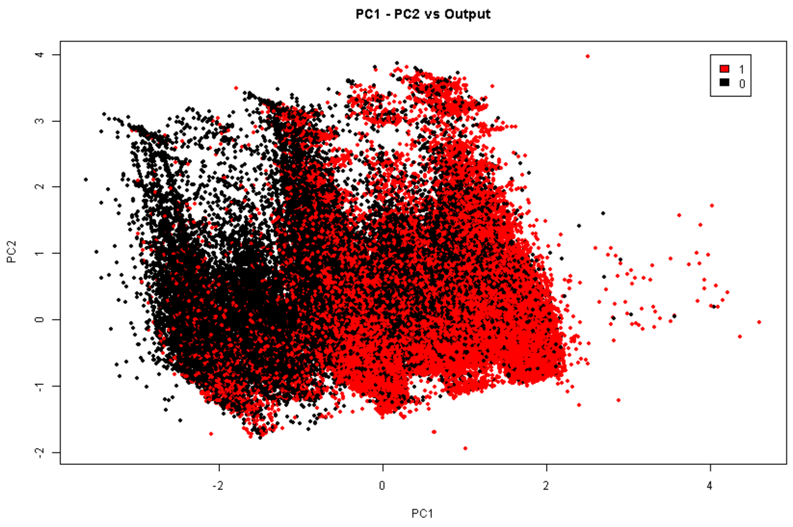
5.3 Visualization of Principal Components

5.3.1 Variance of Principal Components



5.3.2 Outcome vs PC1-PC2

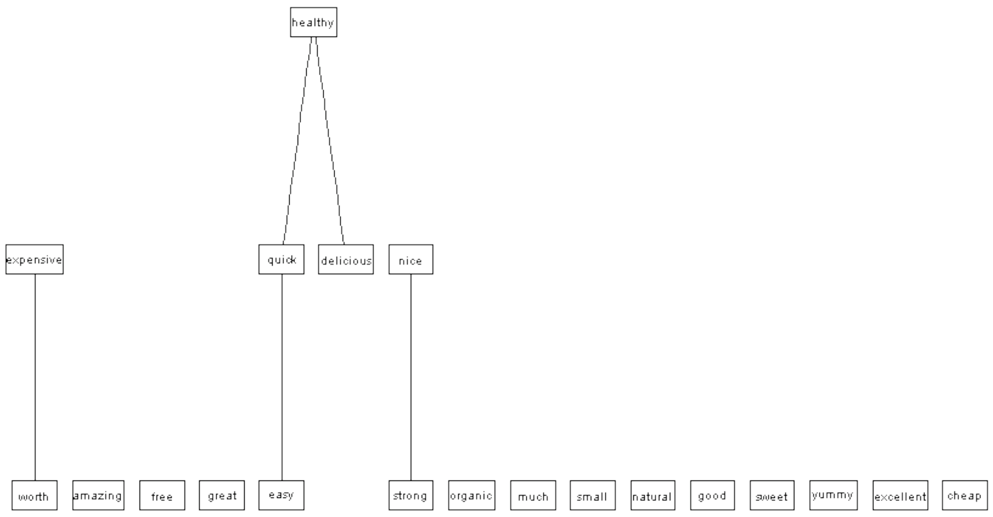
Due to high collinearity, PCA was applied on the data set and the output was visualized with PC1 and PC2:



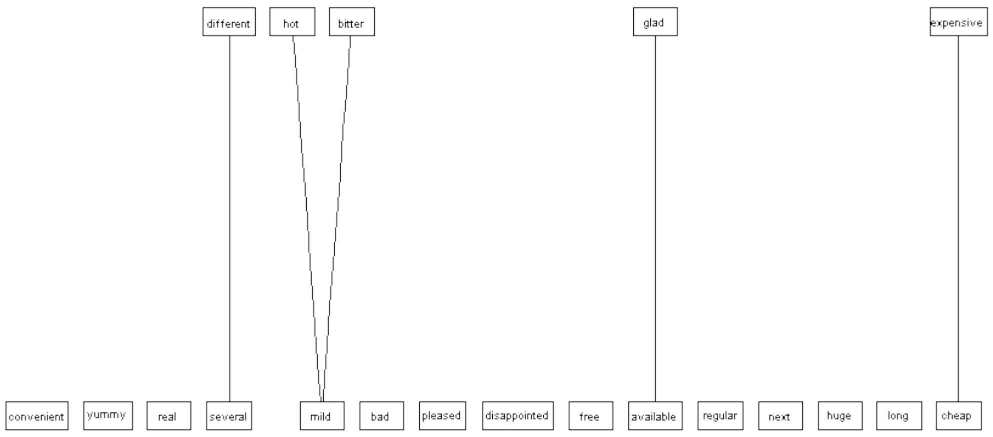
It was observed that the principal components contain some information about the label, but PCA was ignored due to lack of time.

5.4 Exploration of Text

5.4.1 Word correlation plot for Summary

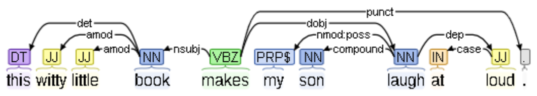


5.4.2 Word correlation plot for Review



5.4.3 Visualization of Dependencies

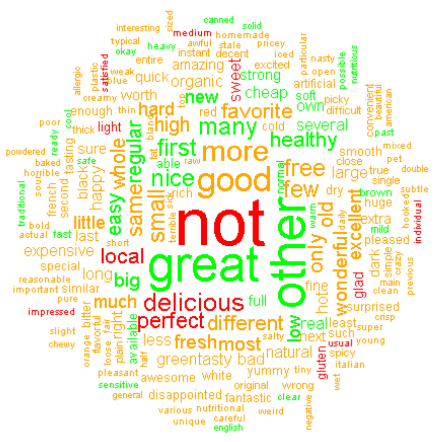
Parts-of-speech tagging and dependency parsing led to useful words. A sample visualization of dependency parsing annotated with parts-of-speech can be found below:



det
amod
amod
nsubj
punct
dobj
nmod:poss
compound
dep
case
DT JJ JJ NN VBZ PRP$ NN NN IN JJ .
this witty little book makes my son laugh at loud .

Manual examination of parts-of-speech and dependency parsing led to the conclusion that not all parts of speech and dependencies were useful. Only the words with relevant parts-of-speech / sentence dependency were chosen and annotated with the outcome (1 for `Score` ≥ 4 and 0 for `Score` ≤ 4).

5.4.4 Visualization of Filtered Words



Wordcloud Key:

- Green: Percentage of `1` >= 70%

- Orange: 70% > Percentage of `1` > 30%
- Red: Percentage of `1` <= 30%

The distribution of colors was found to be skewed. Also, the wordcloud showed that some of the features for negative (`Score` < 4) were positive words (Eg: perfect, delicious, safisfied, etc.). Therefore, capturing negativity of reviews was difficult:

color	count
green	9831
orange	6746
red	2768

6 Analysis

6.1 Data and Feature Engineering

6.1.1 Review Deduplication

The original data set contains 568,454 rows with 74,258 distinct products. The vocabulary size of unprocessed text was 119,242. It was observed that some of the rows had the same review content (`Time`, `Text` and `Summary` columns), but different `ProductId` column. This led to the conclusion that product IDs were duplicated over time. A simple review deduplication scheme was used, leading to 393,579 rows and 67,477 distinct products.

6.1.2 Product Score

Average rating of a product was an important variable for predicting sentiment as it captured trends in performance of a product before the current review. It was created by calculating cumulative sum of `Score` by `ProductId` and `Time` ordered by `Time`. Since first row (sorted by time) of each product will not have average Score, a further 67,477 rows were removed, leading to 326,102 rows.

6.1.3 Text Preprocessing

Amazon fine foods text reviews were extremely unclean. Standard preprocessing of unclean text caused unexpected changes to clean text. Therefore, only the following steps were applied:

- Converted to ASCII from Latin-1.
- Converted to lowercase.
- Removed HTML tags, additional special characters, etc.
- Added space between hyphenated words and sentences that were not separated by space after terminating character ('!', '.', ';', etc.).

6.1.4 Text Annotation and Filtering

During data exploration it was observed that not all words were important in predicting the sentiment of a review. Manual examination of annotated text revealed only the following were useful:

- Dependencies: "amod", "neg" and "acomp".
- Parts-of-speech: "ADJ" which were not captured by dependencies.

6.1.5 Text Mining Vocabulary

- Document frequency: Number of distinct reviews in which a word occured.
- Term frequency: Total number of occurrences of a word in a review.
- Tf-idf: A function of the form $F(TF) * G(DF)$, with sub-linear functions `F` and `G` of term frequency and document frequency respectively.
- Document length normalization: Normalization factor to adjust the tf-idf by a (sub-linear) function of the document length.

6.1.6 Text Features

Filtered words obtained after annotation were used to create tf-idf features with document length normalization. Words were considered for column creation in tf-idf if they satisfy the following conditions:

- `Summary`: Document frequency ≥ 1000
- `Text`: Document frequency ≥ 3000

If a word was present in `Summary` and `Text`, the average of tf-idf from `Summary` and `Text` was considered.

Along with tf-idf features, a pretrained polarity scorer (`qdap` package) was used on the useful text to give weightage to individual words in the review. Aggregates such as minimum, average, maximum and standard deviation of review polarity were calculated and used as features. The algorithm gave fixed positive score to each positive word and fixed negative score to each negative word. Finally, it aggregated the score and normalized the overall score by the inverse of sentence length. This process was performed on `Summary` and `Text`.

Finally, the data set was split into train (49%), validation (21%) and test (30%) sets. The (blind) test set was kept aside and was used only for evaluating the final model.

6.2 Statistical Learning

6.2.1 Model Selection

For the given business problem, the cost of incorrectly identifying a negative review as positive may not be very high. However, the cost of incorrectly identifying a positive review as negative may be high. Therefore, the primary objective was to maximize the ability of the model to predict positive reviews. At the same time, the secondary objective was to reduce negative reviews being misclassified as positive. Both the objectives were simultaneously achieved by tracking the `F-1 score`, the harmonic mean of `Precision` and `Recall`.

$$F_1 = \frac{2 \times P \times R}{P + R}$$

For each model, F_1 score was used to obtain the best threshold and the performance was evaluated on the train and validation sets using the chosen threshold. The model with the highest F_1 score on validation set was used for prediction on the test set.

6.2.2 Hyperparameter Tuning

Binary logistic regression, random forest and (extreme) gradient boosting were used to build models on the training set. Hyperparameters were tuned using the validation set:

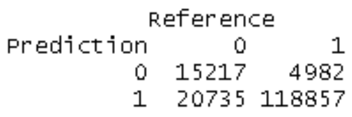
- For random forest, the a grid of 3 `ntree` x 4 `mtry` x 5 `nodesize` values were chosen, leading to 60 distinct combinations of hyperparameters. The resulting model was evaluated on the validation set using the model selection criteria.

- For xgboost variant of gradient boosting, the a grid of 4 `maxdepth` values were chosen as hyperparameter. For each maxdepth, the `nrounds` parameter was tuned using the `logloss` (default) on validation set. A tolerance of 5 rounds was used to allow the model to escape error plateaus.

7 Results and Conclusion

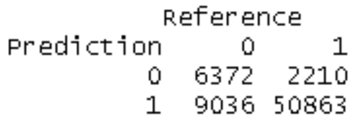
7.1 Confusion Matrix

7.1.1 Train Set



```
          Reference
Prediction     0      1
         0 15217   4982
         1 20735 118857
```

7.1.2 Validation Set



```
          Reference
Prediction     0     1
         0  6372  2210
         1  9036 50863
```

7.2 Comparison

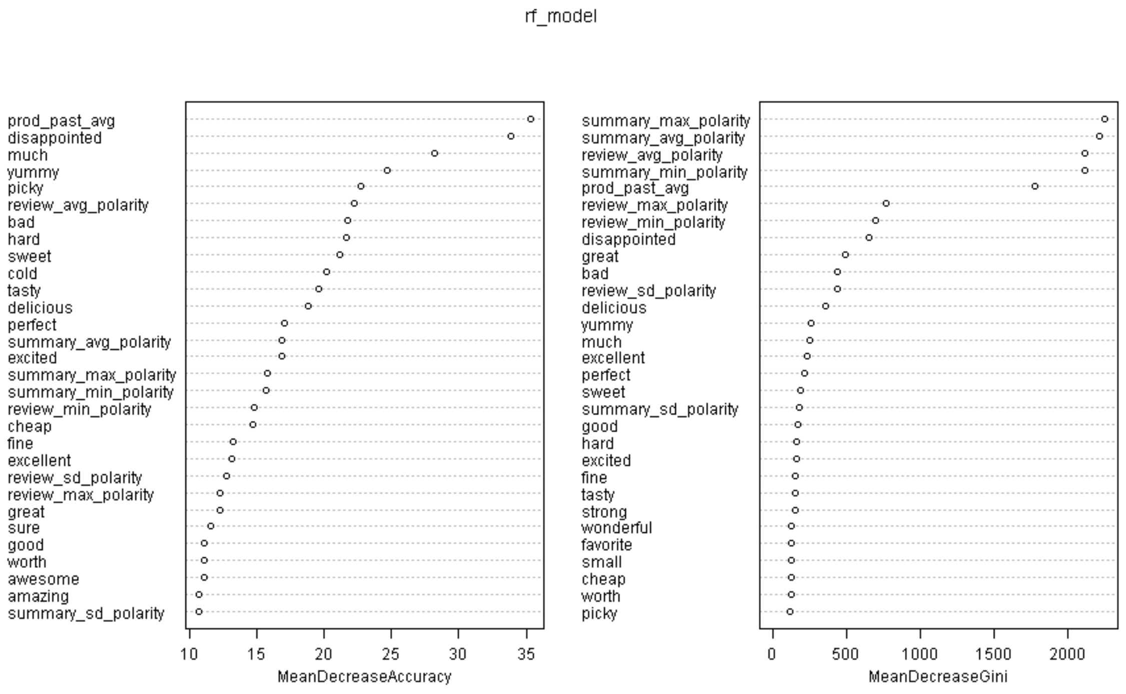
Model	Tr.Acc	Tr.F1	Tr.AUC	Val.Acc	Val.F1	Val.AUC
Logistic Regression	0.827	0.896	0.862	0.828	0.896	0.843
Random Forest	0.855	0.911	0.869	0.835	0.898	0.803
XGBoost	0.839	0.902	0.862	0.836	0.9	0.856

Note: Tr: Train, Val: Validation

We observed that the XGBoost model captures more number of positive reviews than random forest, but fewer positive reviews than logistic regression. However, the performance of negative predictions of XGBoost was better than that of logistic regression. Therefore, XGBoost model was chosen as solution to the given business/research problem.

7.3 Interpretation

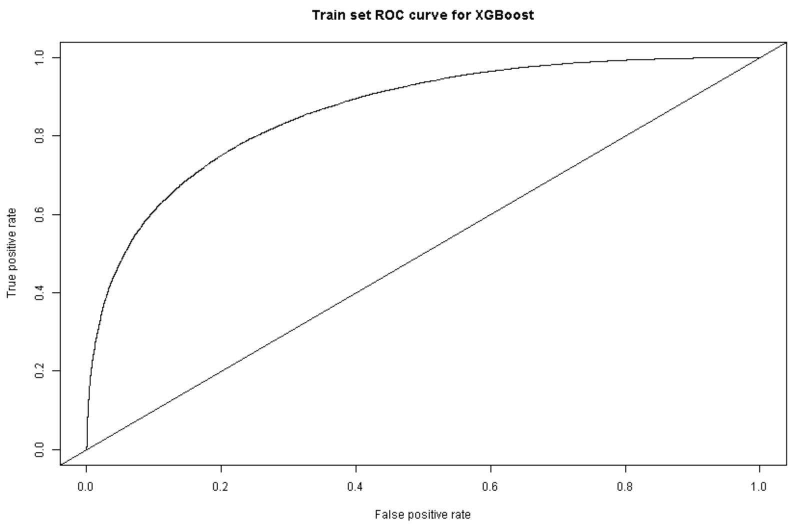
7.3.1 Random Forest Variable Importance



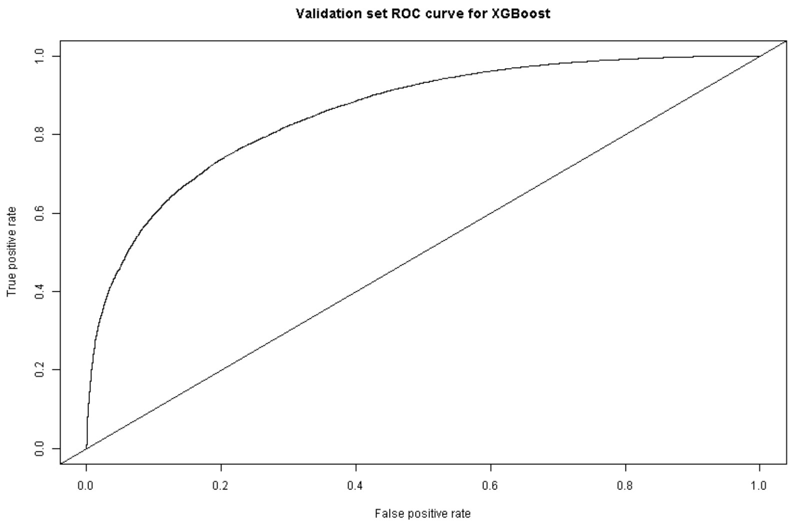
It was observed that that the past average rating of the product strongly affects the positivity of review. The hypothesis that the polarity is a function of words used in the review / summary and polarity was validated. It was also noted that the reviews with high variation in polarity are difficult to classify.

7.4 ROC Curve of Chosen Model: XGBoost

7.4.1 Train Set



7.4.2 Validation Set



7.5 Performance on Test Set

The hold out test set was not seen by any of the models - none of the models were tuned to perform well on the test set. Therefore, XGBoost - the best chosen model was evaluated on the test set. The following values were observed:

- Threshold: 0.479
- Accuracy: 0.835
- F_1 score: 0.899

The performance closely resembles the performance on validation set.

7.6 Limitations / Scope for Improvement

- The hold out test set was chosen from the same population as the train and validation sets. This is unlikely to be the case in real situations. Therefore, test performance similar to validation performance was not surprising.
- Text cleaning was extremely difficult. Coming up with generalized set of data-specific rules for text cleaning takes months (with only incremental improvement in accuracy). Therefore, in the interest of time, we concluded the text cleaning step once we observed that the parts-of-speech tagger and dependency parser were working reasonably well. Also, accurately capturing bigrams such as "not satisfied" requires more text processing steps.
- Product lifecycle was not modeled due to lack of time. Estimated lifecycle stage can be a useful feature.
- User cold start problem was note dealt with. Dealing with this problem will provide additional (estimated) features related to the user, which may be a useful in predicting positivity of the review.
- Product cold start problem was ignored because it led to ˜ 15% reduction in data, which was acceptable. Multiple imputation could have been performed on the 67,477 rows. Instead, these rows were dropped because extending a model across different lifecycle stages may lead to inaccuracies.
- Stacked ensemble was built on validation set by using predictions of each model as input and validation labels as output. However, due to lack of time, other models were not explored for the first and second layers of stack. This led to the following results:
  - Train accuracy, F_1 score: 0.835, 0.900
  - Validation accuracy, F_1 score: 0.835, 0.900

– Test accuracy, F_1 score: 0.834, 0.899

It was observed that the stacked ensemble has very stable performance. However, resampling was not performed to ensure stability due to lack of (run) time.

8 Appendix

8.1 Code Snippets

8.1.1 Text Preprocessing

```
clean_text <- function(text_reviews) {
  ...
  text_reviews <- gsub(text_reviews, pattern = "([a-z])\\.([a-z])",
                       replacement = "\\1. \\2")
  return(text_reviews)
}


extract_phrases <- function(dependencies, tokens) {
  ...
  reqd_tokens <- do.call(rbind, lapply(1:nrow(df), function(i) {
    dep_row <- df[i,]
    toks <- tokens[tokens$id == dep_row[1, 1] & tokens$sid == dep_row[1, 2], ]
    dep_row <- gsub(as.character(dep_row), pattern = "^ ", replacement = "")
    res <- parse_amod_nsubj_neg(dep_row, tokens)
    return(res)
  }))
  ...
  reqd_tokens2 <-
    do.call(rbind,lapply(1:nrow(dep_subset[["neg"]]), function(i) {
      dep_row <- gsub(dep_subset[["neg"]][i, ], pattern = "^ ",
      replacement = "")
      dep_row <- matrix(dep_row, nrow = 1)
      colnames(dep_row) <- colnames(dep_subset[["neg"]])
      if("acomp" %in% names(dep_subset)) {
      dep_row1 <- merge(dep_row, dep_subset[["acomp"]],
      by = c("id", "sid", "tid"))
      if(nrow(dep_row1) > 0) {
        res <- parse_neg_acomp(dep_row1, tokens)
        } else {
        res <- parse_amod_nsubj_neg(dep_row, tokens)
        }
      } else { res <- parse_amod_nsubj_neg(dep_row, tokens)
      }
      return(res)
    }))
  ...
  reqd_tokens <- rbind(reqd_tokens, reqd_tokens2)
  reqd_tokens <- data.frame(reqd_tokens, stringsAsFactors = F)
  return(reqd_tokens)
}
```

8.1.2 Text Features

```
get_tfidf <- function(text, thr = 20, norm = F) {
  ...
  summary_tfidf <- TermDocumentMatrix(corpus, control = list(
    weighting = function(x) weightTfIdf(x, normalize = norm),
    stopwords = T, bounds = list(global = c(thr, Inf))))
  ...
}
```

8.1.3 Product Average Rating (Past)

```
...
df <- reviews %>% dplyr::group_by(ProductId, Time) %>%
  dplyr::arrange(ProductId, Time) %>%
  dplyr::mutate(total_score = cumsum(Score), row_num = row_number())
```

```
...
user_summary_df$user_past_avg <-
  (user_summary_df$total_score - user_summary_df$Score)/
  (user_summary_df$row_num - 1)
...
```

8.1.4 Modeling

```
# Logistic Regression
...
lr_model <- glm(Output ~ ., data = train, family = binomial)
...

# Random Forest
...
rf_model <- randomForest(
  Output ~ ., data = train, method = "rf", ntree = best_params["ntree"],
  mtry = best_params["mtry"], nodesize = best_params["nodesize"],
  importance = T, do.trace = T)
...

# XGBoost
...
xgb_model <- xgb.cv(data = x, nrounds = 500, early_stopping_rounds = 5,
                    folds = folds, params = list(
                      max_depth = max_depth, objective = "binary:logistic"))
...
```

8.1.5 Evaluation

```
...
test_pred <- predict(best_model, x3, "prob")
test_pred01 <- ifelse(test_pred > thr, 1, 0)
print(confusionMatrix(as.factor(test_pred01), test$Output))
```

. . .

References

Anghel, E. (2017), ‘Predicting sentiment and helpfulness’, *Kaggle* .

DLao (2018), ‘Amazon fine food review - sentiment analysis’, *Kaggle* .

Ligade, P. (2018), ‘Text classification using spacy’, *Kaggle* .

Shashank (2018), ‘Text processing using python’, *Kaggle* .