Understanding Product Reviews

Team Chernoff

Feb 15th, 2023

In the **next 15 minutes** we will:

1. Explore **what makes a review helpful** and highlight **interesting patterns** in the reviews
2. Illustrate the **process of building an ML model** to predict the helpfulness of a review
3. Evaluate **the cost and resources required** to develop a sophisticated solution

Amazon Reviews: Overview

Motivation: Amazon customer reviews play an important role in influencing consumer purchasing decisions.

Objective: build a machine learning model to predict if an Amazon customer review will be helpful or not based on its non-text and text features.

Resources: a dataset with 3M+ labels marked as **helpful** and **not helpful**.

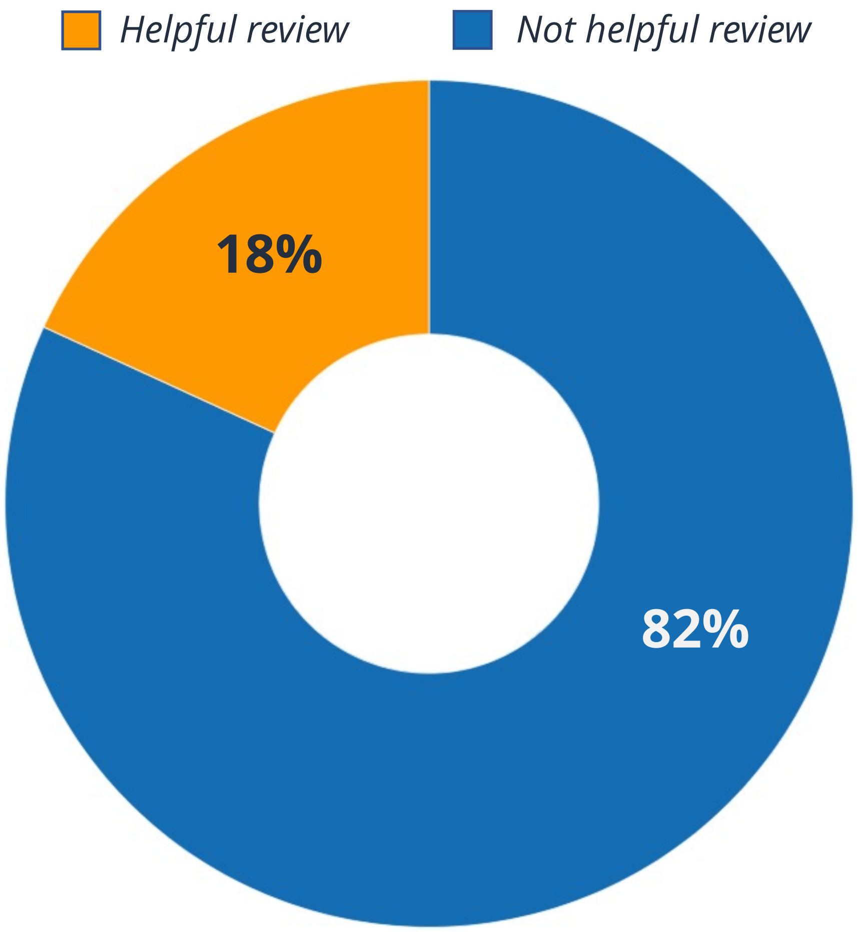


Chart 1: The proportion of helpful reviews

Does length matter?

Our data shows that **helpful reviews are longer**, with more characters typed by reviewers.

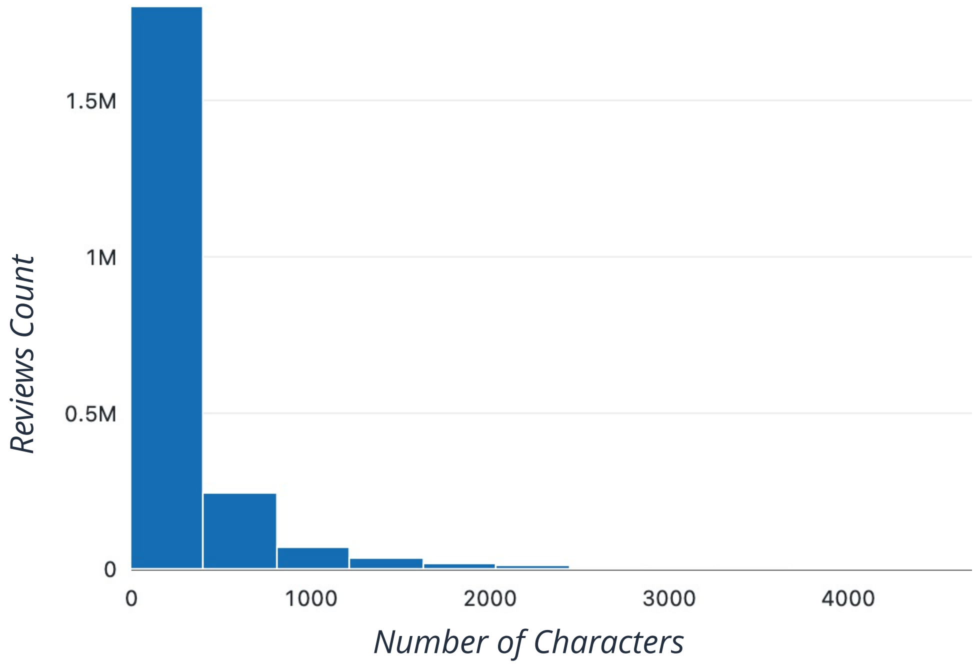


*Chart 2: The distribution of character count in **not helpful** reviews*

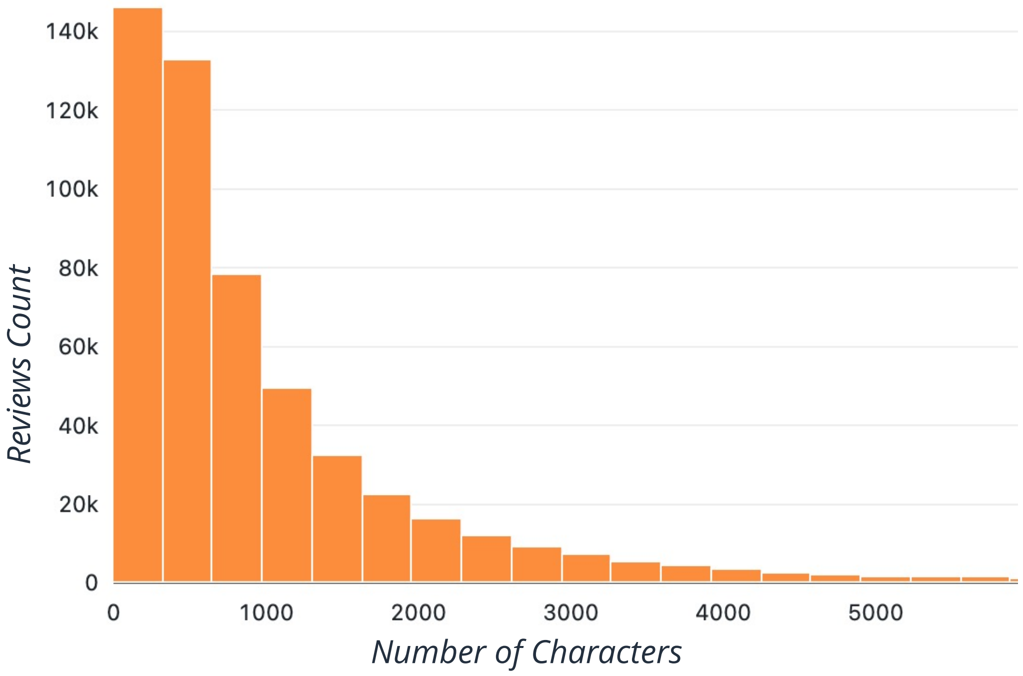


*Chart 3: The distribution of character count in **helpful** reviews*

The Power of a Good Summary

Reviews with ***"<number> stars"* summary** provide very limited information about the product.

One-fifth of reviews had such a summary, **99% out of which were not helpful**

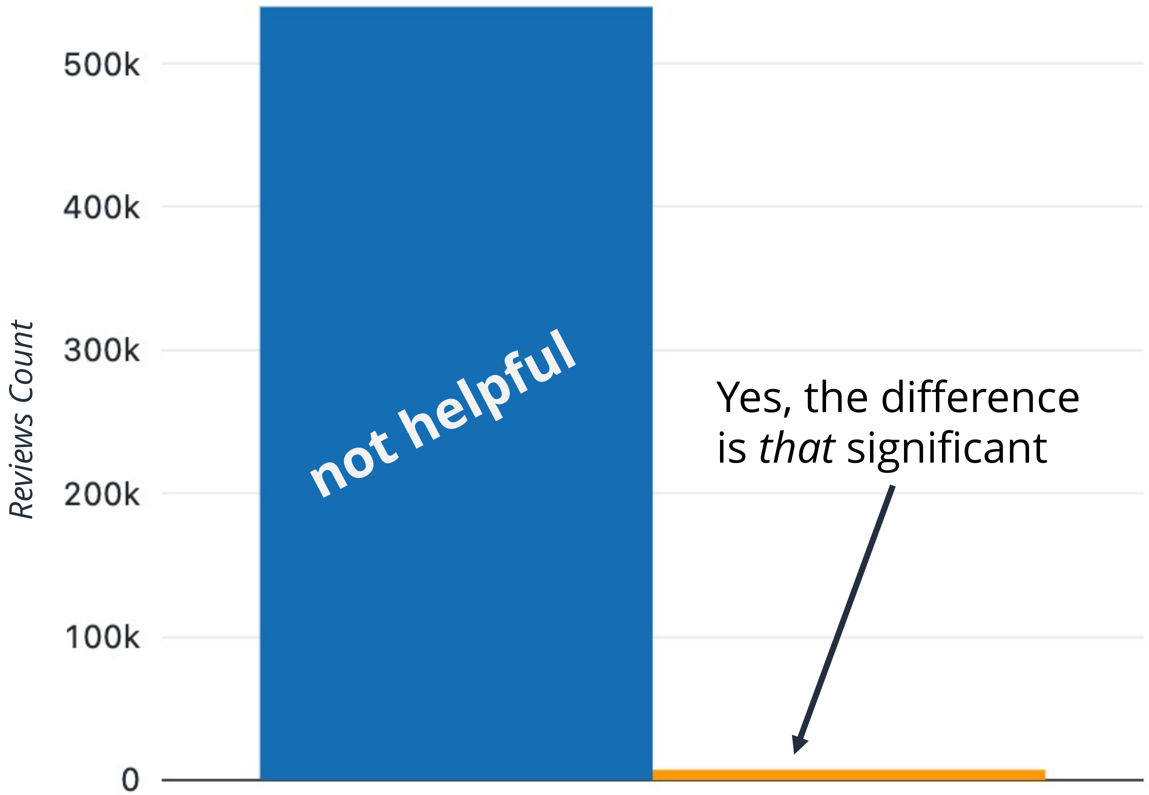
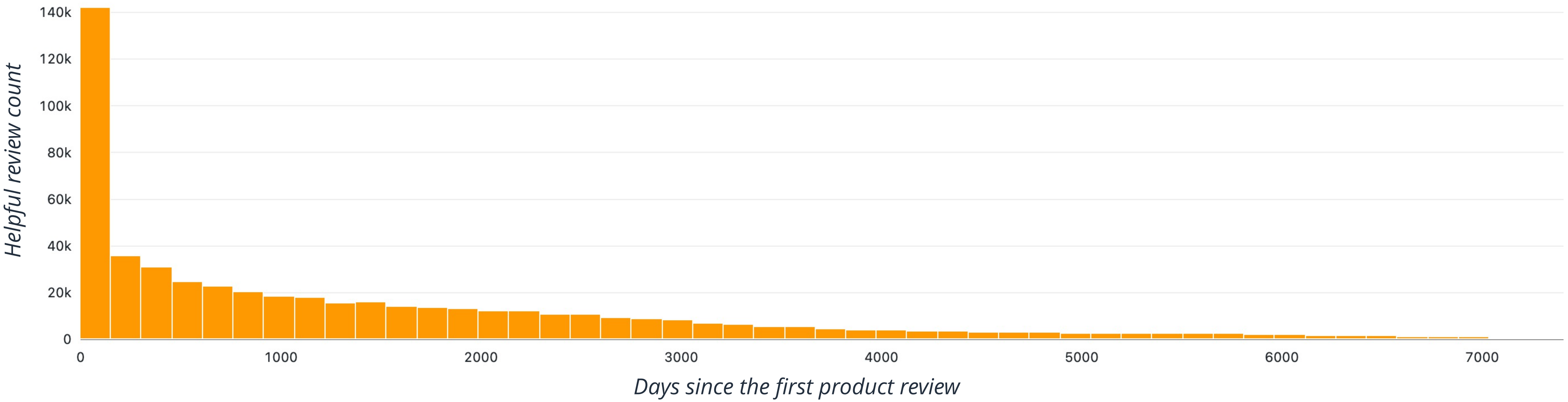


Chart 4: Difference between reviews with a generic summary

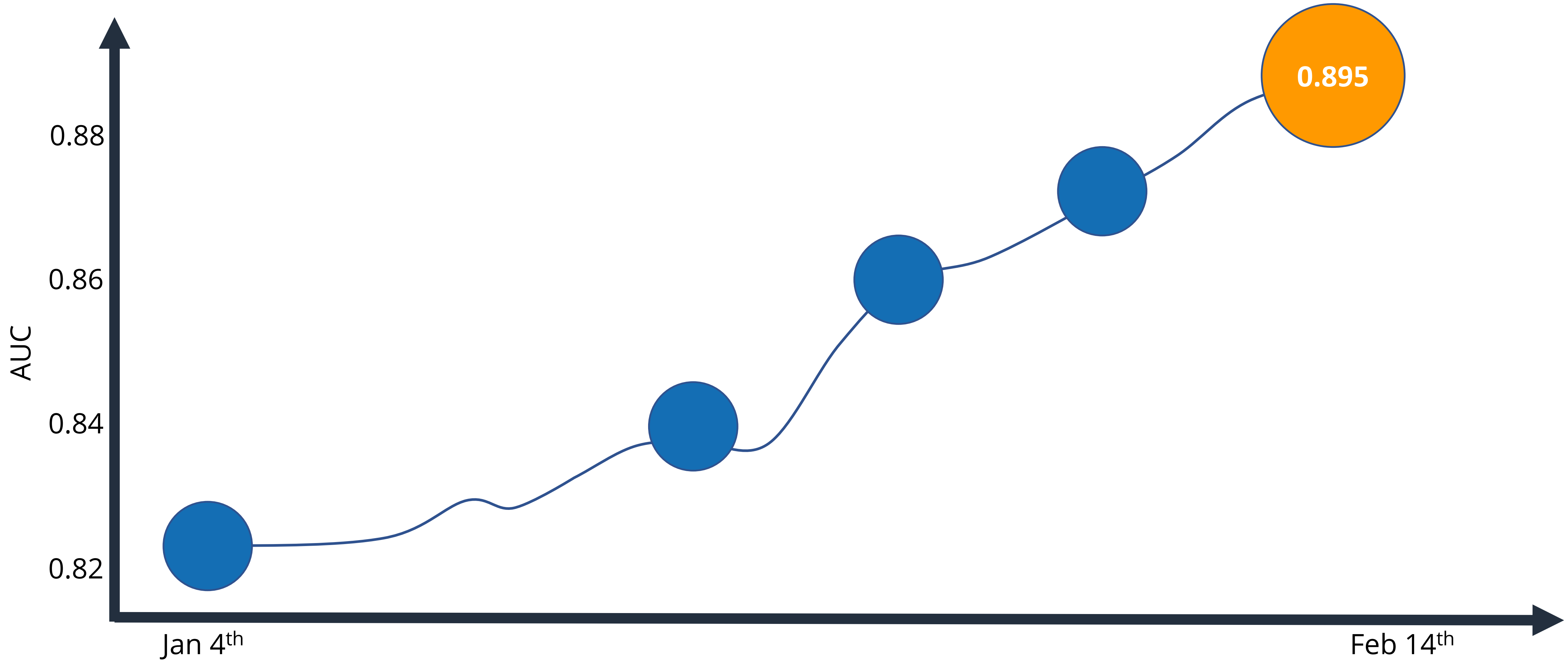
Timing is Everything

As the saying goes, "The early bird gets the worm”, and it also gets a **helpful review**! Over time, reviews are less helpful.

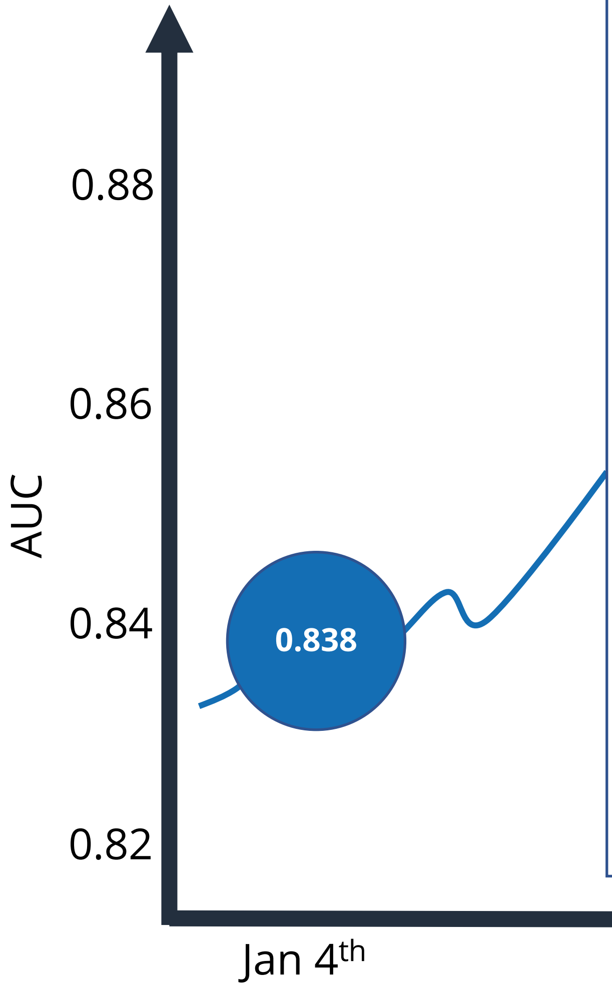


32% of helpful reviews are written **within the first year after the first product review** was posted.

Maximizing Model Performance



Clean Start



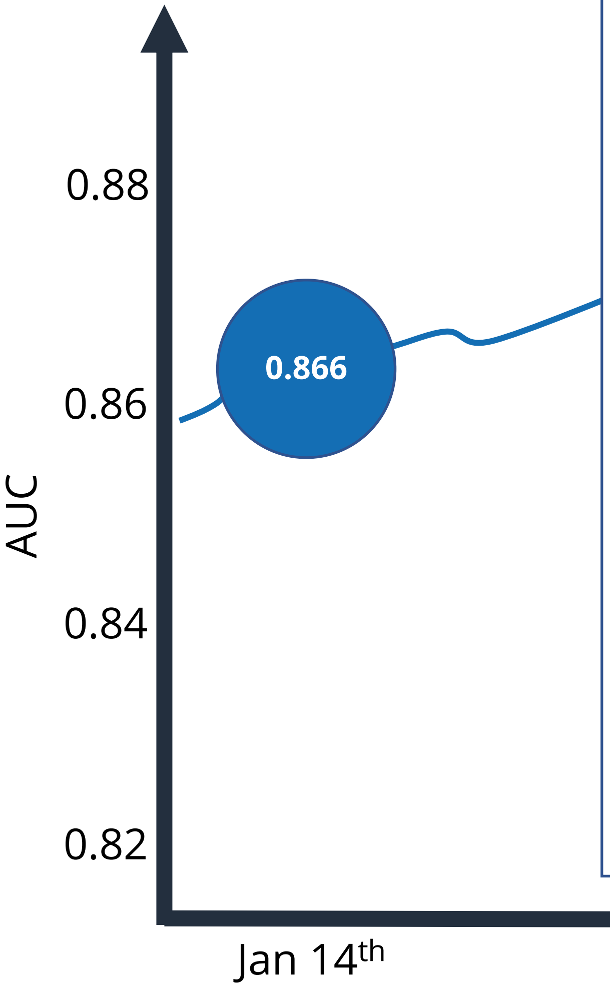
We first focused on **cleaning and preprocessing** the data. Also, we kicked off feature engineering with **the review elements counts**.

Feature engineering:
- TFIDF on review text,
- Character, words, sentences counts
- **Language detection**

Cleaning:
- Dropping **duplicates**, fixing **null values**
- Cleaning summary (HTTP links, punctuation)
- Cleaning review text (HTML content, links, punctuation)

Model: Logistic Regression

Feature Engineering with Timestamps



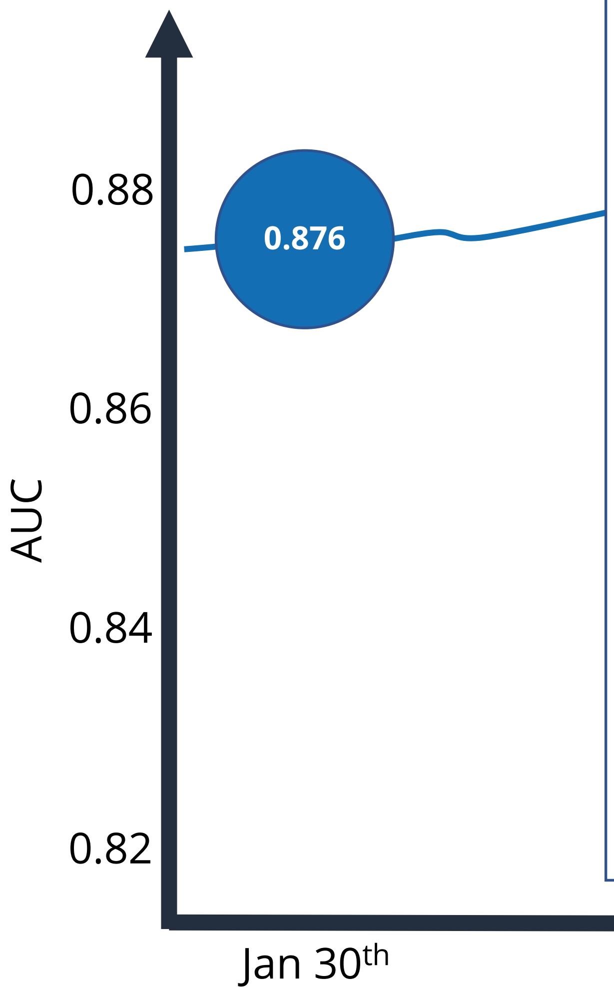
After that, we delved into **feature engineering associated with review date and time**. Additionally, we added **more text preprocessing** for both summary and text reviews.

Feature engineering:
- **UNIX time**: day, year, month, season, weekend
- **Reference table**: days passed since the first product review

Text preprocessing:
- **POS extraction** and count
- Sentiment score
- Generic summary and text reviews

Model: Logistic Regression, GBT Classifier, LGBM

The Key to Unlocking Performance



For the next push, we looked at the **reviewer and product features** that we have not explored yet. We also tried oversampling, **hyper-tuning** and **feature selection** techniques.

Feature engineering:

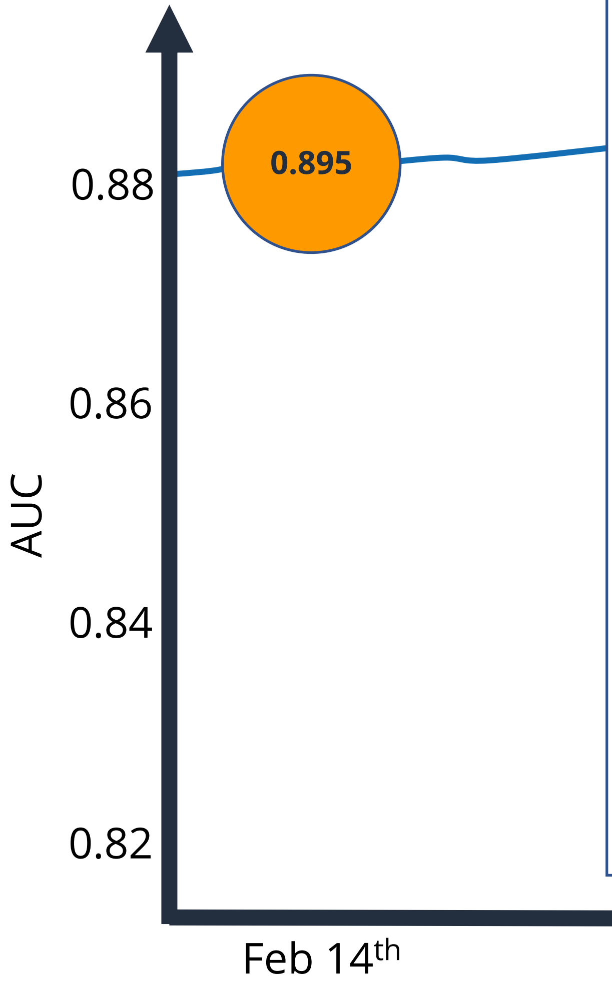
- **Reviewer**: anonymous reviewer, active reviewer
- **Product**: product rating, product "popularity", **book identifier**

Text preprocessing:

- Spellchecker, **extended stop words**, NGrams, keyboard mash
- Stemming, lemmatization

Model: GBT Classifier, XGBoost

The Final Push



For the very last step, we **tuned TFIDF and classifier parameters** and fixed features on our backlog until we reached the desired result.

Final model:

- 27 new features
- 25,000 tokens for review text
- 12,000 tokens for summary
- GBT Classifier with a max depth of 8 over 25 iterations

The Cost Breakdown

Invoice

Description	Unit cost	Hr Rate	Amount
Salary	$50	30/wk	$12,000
Subscription	$99	1 month	$99
Compute	$0.4/DBU	350	$1,400
Storage	$3.3	350	$1,500
		Subtotal	$14,999
		Tax	$1,950
		TOTAL	**$16,950**

The Compute Cost

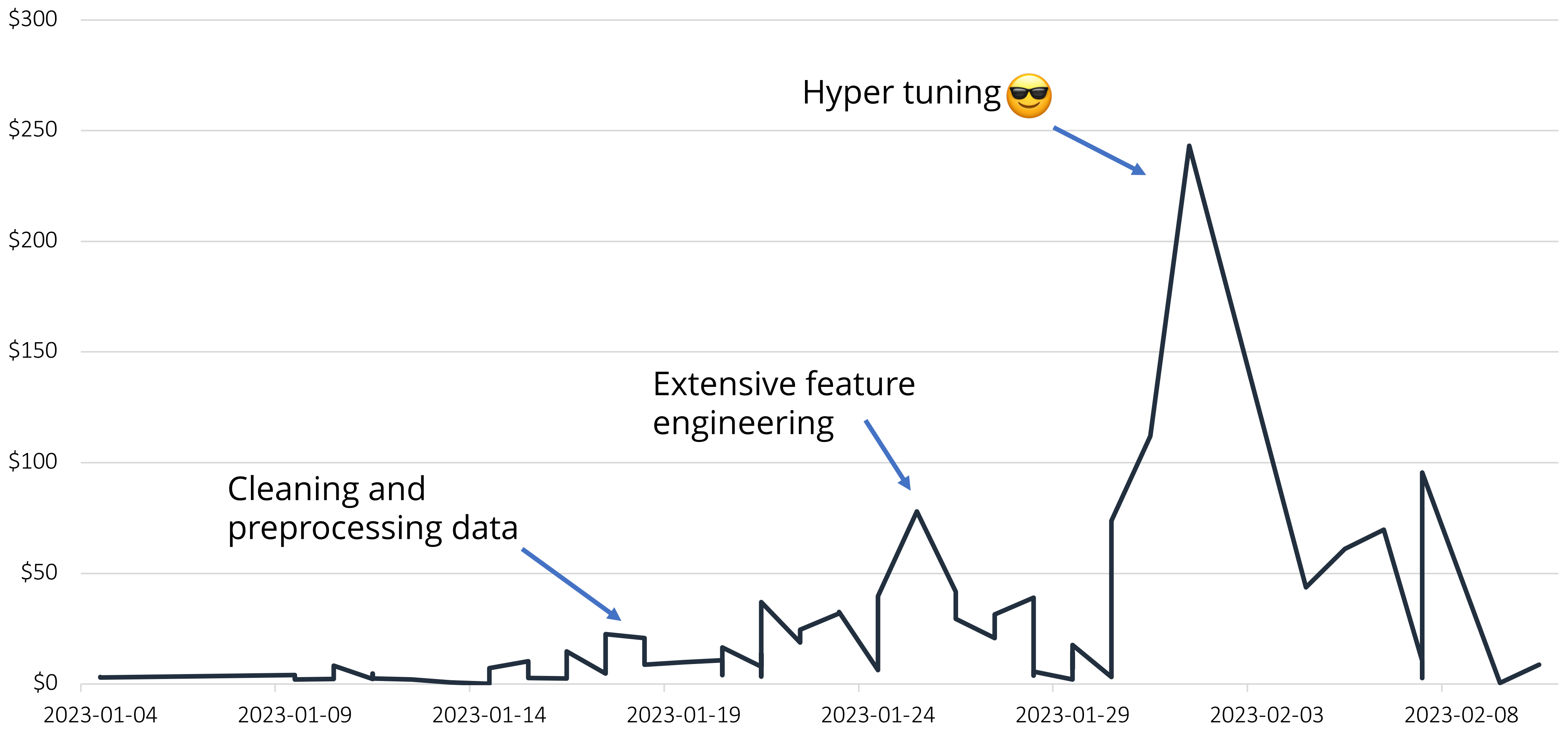


Chart 6: Daily project cost

Insights from Our Modeling Journey

Lessons Learned:

- Take time to **read and understand the data** before diving into modelling
- **Text data contains a lot of features** that can be extracted and used to make predictions
- Avoiding overfitting is crucial for obtaining accurate results

Next steps:

- Continue to **optimize hyperparameters** to achieve even better performance
- Explore and include **additional features**

Any questions?

Appendix

Final Features

```
featuresList = [
    "overall", "verified", 'meanRating', 'reviewsCount',
    "isBook", "year", "month", "isWeekday", "daysSinceReview", "seasonEncoded",
    'daysSinceFirstReview',
    'isAnonReviewer', "activeReviewer",
    "summaryHasLink", "isNASummary", 'isGenericSummary', "summaryFeatures",
    'isGenericReview', "textFeatures",
    'count_nouns', 'count_verbs', 'count_adjs', 'count_advs',
    'sentence_count', 'word_count', 'char_count',
    'sentiment_score', 'helpfulProportion'
]
```

Function Examples

```
def isBook(df):
    """
    Creates a new bool column isBook that identifies if a review was left for a book or not. If the ASIN number should correspond to the ISBN number - a commercial book identifier -- starts with 00 in this dataset
    """
    df = df.withColumn("isBook", F.col("asin").startswith("00"))
    return df
```

```
def isAnonReviewer(df):
    """
    Create a column that identifies if a user has a custom name or Amazon-predefined: Amazon Customer or Kindle Customer
    """
    anon_reviewers = ['amazon customer', 'kindle customer', 'Amazon Customer', 'Kindle Customer']
    df = df.withColumn('isAnonReviewer',
                       F.when(df["reviewerName"].isin(anon_reviewers), True).otherwise(False))
    return df
```

Function Examples 2

```
def extractReviewTextFeatures(df):
    df = cleanParsingErrors(df)
    df = cleanUpText(df)
    df = isGenericReview(df)
    df = applyReviewTransformPipe(df)
    df = countPOSFeatures(df)
    df = extractCountFeatures(df)
    df = getSentimentScore(df)
    return df
```

Command took 0.09 seconds -- by 22oh1@queensu.ca at 2/15/2023, 9:14:21 A

Cmd 10

Main function

```
def preprocDF(df):
    df = extractNonTextFeatures(df)
    df = extractReviewTextFeatures(df)
    df = extractSummaryTextFeatures(df)
    return df
```