Amazon- Understanding Customer Reviews

amazon

MMA 865 Big Data Analytics
Team Adelaide

Oct 05, 2022

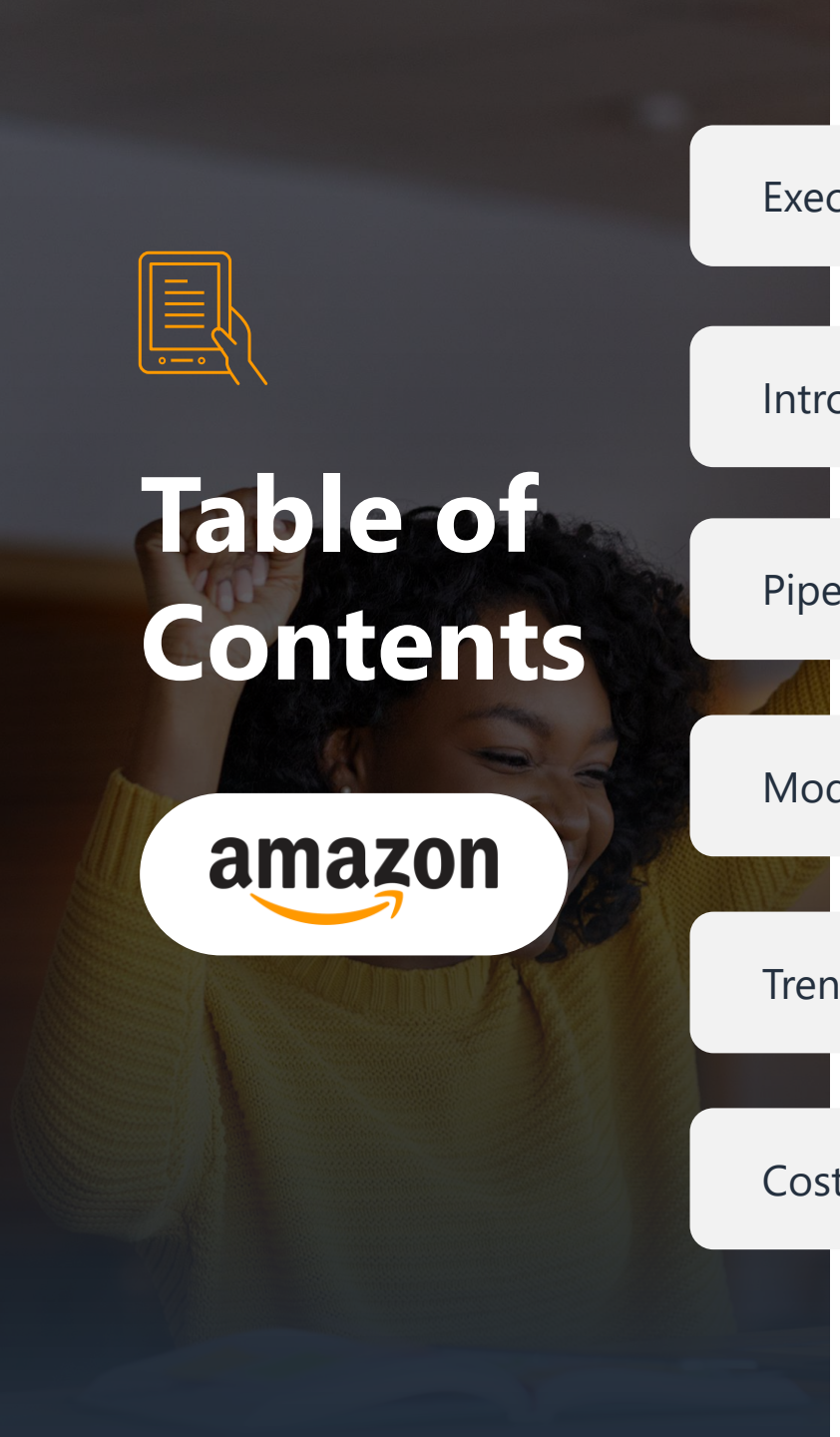


Table of Contents

Executive Summary

Objective

- Predict whether a review is helpful or not

Methodology

- Data Preprocessing
- Feature Engineering
- Logistic Regression

Outcome

- AUC Score: 0.888
- Feature importance
- Recommendation

Introduction and data

Amazon product data

- 3.14 million reviews
- Sep 1996 - Sep 2018


```
{
  "reviewID": "15632",
  "overall": 5.0,
  "verified": true,
  "reviewTime": "09 13, 2009"
  "reviewerID": "A2SUAM1J3GNN3B",
  "asin": "0000013714",
  "reviewerName": "J. McDonald",
  "reviewText": "I bought this for my husband who plays the piano. He is having a wonderful time playing these old hymns. The music is at times hard to read because we think the book was published for singing from more than playing from. Great purchase though!",
  "summary": "Heavenly Highway Hymns",
  "unixReviewTime": 1252800000,
  “label”: 1 ← This is the target column
}
```


amazon

Data Cleaning

- Remove duplicates
  - 3,138,710 → 2,904,161
- Check null values
  - No null values found
- Correct data type
  - Date: string → date format
  - Verified: boolean → integer

Text Preprocessing

1. **Sentence Detector:**

   [fits the bed well, comfortable warm and cozy]

2. **Tokenizer**

   ["fits", "the", "bed", "well", ",", "comfortable", "warm", "and", "cozy"]

3. **Normalizer**

   ["fits", "the", "bed", "well", "comfortable", "warm", "and", "cozy"]

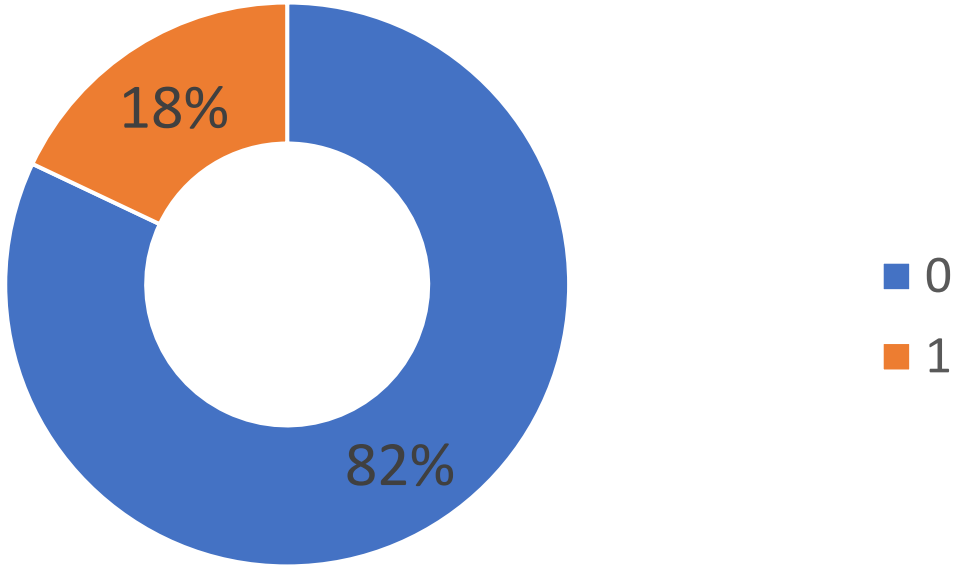
4. **Stop Words Cleaner**

   ["fits", "bed", "well", "comfortable", "warm", "cozy"]

5. **Lemmatizer**

   ["fit", "bed", "well", "comfortable", "warm", "cozy"]

Features vs Target Variables



Label 0
- The average length of review text: 275 characters
- The average date difference: 2,910 days

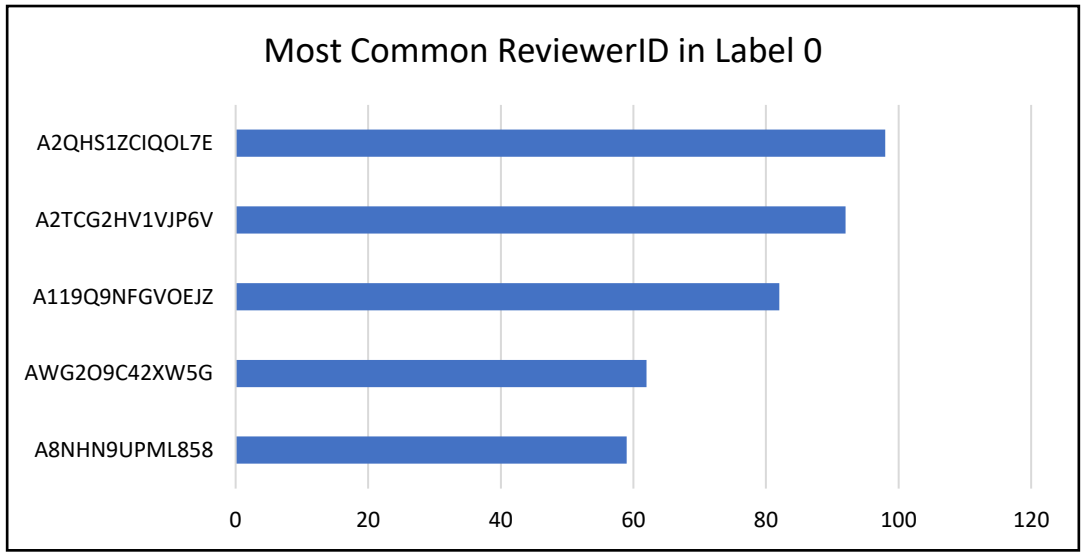
Label 1
- The average length of review text: 1,017 characters
- The average date difference: 4,257 days

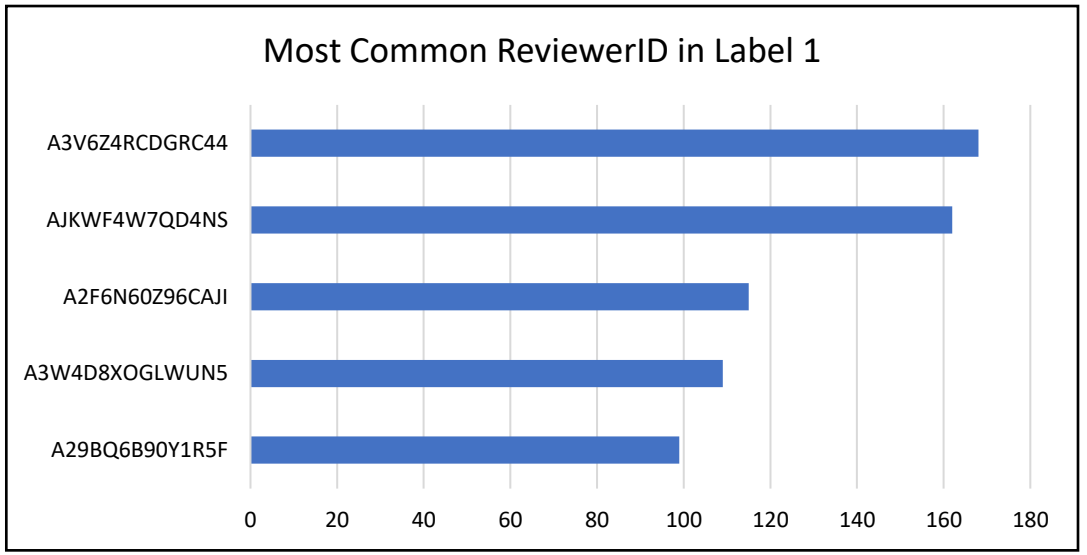
Verified	Label 0	Label 1
True	649,324 (87%)	97,754 (13%)
False	106,272 (61%)	68,112 (39%)

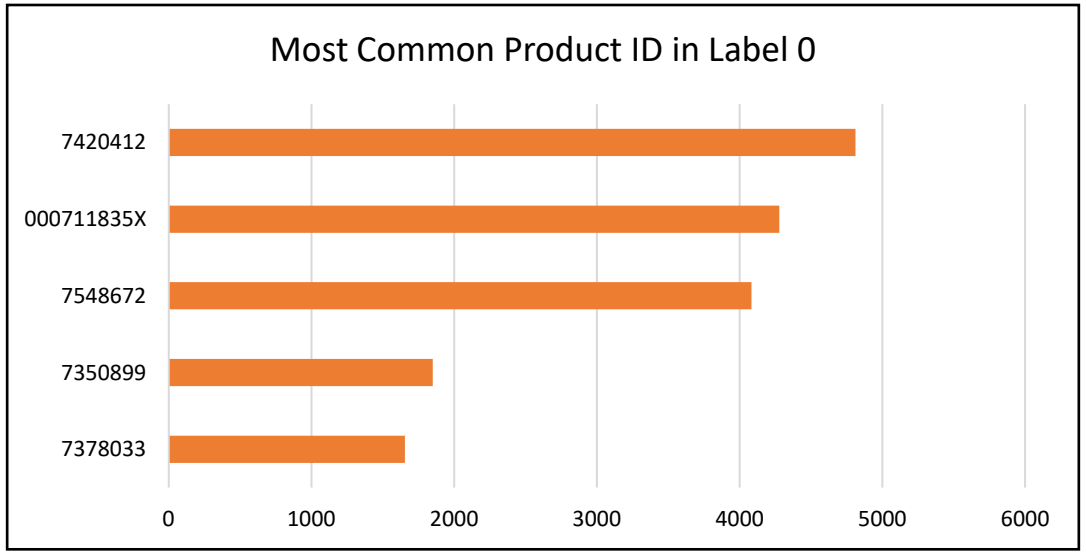
Overall	Label 0	Label 1
1	26,473 (56%)	20,811 (44%)
2	27,144 (67%)	13,386 (33%)
3	58,864 (76%)	18,824 (24%)
4	131,773 (81%)	30,227 (19%)
5	510,467 (86%)	82,492 (14%)

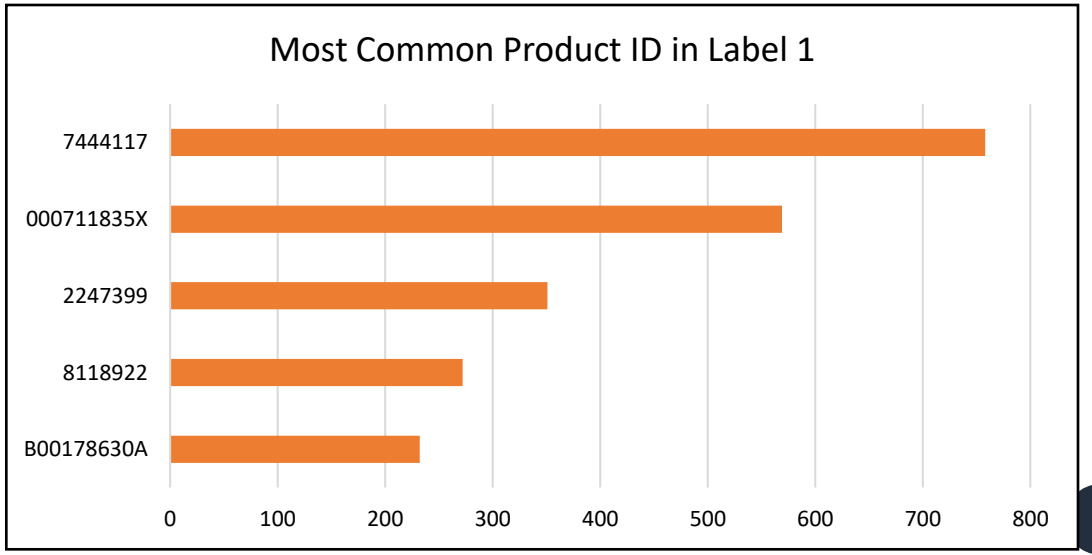
* Data sampled by 30%

Features vs Target Variables





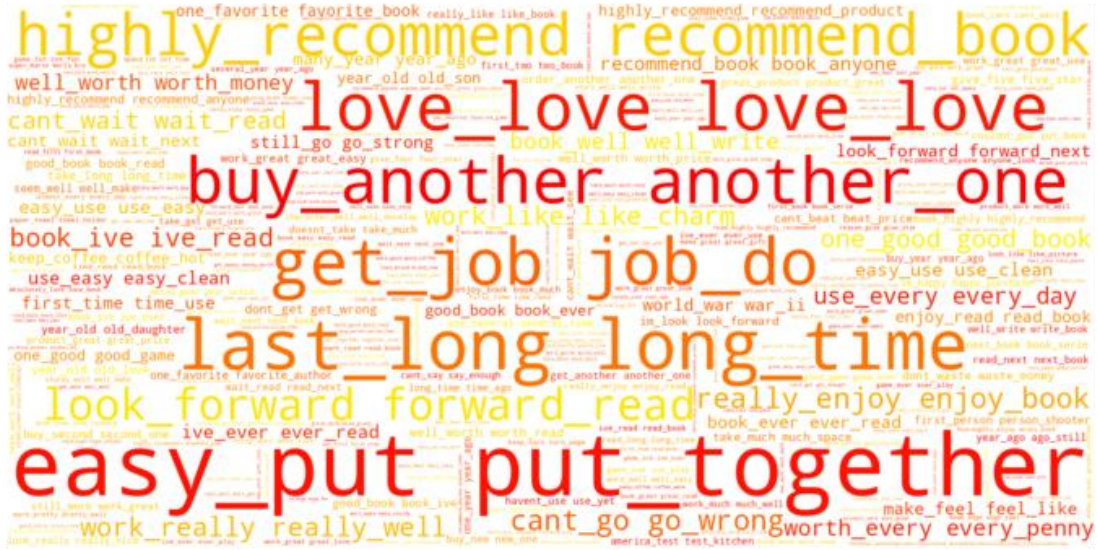




* Data sampled by 30%

Word Frequency

Label 0



Label 1

* Data sampled by 30%

Feature Engineering

New Features

- **Review length:** length of "reviewText" characters
- **Days from review date:** days= current date – review time

One Hot Encoder

- Reviewer ID;
- ASIN;
- Overall

Pipeline & Modelling

Pipeline 1: Review Text

Pipeline 2: Summary

Raw Text → Word → Feature Engineer → Logistic Regression → Prediction

- Document Assembler;
- Tokenizer;

- Normalizer;
- Stop words;
- Lemmatizer;
- Finisher

Vector Assembler

- Indexer
- Encoder;
- TF-IDF

- maxIter=300;
- regParam= 0.01;
- elasticNetParam=0.0

Data Split:

Train: 80%

Test: 20%

Model Output

0.9074

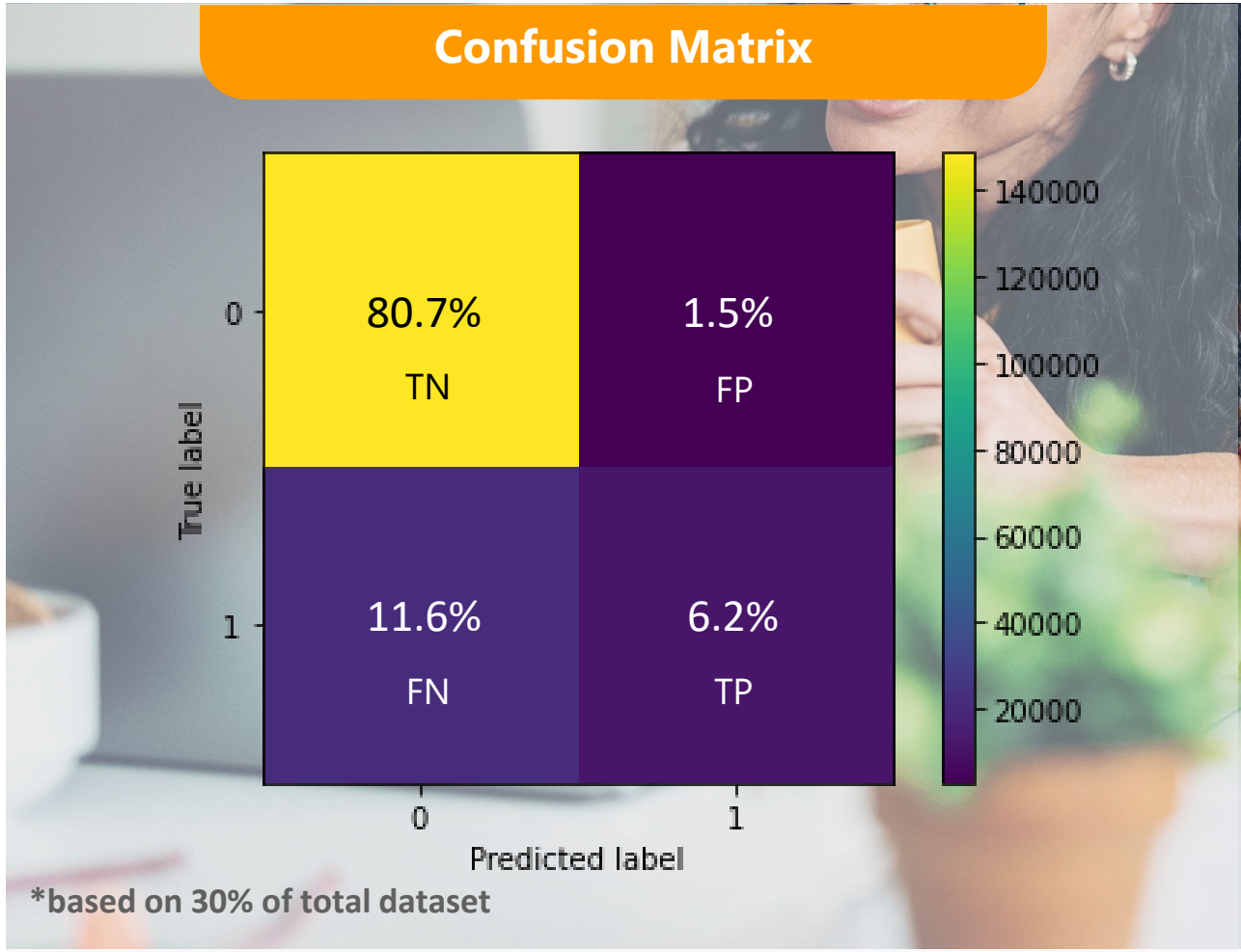
AUC

0.8885

Kaggle AUC

Key Takeaways

- Precision Score: 0.81
- Recall Score: 0.35
- F1 Score: 0.49



Trend Exploration

Based on Regression Coefficients: -

- Length has the highest coefficient
- IDF has the highest negative coefficient
- Days and ReviewerID have the most negligible impact on the target
- Most features have a negative impact on the target as value increases.

Regression Coefficients

Feature Names	Coefficient
Length	0.290
ReviewerID	0.062
Days	-0.078
Asin	-0.107
IDF2	-0.124
Verified	-0.130
Token_Features	-0.131
Overall	-0.150
IDF	-0.170

amazon

* Data sampled by 30%

Trend Exploration

Based on Feature Importance: -

- Most Important feature: Length
- Overall rating and Verified are not strongly related to the usefulness of the review
- Mapped tokens to their index to identify important tokens

Feature Importance

Feature Names	Value	Score
Length	Character length	0.049
IDF_56	“try”	0.034
IDF_282	“style”	0.030
IDF_1041	“intend”	0.028
IDF_75	“bit”	0.027
IDF_258	“although”	0.027

amazon

* Data sampled by 30%

Keywords

Most Important Tokens: -

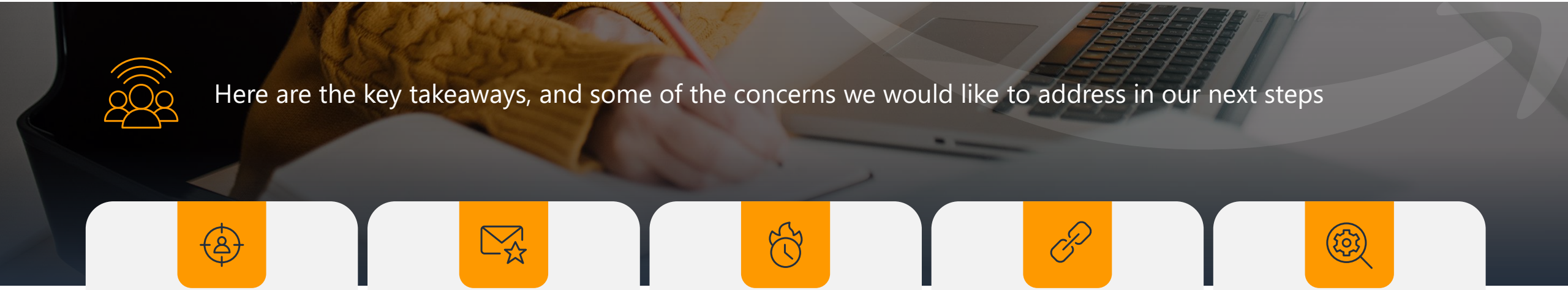
- Try
- Style
- Intend
- Bit
- Although

Least Important Tokens: -

- Scary
- Advantage
- Single
- Quest
- Classic

* Data sampled by 30%

Business Insights



Here are the key takeaways, and some of the concerns we would like to address in our next steps

- The length of a review is the most significant feature. Longer reviews are more helpful

- The frequency of words in reviews and summaries is negatively related to the target.
- Common words are less helpful

- Latest reviews are observed to be more useful than older ones.

- Reviews that have a lower Overall Rating tend to be more useful

- Word tokens have lower explainability
- Verified reviews are less useful

Cost Estimation

Time Cost	Number of days (Project Duration)	4 weeks (28 days)
Human Resource Cost	Number of Analysts	6
	Per Analyst per day hours used	3
	Analyst costs per hour	$50
	Total Cost	$25,200
Computational Cost	DataBricks Model used	Standard, West US region, pay as you go
	Workload	All-Purpose Compute
	Subscription Cost (CAD)	$0.52/DBU-hour
	Processing time per analyst (daily)	5 hrs
	Total Cost	$436.8
Data Collection Cost	Dataset	Amazon Reviews Dataset
	Source	https://jmcauley.ucsd.edu/data/amazon/
	Cost	Free
Data Storage and Management Costs	Storage Location	Databricks DBFS on cloud
	Storage cost	Free with subscription
Total Cost		$25,636.8

amazon

Appendix

amazon

Data Cleaning & Feature Engineering

Cmd 2

```
# Convert Unix timestamp to readable date
from pyspark.sql.functions import *
from pyspark.sql.types import *

# Data Cleaning
df = df.dropDuplicates(['reviewerID', 'asin'])
df = df.withColumn("reviewTime", to_date(from_unixtime(df.unixReviewTime))).drop("unixReviewTime")
df = df.withColumn("verified", df.verified.cast("int"))
```

df: pyspark.sql.dataframe.DataFrame = [reviewID: integer, overall: double ... 8 more fields]

Command took 0.10 seconds -- by 21jc69@queensu.ca at 03/10/2022, 09:39:27 on MMA-2023W-Adelaide

Cmd 3

```
import pyspark.sql.functions as f
# Feature Engineering
df = df.withColumn("len", f.length("reviewText"))
df = df.withColumn('days', datediff(current_date(),col("reviewTime")))
```

df: pyspark.sql.dataframe.DataFrame = [reviewID: integer, overall: double ... 10 more fields]

Command took 0.10 seconds -- by 21jc69@queensu.ca at 03/10/2022, 09:39:27 on MMA-2023W-Adelaide

Final Model Databricks link:
https://adb-3564912089489985.5.azuredatabricks.net/?o=3564912089489985#notebook/4213848583282594/command/1742315359495815

Pipeline

```
# pipeline
pipeline = Pipeline(stages=[
    indexer,
    encoder,
    document_assembler,
    sentence,
    tokenizer,
    normalizer,
    stopwords_cleaner,
    lemmatizer,
    finisher,
    tf,
    idf,
    document_assembler2,
    sentence2,
    tokenizer2,
    normalizer2,
    stopwords_cleaner2,
    lemmatizer2,
    finisher2,
    tf2,
    idf2,
    assembler,
    lr])
```

Mapping Tokens to Index

```
# Create a map between eachtoken and its index
from pyspark.sql.functions import explode, udf, col
from pyspark.sql.types import *

make_list_udf = udf(lambda col: [col], ArrayType(StringType()))
remove_list_udf = udf(lambda col: col[0], StringType())

def get_index(col):
    if len(col.indices) == 0:
        return -1   # Mark the ngram's index as -1 if it is not the top 2^12 ngrams
    else:
        return int(col.indices[0])

get_index_udf = udf(get_index, IntegerType())

token_index = trainingDataTransformed.select(explode(trainingDataTransformed.token_features).alias("token_features")).distinct() \
                            .withColumn("token_features", make_list_udf("token_features"))

tff = CountVectorizer(inputCol="token_features", outputCol="rawF", vocabSize=10000, minTF=1, minDF=50, maxDF=0.40)

trans = tff.fit(trainingDataTransformed)
token_index = trans.transform(token_index)

token_index = token_index.withColumn("token_features", remove_list_udf("token_features")) \
                         .withColumn("index", get_index_udf("rawF")) \
                         .select("token_features", "index")
```