Reviews Sentiment Analysis

ECOMMERCE PRODUCT REVIEWS – PAIRWISE RANKING

Content Index

1.Introduction of Review of Sentiment Analysis

This project analyzes a dataset containing ecommerce product reviews. The goal is to use machine learning models to perform sentiment analysis on product reviews and rank them based on relevance. Reviews play a key role in product recommendation systems.

Objective of the Project

objective Info

Enormous amount of review creates the problem. They are not segregate useful ones. customer not decided about the product based on reviews either they buy or not.
So, the goal is to developed the POC which is use machine learning models to perform sentiment analysis on product reviews and rank them based on relevance.

why reviews so important

Reviews play a key role in product recommendation systems. To improve the credibility of the platform, get the ranking on google via SEO, help the user take a decision to buy a product or not, boost the sales of the product to a good one, and increase the session time of a user on the platform.

What is Diff btwn Good & Bad reviews

Good Reviews

A good review is a review containing some useful information. It is a complete statement (not in single or two-three words)

Bad Reviews

A non informative review is a bad review. A meaningless sentence or single word review

some informative reviews example

★★★★★ **Best Sanetizer**

Reviewed in India on 19 March 2020

Design: Sanitizers | Verified Purchase

Very good sanatizer , Best one available in market, Go for it don't think for a single second.

47 people found this helpful

Helpful | Comment | Report abuse

4★ **Worth the money**

great value for digital thermometer. visible accurate reading. easy to carry in your handbag easy to operate.

Saroj Rawal Certified Buyer, Benaulim 1 month ago

Faisal siddiqui

★☆☆☆☆ **Poor quality, the worst part is quality of bottle which contains oil**

Reviewed in India IN on 8 July 2022

Verified Purchase

5★ **Perfect product!**

good

Flipkart Customer Certified Buyer, Jamal Jul, 2018

21 8

2★ **Slightly disappointed**

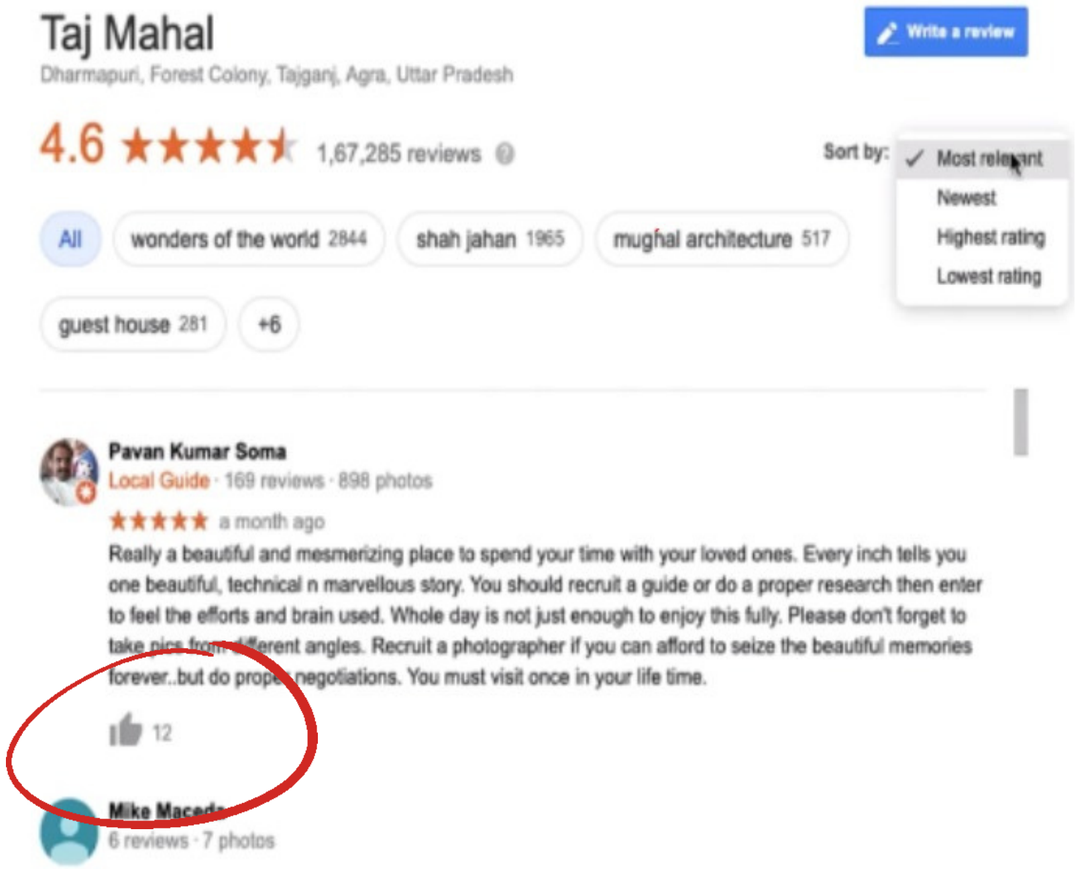
Not a good product

Deepesh Sharma Certified Buyer, Noida Aug, 2018

1 1

5★ **Super!**

Example of Google reviews on Tajmahal customer reviews with upvotes



Problem with the Bad reviews

These enormous amounts of reviews also create problems for customers as they are not able to segregate useful ones. Regardless, these immense proportions of reviews make an issue for customers as it becomes very difficult to filter informative revies this proportional issue has been attempted. The approach that we discuss in the detail later ranks reviews abased on their relevance with the product and ranks down irrelevant reviews

So on what basis google rank out their reviews

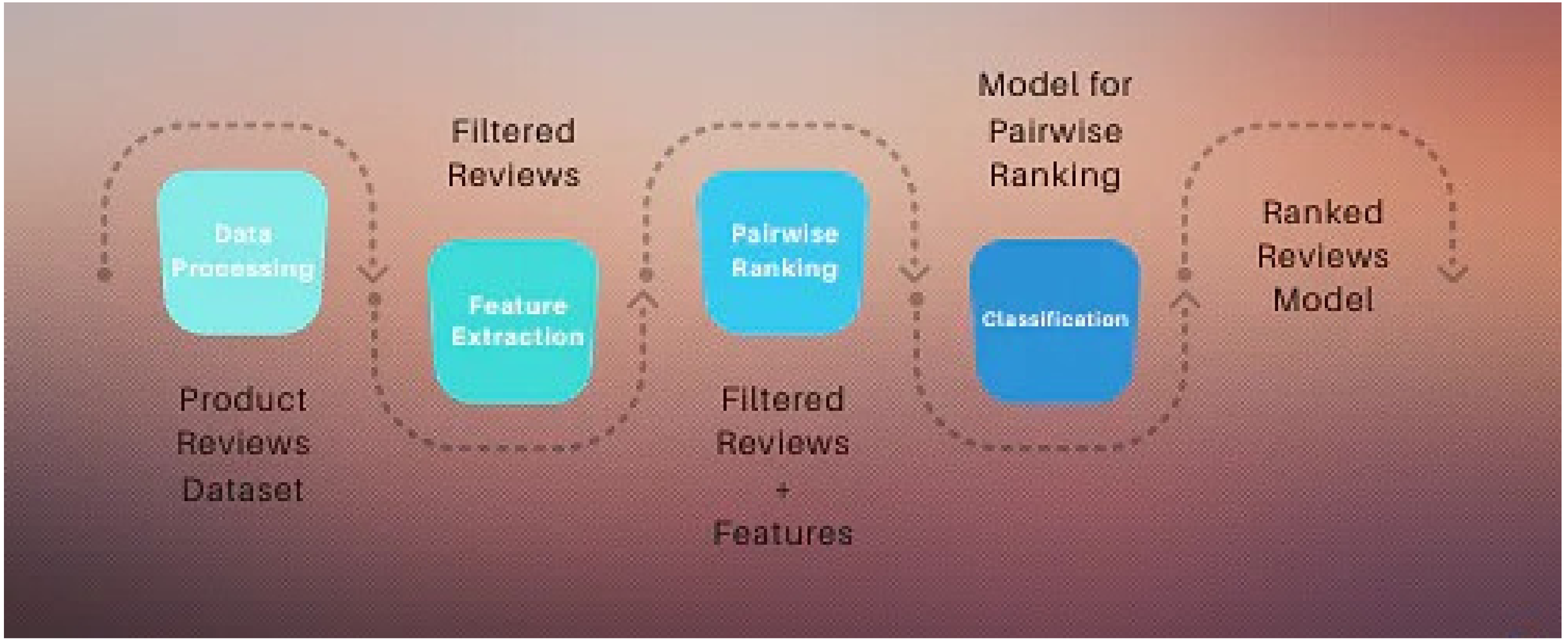
1. Recent Time
2. what kind of context is written out by users i.e is it good context or bad context
3. Yes, Offcourse google also saw the Likes,How much likes on the review

Summarizing Problem Statement

Let suppose we are in the new platform,and the reach of the customers are very less.that if the customers gives the feedback about your product and that reviews are not so much informative.another customers will comes and see the reviews of the customers of -ve and +ve one. That time the probability of the busyness of that product will less because how will confused buy watching reviews if the reviews are not so informative and not rank out on the top most.

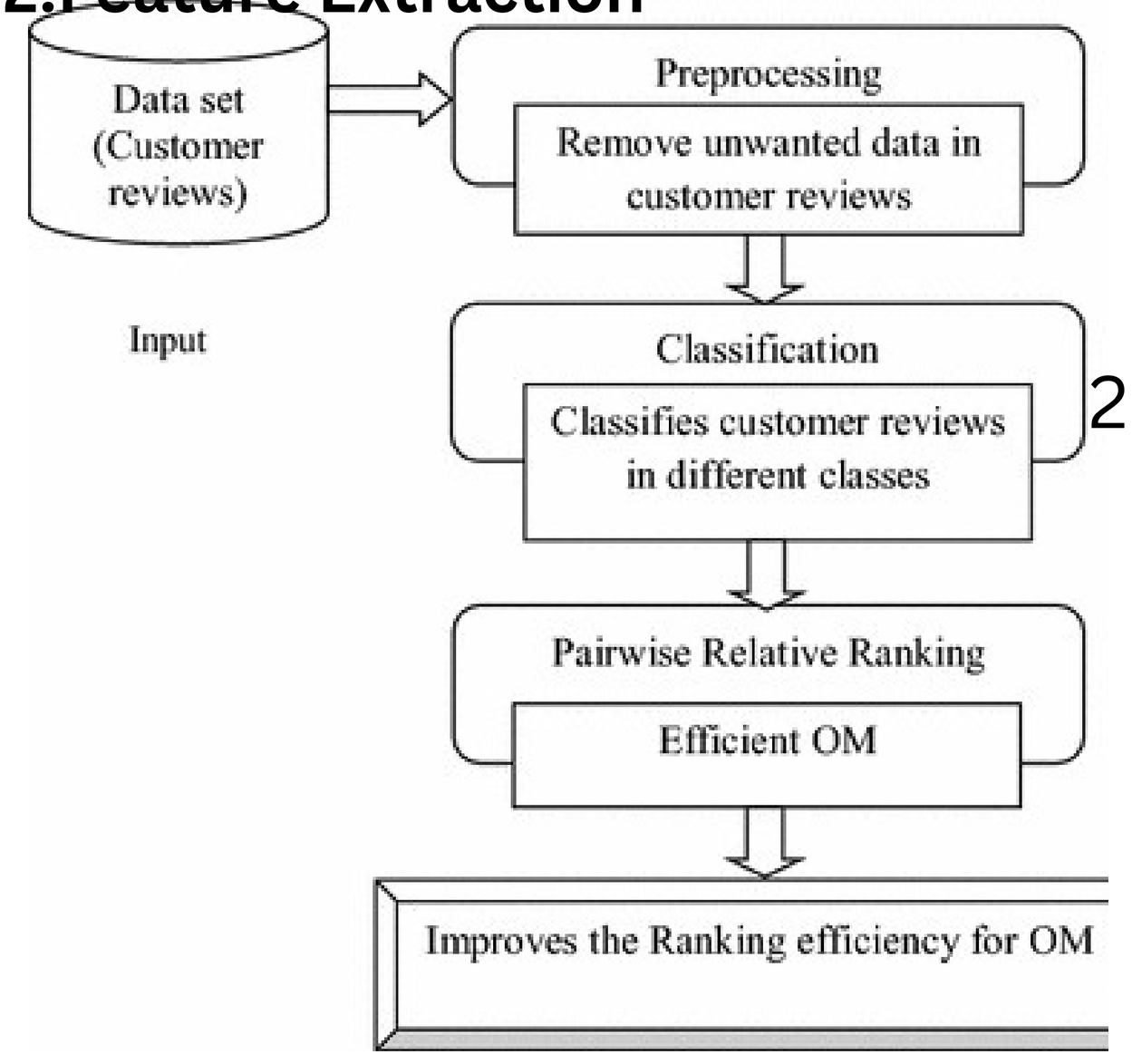
So we are developing the **POC** of **Sentiment analysis and pair wise ranking of Reviews**. With the help of **Machine learning and NLP**.

Approch for Improvement of Ranking of good and important Reviews

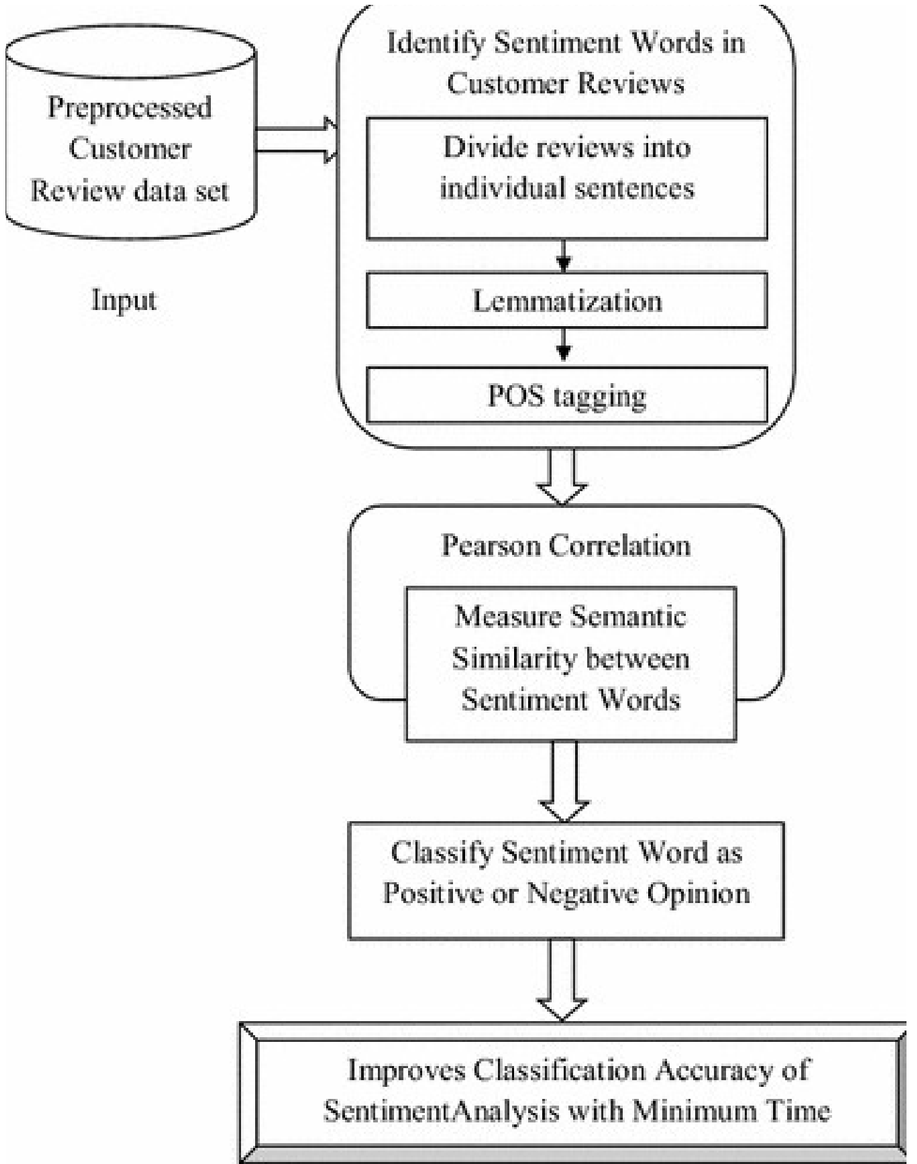


Approch for Improvement of Ranking of good and important Reviews

1.Data Preprocessing
2.Feature Extraction

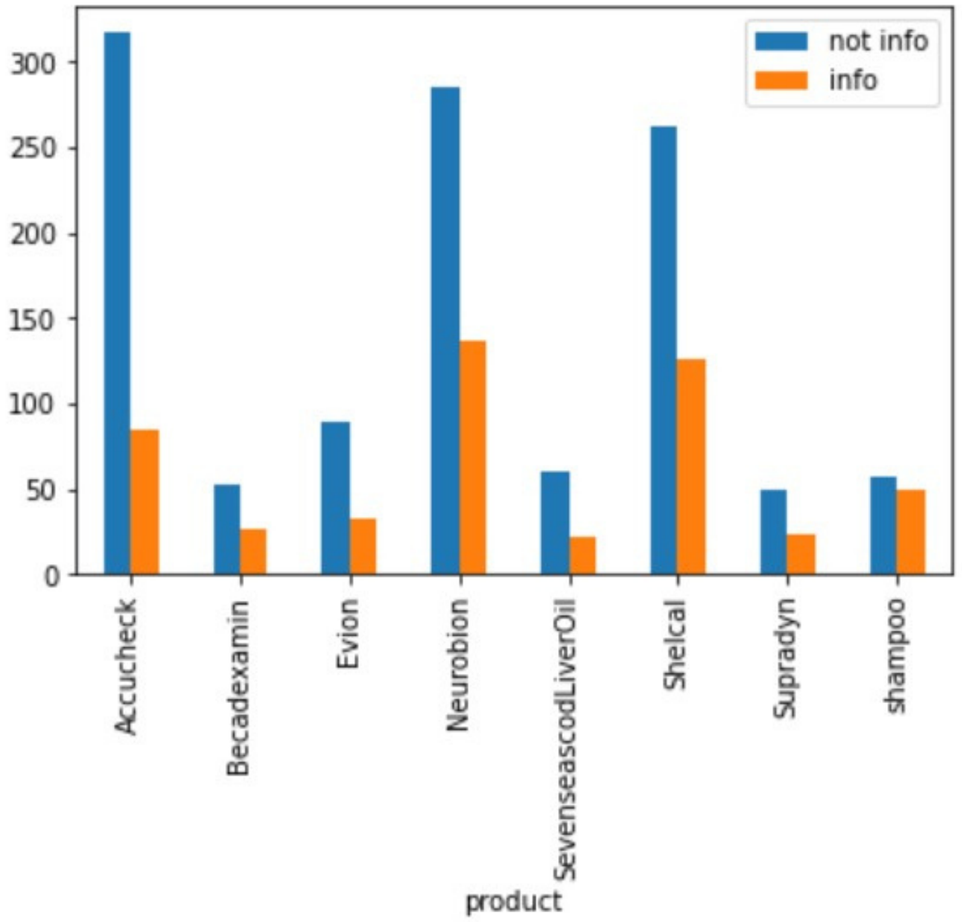


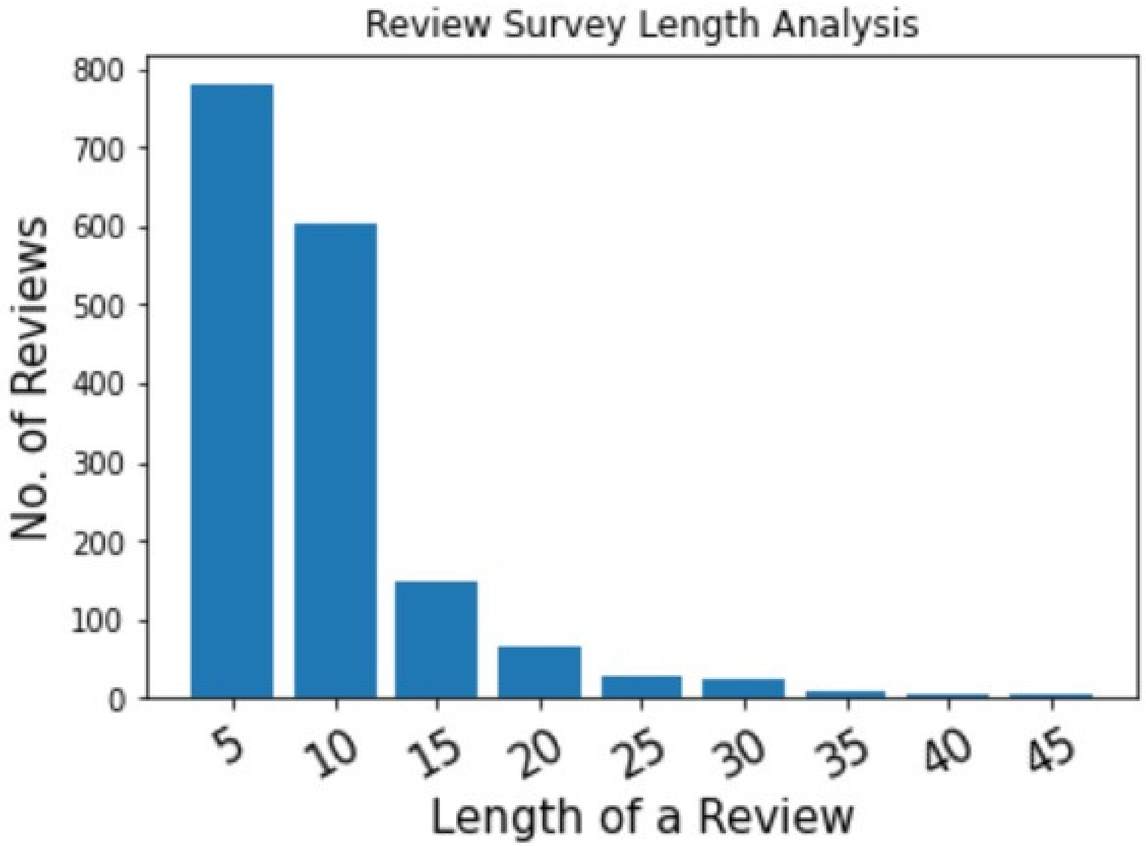
2



Stage1. Data-Preprocessing
&
EDA

Analysis to understand per product who many informative and how many not informative reviews are there.





Different Stages while handling the Reviews

Stage1: Language Detection

Stage 2: Gibberish Reviews

Stage 3: Profanity Detection

Stage 4: Spelling Correction

Stage 5: Company Tag

Stage 6: Slang detection

Stage1: Language Detection

language detection

```
[ ]
    rf.language_detection('भैयाजी कैसे हो आप')

    # hindi language detected
```

```
'hi'
```

```
[ ] rf.language_detection('भाऊ कसा आहेस')
    # marathi language detected
```

```
'mr'
```

```
[ ] rf.language_detection('брат как ты')

    # russian language detected
```

```
'ru'
```

Stage 2: Gibberish Reviews

Gibberish detection

```
[ ] rf.gibberish_detection('cow')
```

```
False
```

```
[ ] rf.gibberish_detection('idffdbfjbf')
    # it detecting correctly
```

```
True
```

Stage 3: Profanity Detection

Profanity detection

```
[ ] rf.english_swear_check('MotherFucker I dont want anything')
```

```
True
```

```
[ ] rf.english_swear_check('mother fucker')
```

```
True
```

Stage 4: Spelling Correction

spelling correction

```
[ ] rf.spell_correct('wokr', 0.9)
```

```
'work'
```

```
[ ] rf.spell_correct('hapyp',0.9)
    # 0.9 is confidance interval where
```

```
'happy'
```

Stage 5: Company Tag

service tagger detection

- it mean rather than talking on product you talk on service of that product

```
[ ] rf.service_tag('amazon')
```

```
0
```

```
[ ] rf.service_tag('google')
    # this not working properly we have to check
```

```
0
```

Stage 6: Slang detection

slang emoji detecion

```
rf.slang_emoji_polarity_compoundscore('😀')
```

```
0.3612
```

Stage2. Feature-Engineering

From these review text we wanted to extract relevance out of these, understanding in depth sense of reviews. Features extraction covers every necessary property/viewpoints and to measure features in a quantitative manner is a much-needed task in order to achieve highly accurate outcomes. Hence, this section discusses all the features extracted from reviews.

Noun Strength (Rn): Nouns are subjects and considered as the most informative part of a language. The amount of subjects shows the importance of review because only a noun describes the prime factors of review (which tells us what the review is about). We did POS Tagging to find nouns in a review and computed score as:

Score(Rn) = TFIDF(noun) / TFIDF(all words)

Review Polarity (Rp): Its value lies between -1 to +1 which tells whether a review has sentiment or negative sentiment. Review Subjectivity (Rs): The subjectivity is a measure of the sentiment being objective to subjective and goes from 0 to 1. Objective expressions are facts while Subjective expressions are opinions that describe a person's feelings. Consider the following expression:

Bournvita is **brown in color: Objective**
Bournvita tastes **very good with milk: Subjective**

Review Complexity (Rc): To evaluate how good and complex a review is, in terms of unique words within a review and across entire review corpus of a particular product. Rc = Number of unique words in a Review / Number of unique words in entire Corpus

Review Word Length (Rw): Word count of a Review

Service Tagger (Rd): The best review is one that talks more about how is the product, how it tastes, what are its uses, and the one which talks about the effectiveness of a product. Reviews are basically to describe a product. So, a dictionary of words is created which would mark reviews as service-based, delivery reviews, and customer support.

Fuzzy matching of every word in a review is done with the words in the dictionary with Levenshtein distance. Levenshtein distance helps in measuring the difference between two sequences and tackle spell errors in review, for example, instead of “My delivery was on time”, Reviews is wrongly written as “My dilivery was on time”. In this case, Fuzzy matching would help us to match both the reviews.

Compound Score (Rsc): To improve the efficiency of the system. We compute the compound score using VaderSentimentAnalyser. This library is taken from VADER (Valence Aware Dictionary and sEntiment Reasoner). This is a lexicon and rule-based sentiment analysis tool that is specifically tuned to determine sentiments expressed in social media content. It has the ability to find the sentiment of Slang (e.g. SUX!), Emoji (😩, 😂), Emoticons (:), :D) and the difference between capitalized word expressions(I am SAD, I am sad are different expressions).

Rsc ≥ 0.5 (Positive Sentiment)

-0.5<Rsc<+0.5 (Neural Sentiment)

Rsc≤ -0.5 (Negative Sentiment)

Miscellaneous: We purposely did not include Reviews Rating as a feature. Inclusion of Ratings totally blunders the entire system because of two reasons:

1. Common confusion between Rating and Reviews. For example, someone who rates the product ‘1’ (On a rating scale of 1–5, ‘1’ being the ‘lowest’ and ‘5’ being the ‘highest’) writes the review comment as ‘very good and useful medicine’.
2. A large portion of Reviews from customers are either 5 stars or 1 star.

DataFrame after passing through all the functions

```
df
```

	product	answer_option	label	review_len	Rn	Rp	Rs	Rc	Rd	Rsc
0	Accucheck	Fast and accurate delivery	0	4	0.232859	0.300000	0.616667	0.005420	1.0	0.0000
1453	Accucheck	Expected a longer expire date. Your Product Li...	0	14	0.596318	-0.100000	0.400000	0.017615	1.0	0.0000
1454	Accucheck	I liked the prompt service	0	5	0.319747	0.600000	0.800000	0.006775	1.0	0.4215
1455	Accucheck	Good product	0	2	0.546925	0.700000	0.600000	0.002710	0.0	0.4404
1456	Accucheck	I not needed	0	3	0.000000	0.000000	0.000000	0.004065	0.0	0.0000
...	...	...	...	...	...	...	...	...	...	...
131	shampoo	Its not much effective as it has been stated i...	0	12	0.500000	-0.300000	0.800000	0.028640	0.0	-0.3724
130	shampoo	Liked it very nicely working now my scalp is a...	1	11	0.166375	0.690000	0.900000	0.026253	0.0	0.5709
129	shampoo	its my regular choice	0	4	0.500000	0.000000	0.076923	0.009547	0.0	0.0000
139	shampoo	Good but not very effective	0	5	0.000000	0.234615	0.607692	0.011933	0.0	-0.4032
192	shampoo	It is better but not a permanent solution for ...	0	10	0.407447	0.500000	0.500000	0.023866	0.0	-0.1263

1655 rows × 10 columns

Stage3.Model Building

Phase3) Model Building and Training

```
[ ] import warnings
    warnings.filterwarnings('ignore')
```

```
    import pandas as pd
    import numpy as np
    from joblib import load, dump
    from copy import deepcopy
    from statistics import mean

    from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
    from collections import Counter
```

Ranking is a canonical problem for humans. It is easy to classify whether a review is useful (informative) or not. However, ranking reviews on the basis of usefulness, is a complex task. Our ranking methodology is based on this simple education.

Pairwise ranking approach is applied to rank reviews in the semi-supervised learning method. The pairwise ranking approach looks at a pair of documents at a time in a loss function and predicts a relative ordering. The objective is not to determine the relevance score but to find which document is more relevant than others. This relevance is developed to judge the preference of one review over another.

In the semi-supervised learning method, mapping is constructed between input and output. This input-output pair in the training model is used to learn the system.
Review Segregation: We segregated two sets of reviews on which we train our model.

Set 0 represents reviews with label 0, i.e., ones that are not informative. These include reviews based on delivery, customer support, packaging, etc. These reviews do not describe the product.

Set 1 represents reviews with label 1, i.e., reviews that are informative and are better than all reviews of Set 0;
How we segregated and determined labels for reviews:

Our entire review ranking system is based on the idea that it is easier for humans to binary classify reviews which we call Set 0 and Set 1.

For each product 'Accucheck', 'Becadexamin', 'Evion', 'Neurobion','SevenseascodLiverOil', 'Shelcal', 'Supradyn','shampoo', we asked 10 different people to label reviews as a 1 (informative review) and 0 (not informative review). Different participants were asked to label so that there is no bias and the model learns to its best.

split the function converge values in 2 sets,{0} & {1 } for binary classification

```
train
```

	review_len_x	Rn_x	Rp_x	Rs_x	Rc_x	Rd_x	Rsc_x	review_len_y	Rn_y	Rp_y	Rs_y	Rc_y	Rd_y	Rsc_y	target
0	5	0.544357	0.52	0.823333	0.006775	0.0	0.0000	4	0.232859	0.300000	0.616667	0.005420	1.0	0.0000	1
1	5	0.544357	0.52	0.823333	0.006775	0.0	0.0000	14	0.596318	-0.100000	0.400000	0.017615	1.0	0.0000	1
2	5	0.544357	0.52	0.823333	0.006775	0.0	0.0000	5	0.319747	0.600000	0.800000	0.006775	1.0	0.4215	1
3	5	0.544357	0.52	0.823333	0.006775	0.0	0.0000	2	0.546925	0.700000	0.600000	0.002710	0.0	0.4404	1
4	5	0.544357	0.52	0.823333	0.006775	0.0	0.0000	2	0.546925	0.700000	0.600000	0.002710	0.0	0.4404	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
212207	10	0.407447	0.50	0.500000	0.023866	1.0	-0.1263	15	0.479132	0.700000	0.600000	0.035800	1.0	0.1531	0
212208	10	0.407447	0.50	0.500000	0.023866	1.0	-0.1263	7	0.392391	0.316667	0.600000	0.016706	0.0	0.2382	0
212209	10	0.407447	0.50	0.500000	0.023866	1.0	-0.1263	25	0.417103	0.233333	0.255556	0.059666	1.0	0.5927	0
212210	10	0.407447	0.50	0.500000	0.023866	1.0	-0.1263	10	0.467972	0.500000	0.500000	0.023866	0.0	0.1779	0
212211	10	0.407447	0.50	0.500000	0.023866	1.0	-0.1263	11	0.166375	0.690000	0.900000	0.026253	0.0	0.5709	0

212212 rows × 15 columns

```
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression()
classifier.fit(X_train,y_train)
```

```
LogisticRegression()
```

Logistic Regression

```
[ ] print("Training Accuracy\n", accuracy_score(y_train,classifier.predict(X_train)))
    print("Test Accuracy\n", accuracy_score(y_test,classifier.predict(X_test)))
```

```
Training Accuracy
 0.7246729379333093
Test Accuracy
 0.7227104587328889
```

Classification Report

accuracy of model

```
print('CLASSIFICATION REPORT')
print("Training\n", classification_report(y_train,classifier.predict(X_train)))
print("Test \n", classification_report(y_test,classifier.predict(X_test)))
```

```
CLASSIFICATION REPORT
Training
              precision    recall  f1-score   support

           0       0.72      0.73      0.73     84884
           1       0.73      0.72      0.72     84885

    accuracy                           0.72    169769
   macro avg       0.72      0.72      0.72    169769
weighted avg       0.72      0.72      0.72    169769

Test
              precision    recall  f1-score   support

           0       0.72      0.72      0.72     21222
           1       0.72      0.72      0.72     21221

    accuracy                           0.72     42443
   macro avg       0.72      0.72      0.72     42443
weighted avg       0.72      0.72      0.72     42443
```

```
[ ] from sklearn.tree import DecisionTreeClassifier

    classifier = DecisionTreeClassifier()
    classifier.fit(X_train,y_train)
```

```
    DecisionTreeClassifier()
```

```
[ ] print("Training Accuracy\n", accuracy_score(y_train,classifier.predict(X_train)))
    print("Test Accuracy\n", accuracy_score(y_test,classifier.predict(X_test)))
```

```
    Training Accuracy
     0.9964127726498948
    Test Accuracy
     0.9816695332563674
```

```
    print('CLASSIFICATION REPORT')
    print("Training\n", classification_report(y_train,classifier.predict(X_train)))
    print("Test \n", classification_report(y_test,classifier.predict(X_test)))
```

```
    CLASSIFICATION REPORT
    Training
                  precision    recall  f1-score   support

               0       0.99      1.00      1.00     84884
               1       1.00      0.99      1.00     84885

        accuracy                           1.00    169769
       macro avg       1.00      1.00      1.00    169769
    weighted avg       1.00      1.00      1.00    169769

    Test
                  precision    recall  f1-score   support

               0       0.98      0.98      0.98     21222
               1       0.98      0.98      0.98     21221

        accuracy                           0.98     42443
       macro avg       0.98      0.98      0.98     42443
    weighted avg       0.98      0.98      0.98     42443
```

Decision Tree algorithm

Ensemble Model: RandomForest

```
[ ] from sklearn.ensemble import RandomForestClassifier

    classifier = RandomForestClassifier(n_estimators=50, n_jobs = -1, oob_score = True,random_state=42)
    classifier.fit(X_train,y_train)
```

```
RandomForestClassifier(n_estimators=50, n_jobs=-1, oob_score=True,
                       random_state=42)
```

```
[ ] print("Training Accuracy\n", accuracy_score(y_train,classifier.predict(X_train)))
    print("Test Accuracy\n", accuracy_score(y_test,classifier.predict(X_test)))
```

```
Training Accuracy
 0.9964127726498948
Test Accuracy
 0.9858869542680772
```

```
print('CLASSIFICATION REPORT')
print("Training\n", classification_report(y_train,classifier.predict(X_train)))
print("Test \n", classification_report(y_test,classifier.predict(X_test)))
```

```
CLASSIFICATION REPORT
Training
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     84884
           1       1.00      1.00      1.00     84885

    accuracy                           1.00    169769
   macro avg       1.00      1.00      1.00    169769
weighted avg       1.00      1.00      1.00    169769

Test
              precision    recall  f1-score   support

           0       0.99      0.99      0.99     21222
           1       0.99      0.99      0.99     21221

    accuracy                           0.99     42443
   macro avg       0.99      0.99      0.99     42443
weighted avg       0.99      0.99      0.99     42443
```

Random Forest

```
[ ] print("Test\nConfusion Matrix: \n", confusion_matrix(y_test, classifier.predict(X_test)))
```

```
Test
Confusion Matrix:
 [[20921   301]
 [  298 20923]]
```

```
## Score of the training dataset obtained using an out-of-bag estimate. This attribute exists only when oob_score is True.
classifier.oob_score_
```

```
0.9861340998651108
```

Confusion Matrix

```
[ ] feature_importance = pd.DataFrame(classifier.feature_importances_,
                        index = train.iloc[:,:-1].columns,
                        columns = ['importance']).sort_values('importance',ascending = False)
```

```
feature_importance
```

	importance
review_len_x	0.125124
Rc_y	0.124150
Rc_x	0.118689
review_len_y	0.113053
Rd_y	0.066992
Rd_x	0.066986
Rsc_y	0.061529
Rsc_x	0.060779
Rn_y	0.060668
Rn_x	0.058612
Rp_x	0.041460
Rp_y	0.040582
Rs_x	0.030987
Rs_y	0.030391

Feature Importance


Thank
You