Institute of Engineering & Technology

GLA
UNIVERSITY
MATHURA
Established vide U.P. Act 21 of 2010.

MINI PROJECT
On
Amazon Reviews Classification

Submitted by:
Tushar Gupta(171500360)
Sarthak Jain(171500294)
Apoorvi Goyal(171500060)

Submitted to:
Vinay Agrawal Sir

Department of Computer Engineering & Applications
Institute of Engineering & Technology

GLA University
Mathura- 281406, INDIA
2019-20

Declaration

I hereby declare that the work presented in this project titled "*Amazon Review Classification System*", submitted to the Computer Science and Engineering Department, GLA University, Mathura for the award of the Bachelor of Technology degree in Computer Science and Engineering, is my original work carried under the supervision of Vinay Agrawal, GLA University, Mathura.

Signature of Candidates:

Group Members: Sarthak Jain, Tushar Gupta, Apoorvi Goyal

Course: B.Tech. (CSE)

Year: 2019

Semester: 5th

CERTIFICATE

Certified that the project work entitled “Amazon Reviews Classification System” is an bonafied work carried out by

Tushar Gupta(171500360)
Sarthak Jain(171500294)
Apoorvi Goyal(171500060)

In partial fulfillment for the award of Bachelor of Engineering in Computer Science And Applications Engineering of the GLA University. Mathura during the year 2019-20. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited is the departmental library. The project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the said Degree.

Name of Guide	Coordinator Name	Name of HOD
(Vinay Agrawal)	(Shashi Shekhar)	(Prof. Anand Singh Jalal)

Synopsis

About the Project:

The problem being addressed in this project is the poor quality of Amazon reviews at the top of the forum despite the helpfulness rating system. The solution this problem is to pre-rate the reviews as ”helpful” or ”not helpful” for new reviews and the most helpful reviews are given position at the top. This way, poor quality reviews will be more unlikely to be shown at the top.

This is a binary classification of reviews on the basis of helpfulness as helpful or not helpful. The models will be trained on the basis of features extracted from the reviews. The recent reviews which are not rated as helpful by the users will be of poor quality but are still on top, we tried to make the helpful review to be at the top of the reviews shown to the buyer.

Motivation:

The reviews are particularly used so that new users can see the rated comments in order to help them make their own purchasing decisions. The quality of the comments made on internet forums has always suffered due to it’s users. Different websites have tried different methods for extracting more useful comments so that user can see top rated comments.

Future Prospects:

Our future work will be improving the accuracy of the models implemented. In order to improve the accuracy of our model, the following actions could be taken:

• There are lots of spelling mistakes in the reviews and abbreviations are used. This would result in a model that potentially has less features, as certain spelling errors would have been corrected/eliminated.

• Reviews with poor grammar, punctuation and improper word endings are more difficult to understand and would possibly lead to people rating them as less helpful.These reviews may contain useful information and may be helpful but certain other factors made them rated as not helpful. Thus preprocessing of data can be improved either by varying the values of threshold or the reviews that has not been selected because the product is not much rated.

Requirements:

a) Hardware :- 8 GB RAM, Intel i7 7th Gen., 1 TB ROM
b) Software :- Jupiter Notebook, Windows 10, MS Excel, LaTex

Acknowledgement

It gives us a great sense of pleasure to present the report of the B. Tech Mini Project undertaken during B. Tech. Third Year. This project in itself is an acknowledgement to the inspiration, drive and technical assistance contributed to it by many individuals. This project would never have seen the light of the day without the help and guidance that we have received.

Our heartiest thanks to Dr. (Prof). Anand Singh Jalal, Head of Dept., Department of CEA for providing us with an encouraging platform to develop this project, which thus helped us in shaping our abilities towards a constructive goal.

We owe special debt of gratitude to Vinay Agarwal, Department of CEA, for his constant support and guidance throughout the course of our work. His sincerity, thoroughness and perseverance have been a constant source of inspiration for us. He has showered us with all his extensively experienced ideas and insightful comments at virtually all stages of the project & has also taught us about the latest industry-oriented technologies.

We also do not like to miss the opportunity to acknowledge the contribution of all faculty members of the department for their kind guidance and cooperation during the development of our project. Last but not the least, we acknowledge our friends for their contribution in the completion of the project.

Sarthak Jain
Tushar Gupta
Apoorvi Goyal

Abstract

Customer reviews are increasingly available online for a wide range of products and services. They supplement other information provided by electronic storefronts such as product descriptions, reviews from experts, and personalized advice generated by automated recommendation systems. While researchers have demonstrated the benefits of the presence of customer reviews to an online retailer, a largely uninvestigated issue is what makes customer reviews helpful to a consumer in the process of making a purchase decision. Drawing on the paradigm of search and experience goods from information economics, we develop and test a model of customer review helpfulness. An analysis of 1,587 reviews from Amazon.com across six products indicated that review extremity, review depth, and product type affect the perceived helpfulness of the review. Product type moderates the effect of review extremity on the helpfulness of the review. For experience goods, reviews with extreme ratings are less helpful than reviews with moderate ratings. For both product types, review depth has a positive effect on the helpfulness of the review, but the product type moderates the effect of review depth on the helpfulness of the review. Review depth has a greater positive effect on the helpfulness of the review for search goods than for experience goods. We discuss the implications of our findings for both theory and practice.

Contents

1 Introduction

The reviews are particularly used so that new users can see the rated comments in order to help them make their own purchasing decisions. The quality of the comments made on internet forums has always suffered due to its users. Different websites have tried different methods for extracting more useful comments so that user can see top rated comments.

Amazon's system, in particular, then allows for the higher rated comments to be displayed at the top of the review forum. Even though it is not effective as poor quality reviews can still be seen on top .As a major reason that people are willing to buy consumer goods online without seeing the items themselves, is that they have access to other people opinions of the item to buy the product. This effects the sale on Amazon.

The reason for the failure is part of the algorithm for determining the order of the reviews relies on how recently the review has been made .The system should predict that the comment is helpful or not so that poor quality reviews should not be at the top.

2 Problem Statement

The problem being addressed in this project is the poor quality of Amazon reviews at the top of the forum despite the helpfulness rating system. The solution this problem is to pre-rate the reviews as "helpful" or "not helpful" for new reviews and the most helpful reviews are given position at the top .This way, poor quality reviews will be more unlikely to be shown at the top.

This is a binary classification of reviews on the basis of helpfulness as helpful or not helpful . The models will be trained on the basis of features extracted from the reviews.The recent reviews which are not rated as helpful by the users will be of poor quality but are still on top, we tried to make the helpful review to be at the top of the reviews shown to the buyer.

3 Methodology

3.1 Data Exploration

We converted the given in "json" to a DataFrame in order to perform our analysis.

The data looks like this,

```
In [11]: df = DataFrame(data)
         df.head(3)
```

Out[11]:

	asin	helpful	overall	reviewText	reviewTime	reviewerID	reviewerName	summary	unixReviewTime
0	120401325X	[0, 0]	4.0	They look good and stick good! I just don't li...	05 21, 2014	A30TL5EWN6DFXT	christina	Looks Good	1400630400
1	120401325X	[0, 0]	5.0	These stickers work like the review says they ...	01 14, 2014	ASY55RVNIL0UD	emily l.	Really great product.	1389657600
2	120401325X	[0, 0]	5.0	These are awesome and make my phone look so st...	06 26, 2014	A2TMXE2AFO7ONB	Erica	LOVE LOVE LOVE	1403740800

Figure 1: The Dataset

The data contain ratings for each products from the users rated from one to five based on their experience.

```
In [26]: sns.countplot(divy1['overall'],label = "Count")
         plt.show()
```

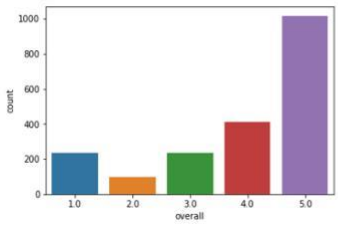

Figure 2: Frequency of rating

The user can either rate the review as ’helpful’ or ‘not helpful’. The dataset records each of these in an array. For our problem we want to classify the new review as either ‘helpful’ or not ‘helpful’. For training, this label can be generated by dividing the ‘helpful’ ratings by the total ratings and seeing if it exceeds a certain thresh-old.

The ’reviewText’ will be used to generate features using natural language processing.

The ’helpful’ rating will be used to generate labels. We will train our model using these training labels.

Predict the label using the test features, and measure the success of our model using the test labels.

```
In [16]: print(divy['helpful_numerator'].idxmax(axis = 0,skipna = True))
         print (divy['helpful_denominator'].idxmax(axis=0, skipna=True))

         64745
         64745

In [17]: divy.iloc[[64745,64746]]

Out[17]:
```

	overall	reviewText	helpful_numerator	helpful_denominator
64745	3.0	This solar charger comes with a one page "manu...	1984	2031
64746	4.0	UPDATE 5/29/12: With the number of unhelpful v...	95	105

Figure 3: Splitting helpful array

We selected the required columns which includes 'ReviewText', 'helpful numerator', 'overall rating' and 'helpful denominator'.

Now let's do a visualization and a count of the data in order to get a sense of the correlation and distribution.

```
In [31]: correlations = divy1.corr()
         plt.figure(figsize = FIG_SIZE)
         plt.title("Heatmap of correlations in each catagory")
         _ =sns.heatmap(correlations, vmin=0, vmax=1, annot=True)
```

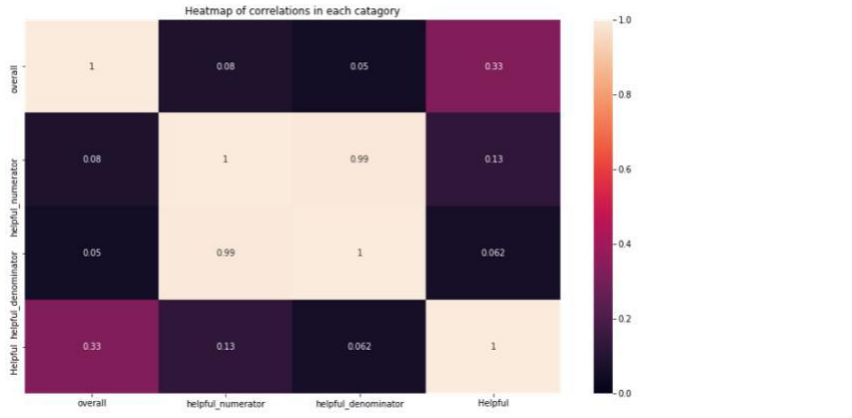

Figure 4: Correlation

3.2 Data Preprocessing

The product which has less than 25 ratings is trimmed from the dataset. As some good reviews which are recent which are not marked as helpful or not helpful will be rated as not helpful.

```
In [19]: divy1 = divy[(divy.helpful_denominator>25)].copy()
         divy1.shape

Out[19]: (1994, 4)
```

Figure 5: Data Selection

To determine whether the review is helpful or not . For this we use a threshold of 'helpful numerator' divided by 'helpful denominator'.If

this ratio exceeds a certain threshold value, we can label the training data as 'helpful' = 1, or 'non-helpful' = 0. For this analysis, the threshold is set to 0.5.

```
In [47]: threshold = 0.5
         divy1.loc[:,'Helpful'] = np.where(divy1.loc[:,'helpful_numerator']\
                                           / divy1.loc[:,'helpful_denominator']>threshold,1,0)
         divy1.head(5)
```

Out[47]:

	overall	reviewText	helpful_numerator	helpful_denominator	Helpful
357	5.0	you may not need these types of screwdrivers o...	38	51	1
390	5.0	i ordered this to replace my iphone battery, i...	27	32	1
448	1.0	these things are just a confidence trick. they...	103	121	1
586	5.0	[apart from minor issues like stability of the...	170	176	1
597	2.0	this phone is flashy and feels good in the han...	19	31	1

Figure 6: Classification on basis of helpfulness

The preprocessing of review text is performed by:

- Stemming-The suffix of words are taken out.
- Tokenizing- Splits sentences up into single words.
- Remove Stop Word-This moves words such as 'the' 'a' and 'it'.
- ngrams- Makes groups of words that are 'n' long.

Finally we will generate TF-IDF scores for each of the stemmed and tokenized words and ngrams. TF-IDF is short for Term Frequency Inverse Document frequency. TF-IDF is a numerical statistic that is intended to react how important a word is to a document in a collection or corpus. It is often used as a weighting factor in information retrieval and text mining.

```
In [38]: df6 = DataFrame(features)
         df6.head()
```

Out[38]:

	0
0	(0, 1376)\t0.14509215909864706\n (0, 2184)\...
1	(0, 1460)\t0.36796392856457266\n (0, 1739)\...
2	(0, 684)\t0.13604856083387 23\n (0, 1529)\t0...
3	(0, 1376)\t0.05121189427783156\n (0, 1409)\...
4	(0, 1376)\t0.19847100329676204\n (0, 1022)\...

Figure 7: Review TF-IDF score

3.3 Benchmark model

The following algorithms were used:

- Logistic Regression.
- Gaussian NB.
- AdaBoost Classifier.
- Random Forest Classifier.
- Decision Tree Classifier.

```
In [39]: from sklearn.model_selection import train_test_split
         X_train,X_test,y_train,y_test = train_test_split(features,divy1['Helpful'],test_size=0.2,random_state = RAN_STATE)
```

Figure 8: Test and Train split

Now we will split the data into 80 percent training and 20 percent testing.

The benefit to splitting the data into testing and training sets is that this allows simulated evaluation of how well the model is performing before using it in the real world to make predictions.

```
In [3]: from sklearn.naive_bayes import GaussianNB
        from sklearn.ensemble import AdaBoostClassifier
        from sklearn.ensemble import RandomForestClassifier
        from sklearn.linear_model import LogisticRegression
        from sklearn.tree import DecisionTreeClassifier
```

Figure 9: Model Selection

Logistic Regression: Logistic regression, despite its name, is a linear model for classification rather than regression. Logistic regression is also known in the literature as logistic regression, maximum-entropy classification (MaxEnt) or the log-linear classifier. In logistic regression, the dependent variable is a binary variable that contains data coded as 1 (yes, success, etc.) or 0 (no, failure, etc.). In other words, the logistic regression model predicts P(Y=1) as a function of X.

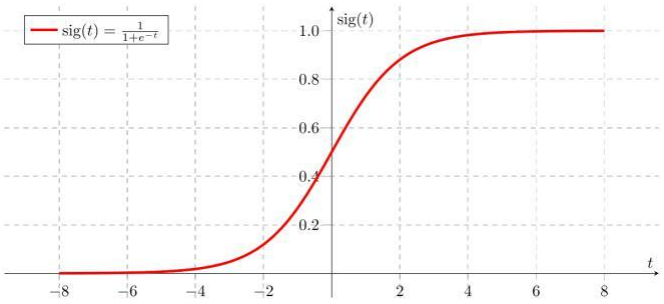

Figure 10: Sigmoid Activation Funtion[5]

If ‘Z’ goes to infinity, Y(predicted) will become 1 and if ‘Z’ goes to negative infinity, Y(predicted) will become 0.

Random Forest: A Random Forest is assembling algorithm that to a number of decision tree classifiers on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over writing. It does not implement boosting on the trees, and instead counts a vote of each individual tree in order to produce the nal class label.

It is a statistical algorithm that is used to cluster points of data in functional groups. When the data set is large and/or there are many variables it becomes di cult to cluster the data because not all variables can be taken into account, therefore the algorithm can also give a certain chance that a data point belongs in a certain group.

It has a variety of applications, such as recommendation engines, image classification and feature selection. It can be used to classify loyal loan applicants, identify fraudulent activity and predict diseases. It lies at the base of the Boruta algorithm, which selects important features in a dataset.

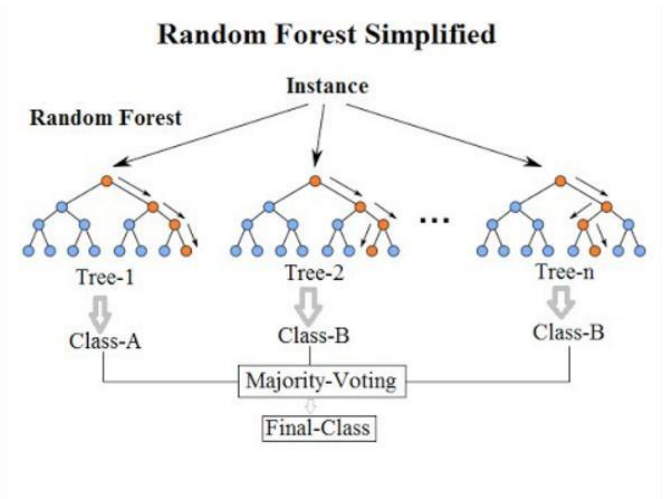

Figure 11: Random Forest Simplified [5]

Adaboost: AdaBoost or "adaptive boosting" begins by splitting a "weak" classifier on the original dataset. It then to additional copies of the classifier on the same dataset and adjusts the weights of incorrectly classified instances such that subsequent classifiers focus more on di cult cases.

AdaBoost classifier builds a strong classifier by combining multiple poorly performing classifiers so that you will get high accuracy strong classifier.

The basic concept behind Adaboost is to set the weights of classifiers and training the data sample in each iteration such that it ensures the accurate predictions of unusual observations. Any machine learning algorithm can be used as base classifier if it accepts weights on the training set. Adaboost should meet two bullet point conditions:

- The classifier should be trained interactively on various weighed training examples.
- In each iteration, it tries to provide an excellent t for these examples by minimizing training error.

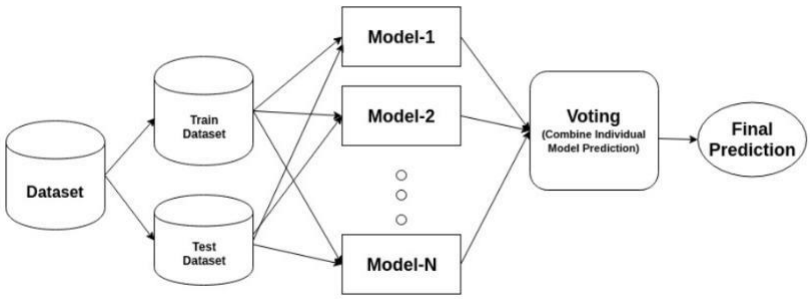

Figure 12: AdaBoost classifier [5]

Gaussian NB: Gaussian Naive Bayes is a statistical classification technique based on Bayes Theorem. It is one of the simplest super-vised learning algorithms. Naive Bayes classifier is the fast, accurate and reliable algorithm. Naive Bayes classifiers have high accuracy and speed on large datasets.

$$P\left(\frac{h}{D}\right) = \frac{P(D|h)P(h)}{P(D)} \quad (1)$$

- P(h): the probability of hypothesis h being true (regardless of the data). This is known as the prior probability of h.
- P(D): the probability of the data (regardless of the hypothesis). This is known as the prior probability.
- P(h|D): the probability of hypothesis h given the data D. This is known as posterior probability.
- P(D|h): the probability of data d given that the hypothesis h was true. This is known as posterior probability.

Decision Tree : A decision tree is a flowchart like tree structure where an internal node represents feature(or attribute), the branch represents a decision rule, and each leaf node represents the out-come. The topmost node in a decision tree is known as the root node. It learns to partition on the basis of the attribute value. It partitions the tree in recursively manner call recursive partitioning. This flowchart-like structure helps you in decision making. It's visualization like a flowchart diagram which easily mimics the human level thinking. That is why decision trees are easy to understand and interpret.

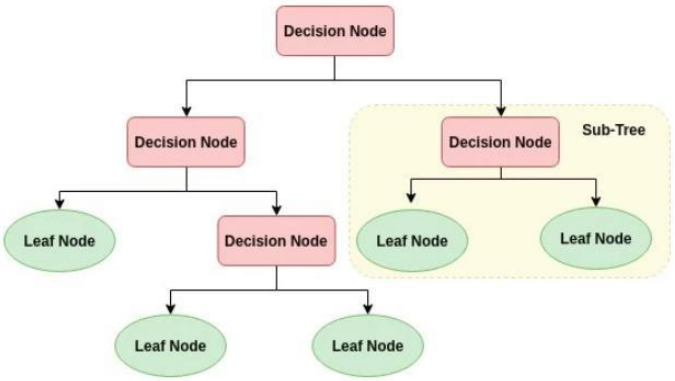

Figure 13: Sample example (image) of Decision [5]

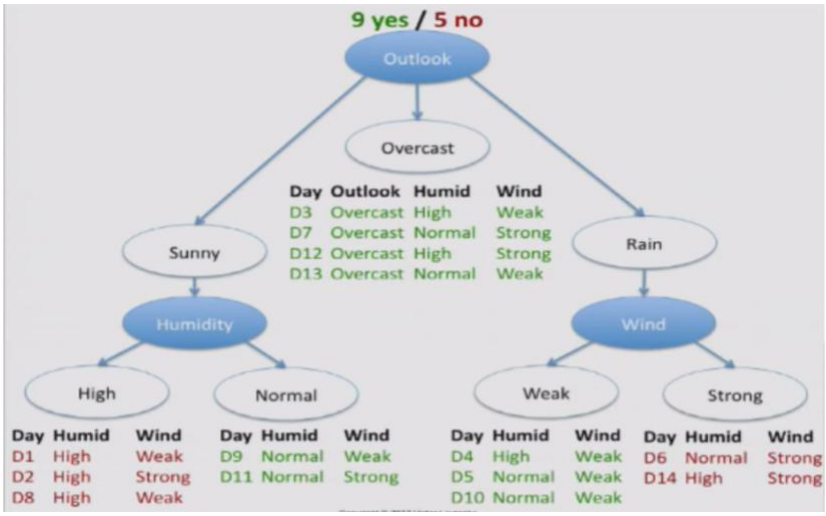

Figure 14: Sample example (image) of Decision [5]

Project Code

```
import pandas as pd
from pandas import DataFrame
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from IPython.display import display

import warnings
warnings.filterwarnings('ignore')
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
pip install -U textblob
import gzip
import seaborn as sns
import string
from time import time
import textblob.exceptions
from textblob import TextBlob
import unicodedata
from six.moves import cPickle as pickle
```

```
import os


import sys
pip install --user -U nltk
import nltk
from nltk.corpus import stopwords
stops = set(stopwords.words("english"))
import numpy as np
import json


data = []
with open(r"C:\Users\tushargpt712\Desktop\Project 1 (1)\Project
1\Cell_Phones_and_Accessories_5.json") as f:
    for line in f:
        data.append(json.loads(line))
data
df = DataFrame(data)
df.head()
tas = df.iloc[:,[2,3,1]]
tas.head()
tas['helpful_numerator'] = df['helpful'].apply(lambda x: x[0])
tas['helpful_denominator'] = df['helpful'].apply(lambda x: x[1])
del tas['helpful']
tas.describe()
print(tas['helpful_numerator'].idxmax(axis = 0,skipna = True))
print (tas['helpful_denominator'].idxmax(axis=0, skipna=True))
tas.iloc[[64745,64746]]
tas.shape
```

```
tas1 = tas[(tas.helpful_denominator>25)].copy()
tas1.shape



print (tas1.isnull().sum())
FIG_SIZE = (14,8)
#tas1 = tas1.iloc[:,[0,2,3]]
#tas1
#plt.figure(figsize=FIG_SIZE)
#plt.title('Box plot of Features')
#plt.ylabel('Spread')
#plt.xlabel('Features')
#display(sns.boxplot(tas1[tas1.columns]))
#display(sns.boxplot(tas.iloc[:,[0,2,3]]))
#sns.boxplot(tas1.iloc[:,[0,2,3]])
tas90 = tas1.groupby('overall').size()
tas90.plot()
plt.grid(True)
plt.show()
sns.countplot(tas1['overall'],label = "Count")
plt.show()
threshold = 0.5
tas1.loc[:,'Helpful'] = np.where(tas1.loc[:,'helpful_numerator']\
                                 /
tas1.loc[:,'helpful_denominator']>threshold,1,0)
tas1.head(10)
tas1.shape
FIG_SIZE = (14,8)
```

```
from sklearn.feature_extraction.text import TfidfVectorizer

from nltk.stem.snowball import SnowballStemmer

correlations = tas1.corr


plt.figure(figsize = FIG_SIZE)

plt.title("Heatmap of correlations in each catagory")

_ =sns.heatmap(correlations, vmin=0, vmax=1, annot=True)

tas1.loc[:,'reviewText'] = tas1['reviewText'].str.lower()

tas1.loc[:,'reviewText'].head()

stemmer = SnowballStemmer("english")

stemmer

def tokens(x):

    x = x.split()

    stems = []

    [stems.append(stemmer.stem(word)) for word in x]

    return stems

vectorizer = TfidfVectorizer(tokenizer = tokens,stop_words =
'english',ngram_range = (1,1),min_df = 0.01)

features = vectorizer.fit_transform(tas1['reviewText'])

features

RAN_STATE = 42

tas1['Helpful'].shape,features.shape

df6 = DataFrame(features)

df6.head()

from sklearn.model_selection import train_test_split

X_train,X_test,y_train,y_test =
train_test_split(features,tas1['Helpful'],test_size=0.2,random_state =
RAN_STATE)

X_train.shape,X_test.shape,y_train.shape,y_test.shape
```

```
from sklearn.metrics import roc_auc_score,roc_curve
def train_classifier(clf,X_train,y_train):
    start = time()

    clf.fit(X_train,y_train)
    end = time()
    print("Trained model in {:.4f} seconds".format(end - start))
def predict_labels(clf,features,target):
    start = time()
    probas = clf.predict_proba(features)
    end = time()
    print("Made predictions in {:.4f} seconds.".format(end-start))
    return roc_auc_score(target.values,probas[:,1].T)
def train_predict(clf, X_train, y_train, X_test, y_test):
    ''' Train and predict using a classifer based on roc_auc score. '''

    # Indicate the classifier and the training set size
    print ("Training a {} using a training set size of {}. .
.".format(clf._class.name_, X_train.shape[0]))

    # Train the classifier
    train_classifier(clf, X_train, y_train)

    # Print the results of prediction for both training and testing
    print ("ROC_AUC score for training set:
{:.4f}.".format(predict_labels(clf, X_train, y_train)))
    print ("ROC_AUC score for test set:
{:.4f}.\n".format(predict_labels(clf, X_test, y_test)))
```

```
def clf_test_roc_score(clf, X_train, y_train, X_test, y_test):
    clf.fit(X_train, y_train)
    probas = probas =clf.predict_proba(X_test)

    return roc_auc_score(y_test, probas[:,1].T)
clf_list = [GaussianNB(),AdaBoostClassifier(random_state =
RAN_STATE),
          RandomForestClassifier(random_state = RAN_STATE),
           LogisticRegression(solver='newton-
cg',multi_class='multinomial',random_state = RAN_STATE),
           DecisionTreeClassifier(random_state = RAN_STATE)]
x_tr = X_train.toarray()
x_te = X_test.toarray()

test_feature_list = [x_te[0:2000],x_te[0:5000],x_te]
test_target_list = [y_test[0:2000], y_test[0:5000], y_test]

# Set up the training set sizes for 100, 200 and 300 respectively.
train_feature_list = [x_tr[0:500],x_tr[0:800],x_tr[0:1400],x_tr]
train_target_list = [y_train[0:500], y_train[0:800],y_train[0:1400],
y_train]



# Execute the 'train_predict' function for each of the classifiers and
each training set size

for clf in clf_list:
    for a, b in zip(train_feature_list, train_target_list):
        train_predict(clf, a, b, x_te, y_test)
```

```
for clf in clf_list:
    x_graph = []
    y_graph = []
   for a, b in zip(train_feature_list, train_target_list):
        y_graph.append(clf_test_roc_score(clf, a, b, x_te, y_test))
        x_graph.append(len(a))
    plt.scatter(x_graph,y_graph)
    plt.plot(x_graph,y_graph, label = clf._class.name_)


plt.title('Comparison of Different Classifiers')
plt.xlabel('Training Size')
plt.ylabel('ROC_AUC score on training set')
plt.legend(bbox_to_anchor=(1.6, 1.05))
plt.figure(figsize=FIG_SIZE)
plt.show()
```

4 Simulations and Results

We tested the above classifiers using above function. All of the classifiers from sklearn will be tested; we will train and test a bunch of the classifiers on four different training sizes in order to select which one will be our benchmark.

```
In [44]: clf_list = [GaussianNB(),AdaBoostClassifier(random_state = RAN_STATE),
                    RandomForestClassifier(random_state = RAN_STATE),
                     LogisticRegression(solver='newton-cg',multi_class='multinomial',random_state = RAN_STATE),
                     DecisionTreeClassifier(random_state = RAN_STATE)]
         x_tr = X_train.toarray()
         x_te = X_test.toarray()

         # Set up the training set sizes for 100, 200 and 300 respectively.
         train_feature_list = [x_tr[0:500],x_tr[0:800],x_tr[0:1400],x_tr]
         train_target_list = [y_train[0:500], y_train[0:800],y_train[0:1400], y_train]


         # Execute the 'train_predict' function for each of the classifiers and each training set size

         for clf in clf_list:
             for a, b in zip(train_feature_list, train_target_list):
                 train_predict(clf, a, b, x_te, y_test)
```

Figure 15: Training and Testing

The results obtained are as follows:

Table 1: Model and Training set size accuracy table

Model	Training Set Size	Train Set Score	Test Set Score
GaussianNB	500	0.9958	0.4908
	800	0.9934	0.4894
	1400	0.9906	0.5106
	1590	0.9888	0.5330
AdaBoostClassifier	500	1.0000	0.6894
	800	1.0000	0.6703
	1400	0.9981	0.7588
	1590	0.9933	0.7536
RandomForestClassifier	500	1.0000	0.5542
	800	1.0000	0.5858
	1400	1.0000	0.5477
	1590	1.0000	0.6147
LogisticRegression	500	1.0000	0.8285
	800	.9984	0.8509
	1400	0.9973	0.8281
	1590	0.9967	0.8533
DecisionTreeClassifier	500	1.0000	0.5420
	800	1.0000	0.5420
	1400	1.0000	0.5460
	1590	1.0000	0.5763

We found that Logistic Regression classifier did the best and most suitable with maximum testing set score as 0.8533 and GaussianNb is least suitable with test set score 0.5330

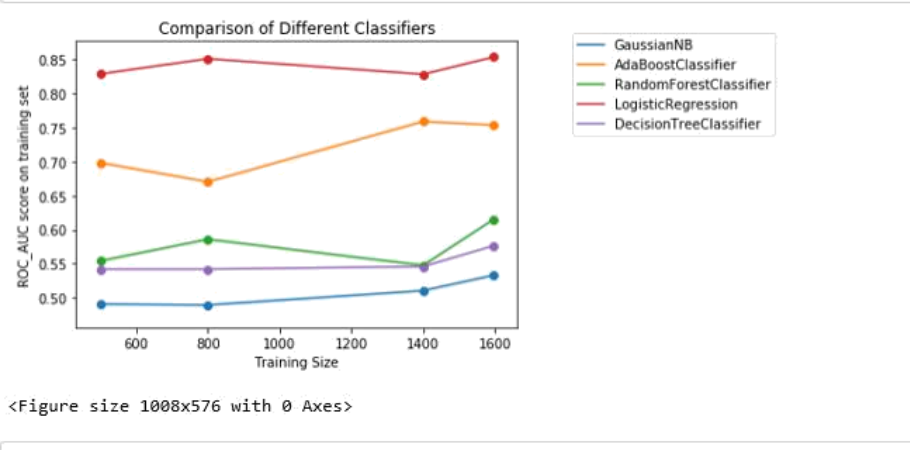

Figure 16: Comparison of Different Classifiers

The accuracy score for test set for four different sizes of train dataset is plotted for all models which shows Logistic Regression as the most suitable and GaussianNB as least suitable.

5 Conclusions and Future Work

Our model takes in existing 'reviewText' and transforms it into numerical TF-IDF scores. It then adds the existing 'overall' score of the reviews to create a features set for each review. It trains a Logistic Regression model using labels generated by taking existing 'helpful-ness numerator' data and dividing it by 'helpfulness denominator' data and thresholding the result at 0.5. The products which have less than 25 reviews are not considered for training the classifier. New reviews with no helpfulness data which are recently added are classified as being 'helpful' or 'non-helpful'. By using this system, Amazon can work to make sure that more helpful reviews are shown at the top of their forums.

Mainly, we looked at the TFIDF features generated from amazon review text and added the 'overall rating' that was given to the product by the reviewer. We used these features to predict how 'helpful' other users would give the review.

In order to improve the accuracy of our model, the following actions could be taken:

- There are lots of spelling mistakes in the reviews and abbreviations are used. This would result in a model that potentially has less features, as certain spelling errors would have been corrected/eliminated.

- Reviews with poor grammar, punctuation and improper word endings are more di cult to understand and would possibly lead to people rating them as less helpful. These reviews may contain useful information's and may be helpful but certain other factors made them rated as not helpful. Thus preprocessing of data can be improved either by varying the values of threshold or the reviews that has not been selected because the product is not much rated.

Bibliography

[1] Liaw A. Weiner M., Classification and Regression by random-Forest. R News. 2002;Vol 2(2):18-22.

[2] J. McAuley, R. Pandey, J. Leskovec Knowledge Discovery and Data Mining, 2015.

[3] HUI BWU Y. Anti-spam model based on semi-Naive Bayesian classification model. Journal of Computer Applications. 2009;29(3):903-904.

[4] Quinlan, J. R. (1986). Induction of decision trees. Machine learning, 1(1), 81-106.

[5] https://drrajeshkumar.wordpress.com/downloads/