Fake Review Detection in Amazon Reviews

Project-I (NA47005) report submitted to

Indian Institute of Technology Kharagpur

in partial fulfilment for the award of the degree of

Bachelor of Technology

in

Ocean Engineering and Naval Architecture

by

Duvvu Avinash

(16NA30028)

Under the supervision of

Professor Sujoy Bhattacharya

Vinod Gupta School of Management

Indian Institute of Technology Kharagpur

Autumn Semester, 2019-20

November, 2019

DECLARATION

I certify that

(a) The work contained in this report has been done by me under the guidance of my supervisor.

(b) The work has not been submitted to any other Institute for any degree or diploma.

(c) I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.

(d) Whenever I have used materials (data, theoretical analysis, figures, and text) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references. Further, I have taken permission from the copyright owners of the sources, whenever necessary.

Date: November, 2019 (Duvvu Avinash)

Place: Kharagpur (16NA30028)

VINOD GUPTA SCHOOL OF MANAGEMENT
INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR
KHARAGPUR - 721302, INDIA

CERTIFICATE

This is to certify that the project report entitled "**Fake Review Detection in Amazon Reviews**" submitted by **Duvvu Avinash** (Roll No. 16NA30028) to Indian Institute of Technology Kharagpur towards partial fulfilment of requirements for the award of degree of Bachelor of Technology in Ocean Engineering and Naval Architecture is a record of bona fide work carried out by him under my supervision and guidance during Autumn Semester, 2019-20.

Date: November, 2019
Place: Kharagpur

Professor Sujoy Bhattacharya
Vinod Gupta School of Management
Indian Institute of Technology Kharagpur
Kharagpur - 721302, India

Abstract

Name of the student: **Duvvu Avinash** Roll No: **16NA30028**

Degree for which submitted: **Bachelor of Technology**

Department: **Vinod Gupta School of Management**

Thesis title: **Fake Review Detection in Amazon Reviews**

Thesis supervisor: **Professor Sujoy Bhattacharya**

Month and year of thesis submission: **November, 2019**

The impact of online reviews on businesses has grown significantly during last years, being crucial to determine business success in a wide array of sectors, ranging from restaurants, hotels to e-commerce. Unfortunately, some users use unethical means to improve their online reputation by writing fake reviews of their businesses or competitors.The purpose of this project is to contribute to the growing body of studies on identification of fake reviews and reviewers by building a model that classifies the reviews as real or fake from the multiple layers of Amazon product and review data to identify fake reviews,the model built used linguistic features and reviews are discretized into 3 categories based on review rating less than 3 and greater than 3 and equal to three so the problem is simplified now to a 3 class 2 labelled.

Acknowledgements

I place on record, my sincere thanks to Prof.Sujoy Bhattacharya, supervisor for this project, for his suggestions and guidance,continuous assessment, inputs, sharing expertise, and sincere and advice and encouragement extended to me.

Contents

Chapter 1

Project Description

1.1 Introduction

Online consumer product reviews are playing an increasingly important role for customers, constituting a new type of WOM information,Recent research shows that 52percent of online consumers use the Internet to search for product information(1), while 24percent of them use the Internet to browse products before making purchases As a result,online reviews has a strong impact on consumers' decision purchase in e-commerce, affecting the most relevant areas, such as travel and accommodations Moreover, online reviews of the same product can be found in multiples sources of information, which can be classified according to the parties that host WOM information into internal WOMs, hosted by retailers (e.g. Amazon, Walmart, Best-Buy, etc.) and external ones, hosted by independent product review providers (e.g. CNET, Yelp, TripAdvisor, Epinions, etc.).Individuals analyze reviews by focusing not only on the summary (i.e., star ratings) but also on the content of customers' natural texts based on subjectively experienced intangibles Nevertheless, only credible reviews have a significant impact on consumers' purchase decision Moreover, product category affects significantly the credibility of WOMs consumer electronics are the product most influenced by online reviews, influencing the 24 of products acquired in this category, and being WOMs the second most influential source after search engines in this product category. On the other hand, consumers tend to research on consumer electronics products because these products change very

frequently, with new products and updates of existing ones.In the case of the hospitality and tourism industry, consumers trust reviews as they are independent from official or corporate information and they show the previous experience of other travellers using their own words.However, the concept of a credible source in Social Media is different to that of traditional word-of-mouth, where the source is someone belonging to the inner circle of the consumer. In the case of online reviews, users are anonymous and only identified by an alias. One possible source of credibility is the reviewer's reputation,which is rated by other users.malicious users can easily manipulate online reputation systems, as the reputation is based on simple rules that can be distorted through false accounts that perform false scoring to artificially improve their trust worthiness

Given the importance of reviews for businesses and the difficulty of obtaining a good reputation on the Internet, several techniques have been used to improve online presence, including unethical ones. Fake reviews are one of the most popular unethical methods which are present on sites such as Yelp or TripAdvisor.We apply a machine learning approach based on extracting the best features of the reviews to classify online hotel reviews as positive or negative.

Related Work The major motivations behind the project are: (1) The task of fake review detection has been studied since 2007, with the analysis of review spamming (Jindal Liu, 2007b). In this work, the authors analyzed the case of Amazon, concluding that manually labeling fake reviews may result challenging, as fake reviewers could carefully craft their reviews in order to make them more reliable for other users. Consequently, they proposed the use of duplicates or nearly-duplicates as spam in order to develop a model that detects fake reviews (Jindal Liu, 2007b). Research on distributional footprints has also been carried out, showing a connection between distribution anomalies and deceptive reviews from Amazon products and TripAdvisor hotels

(12)Fake review detection is a specific application of the general problem of deception detection, where both verbal and nonverbal clues can be used Fake review detection research has mainly exploited textual and behavioral features, while other approaches have taken into account social or temporal aspects.

(7)Previous literature has considered three main different approaches for obtaining relevant features of fake reviews: linguistic or review centric methods, reviewer centric, and network approaches,Reviewer centric approaches focus on features collected from reviewer profile characteristics and behavioural patterns. Features such as the number of reviews, the percentage of positive reviews, the deviation from the average review rating, the review length, and the presence of similar reviews for different products by the same reviewer, or the variety of products or services where the reviewer is posting reviews,are considered,Finally, the network approaches refer to the analysis of interdependencies through the links or edges between objects (either reviewers or reviews) to obtain the behaviour of users in online reviews and eWOM websites.(6) The interactions are modelled as a social network,and the micro and macro analysis can then reveal suspicious behaviour that can be associated to fraudsters Method based on Content of reviews:-the similarity between the linguistic features are identified in the reviews,like the similarity of vocabulary used cotent consistency,by sentiment tendencies and sematics we can find the deception detection (9)using crowdsourcing platform (AMT) to construct datasets and used comprehensive method of natural language to acquire linguistic features from multiple perspectives. They trained many kinds of classifiers and compared their performance.on the business data set the test results were not very well (11)using naive Bayesian machine learning algorithm for deceptive reviews detection. A two-sided cotraining semisupervised learning method was proposed to mark a large number of unlabelled reviews. And they used it as follow-up deceptive reviews test datasets (2)this method incorporated new features with semi supervised learning methods to increment of F-Score (3)proved that deep syntactic information of texts is very effective in deceptive reviews detection. They used probabilistic context-free syntax PCFG. The deep syntactic features of the reviews texts are generated by the generative rules of the PCFG syntax analysis tree and the SVM classifier is trained to identify the deceptive reviews. Method based on Behaviour:-In this method, most of the features are extracted based on the metadata of the reviews (time of reviews, frequency of reviews, information of the first reviewers of the product, etc.),(5) focused on the behavior of reviewers to find the deceptive reviews. They considered that it was better to study reviewers than reviews because the information obtained from the reviewers' . So they proposed a method to detect the deceptive reviewers based on the score of reviewers. They constructed a model from the multifaceted

behaviors of reviewers, and designed a deceptive degree scoring function to calculate whether the reviewers are deceptive. (13)proposed a multi-time scale detection method and found time windows that concentratedly distributed deceptive reviews through time series analysis. They considered that the singleton review in such time windows is highly likely to be deceptive, where singleton review means that the reviewer of the review posted only this one review. Their method that makes use of features such as the release date of the review and the historical record of the reviewer is an unsupervised learning method. (8)proposed an unsupervised model of hypothetical reviewers named ASM. They considered the distribution of different behaviors of deceptive reviewers and normal reviewers and set falsehood as an implicit variable and reviewers' behavior as an observation variable. They used a clustering algorithm to identify fake reviewers to identify deceptive reviews. Method based on Relationship:-the method studies the complex relation between users social media connections are taken into account,graph based approach is used to form the network of the relation ship among the users

Hybrid methods combining previous approaches have also been treated in the previous works. For instance,added social features (friends,followers, votes...)to the reviewer centric features in the case of consumer electronics.frequent methods reported in the literature: linear/logisticregression models, naive Bayesian models, SVM, nearest-neighbour algorithms (such as k-NN), least squares, ensembles of classifiers and multi-layer perceptrons

1.2 Objective

For the sake brevity, we state the objectives the project here:

1. A model model that classifies the reviews as real or fake. Used both the review text and the additional features,the additional features containes classes of rating less than 3, greater than 3 and eqaual to 3 without using any deep learning techniques..

Chapter 2

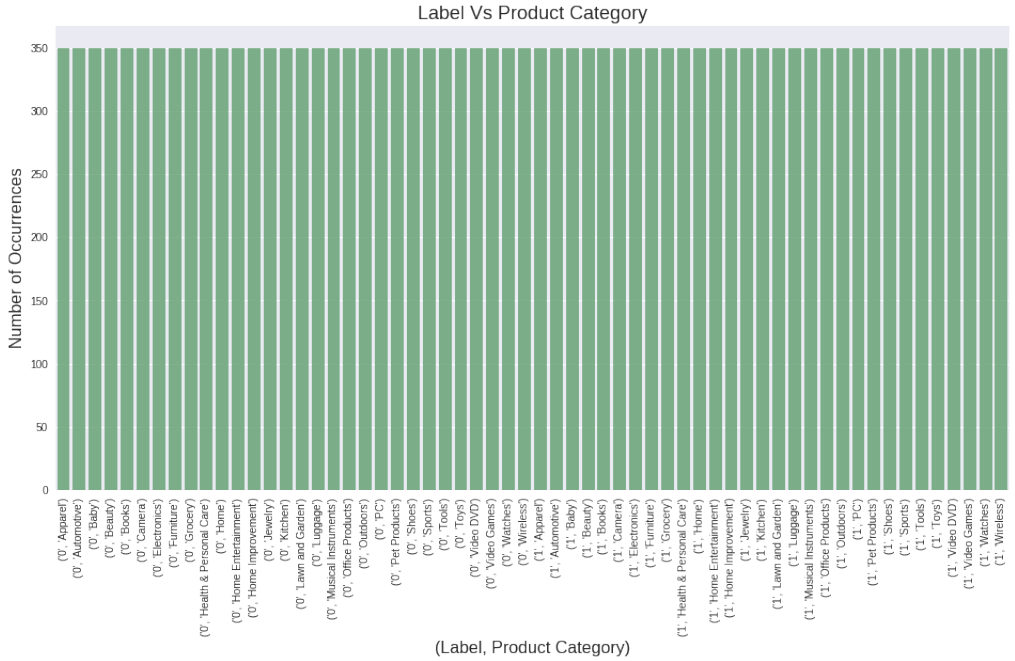
Data acquisition and cleansing

2.1 Introduction

the data is from amazon corpus released,it has porduct related features,text review and rating on a scale of five,labels,verified-purchase,some of the products are electronics,books,dvds,video games etc,it has total of 60 product categories,total entries of the products are 21000

2.2 Data visualisation

after loading the amazon review data set we start visualizing using matplotlib,seaborn,initially we see the number of products in the dataset there were a total of 60 product types each have 350 different entries,each product is labelled 1 and 0 in the dataset,we plot label vs product category



(10) we group the data according to the rating values and found out that

label.no	rating	count
0	5	6151
0	4	1974
0	3	942
0	1	868
0	2	565
1	5	6059
1	4	1999
1	3	926
1	1	889
1	2	627

TABLE 2.1: rating counts for 2 labels

we visualize according to rating and product category

next we see label count based on verified label

the text data is being processed for counting the text length for label 0,and label 1 label 0 has an average length of 428.102,and label 1 has an average length of 316.55

verified*label*	label	count
N	1	7623
	0	1679
Y	0	8821
	1	2877

TABLE 2.2: label count based on verification

2.3 Data preprocessing

new columns named text-length,num-sentences,Fk-score is added,the average fk-score of label 0 is 5.84,label 1 is 5.79

```
data['TEXT_LENGTH'] = data['REVIEW_TEXT'].apply(len)
data['num_sentences'] = data['REVIEW_TEXT'].apply(lambda x: len(str
    (x).split('.')))
from textstat.textstat import textstat
data["FK_Score"] = data["REVIEW_TEXT"].apply(textstat.
    flesch_kincaid_grade)
```

LISTING 2.1: adding extra features

we use nltk module for adding features stop-count,caps-count,punct-count (4)

```
import nltk
wpt = nltk.WordPunctTokenizer()
stop_words = nltk.corpus.stopwords.words('english')
def stopCount(x):
    sum =0
    for char in x.split():
        sum+= char in stop_words
    return sum
data['stop_count'] = data['REVIEW_TEXT'].apply(stopCount)
def capsCount(x):
    sum =0
    for char in x:
        sum+= char in "QWERTYUIOPASDFGHJKLZXCVBNM"
    return sum
data['caps_count'] = data['REVIEW_TEXT'].apply(capsCount)
import string
count = lambda l1,l2: sum([1 for x in l1 if x in l2])
```

```
def punctCount(x):
    return count(x, set(string.punctuation))
data['punct_count'] = data['REVIEW_TEXT'].apply(punctCount)
```

Listing 2.2: adding extra features

we use extra features matchesDF for product name matches count

```
import re
import string
match_list = []

def checkName(title,text):
    matches = []
    for word in title.split():
        #removing punctuation
        word = "".join((char for char in word if char not in string
    .punctuation))
        #print(word)
        myreg = r'\b'+word+r'\b'
        r = re.compile(myreg, flags=re.I | re.X)
        matches.append(r.findall(text))
    return len(matches)


for a,b in zip(data.PRODUCT_TITLE, data.REVIEW_TEXT):
    number_of_matches = checkName(a,b)
    match_list.append(number_of_matches)

data["matchesDf"] = match_lis

```

Listing 2.3: feature engineering

from review text we can also obtain the emojis used in the review we add another column for emojis

```
data["emojis"] = data["REVIEW_TEXT"].apply(lambda x: 1 if ";)" in
  x.split() or ":)" in x.split() or ":-)" in x.split() else 0)

```

Listing 2.4: adding extra features

we use sentiment classifier for rating rating values less than 3 and greater than 3 and rating value is eaul to 3

```
data.loc[data["RATING"] < 3, "RATING"] = 0
data.loc[data["RATING"] > 3, "RATING"] = 1
data1 = data.loc[data['RATING'] == 1]
data2 = data1.sample(frac=0.2, replace=True)
data3 = data1 = data.loc[data['RATING'] == 0]
data4 = pd.concat([data2, data3], ignore_index=True)
```

Listing 2.5: Sentiment classifier

the raw data with features "Review text","Rating",""PRODUCT-CATEGORY", "LABEL", "RATING" are taken and vectorized

```
rawData = data4[["REVIEW_TEXT", "VERIFIED_PURCHASE", "
    PRODUCT_CATEGORY", "LABEL", "RATING"]]
rawData = [tuple(x) for x in rawData.values]
table = str.maketrans({key: None for key in string.punctuation})
def preProcess(text):
    # Should return a list of tokens
    lemmatizer = WordNetLemmatizer()
    filtered_tokens=[]
    lemmatized_tokens = []
    stop_words = set(stopwords.words('english'))
    text = text.translate(table)
    for w in text.split(" "):
        if w not in stop_words:
            lemmatized_tokens.append(lemmatizer.lemmatize(w.lower()
    ))
        filtered_tokens = [' '.join(l) for l in nltk.bigrams(
    lemmatized_tokens)] + lemmatized_tokens
    return filtered_tokens

featureDict = {} # A global dictionary of features

def toFeatureVector(tokens, verified_Purchase, product_Category,
    labels):
    localDict = {}

#Labels

```

```
    featureDict["L"] = 1
    localDict["L"] = labels

#Verified_Purchase

    featureDict["VP"] = 1

    if verified_Purchase == "N":
        localDict["VP"] = 0
    else:
        localDict["VP"] = 1

#Product_Category


    if product_Category not in featureDict:
        featureDict[product_Category] = 1
    else:
        featureDict[product_Category] = +1

    if product_Category not in localDict:
        localDict[product_Category] = 1
    else:
        localDict[product_Category] = +1


#Text

    for token in tokens:
        if token not in featureDict:
            featureDict[token] = 1
        else:
            featureDict[token] = +1

        if token not in localDict:
            localDict[token] = 1
        else:
            localDict[token] = +1

    return localDict
```

LISTING 2.6: Feature Vectorization

the data is being split in 80:20 ration for training and testing now we train and test the results with SVM,logistic regression,Random forest,decision tree,Xgboost classifiers and check their precision score with cross validation

```
def splitData(percentage):
    dataSamples = len(rawData)
    halfOfData = int(len(rawData)/2)
    trainingSamples = int((percentage*dataSamples)/2)
    for (Text, verified_Purchase, product_Category, Label, Rating)
   in rawData[:trainingSamples] + rawData[halfOfData:halfOfData+
   trainingSamples]:
        trainData.append((toFeatureVector(preProcess(Text),
   verified_Purchase, product_Category, Label), Rating))
    for (Text, verified_Purchase, product_Category, Label, Rating)
   in rawData[trainingSamples:halfOfData] + rawData[halfOfData+
   trainingSamples:]:
        testData.append((toFeatureVector(preProcess(Text),
   verified_Purnchase, product_Category, Label), Rating))
```

LISTING 2.7: splitting the dataset

2.4 data training

```
from sklearn.svm import NuSVC
def trainClassifier(trainData):
    print("Training Classifier...")
    pipeline =  Pipeline([('svc', NuSVC(nu=0.5,kernel='rbf',gamma='
   scale'))])
    return SklearnClassifier(pipeline).train(trainData)
#rawData = []           # the filtered data from the dataset file (
   should be 21000 samples)
#preprocessedData = [] # the preprocessed reviews (just to see how
   your preprocessing is doing)
trainData = []         # the training data as a percentage of the
   total dataset (currently 80%, or 16800 samples)
```

```
9  testData = []
10
11 #print("Now %d rawData, %d trainData, %d testData" % (len(rawData),
       len(trainData), len(testData)),
12  #      "Preparing the dataset...",sep='\n')
13
14 # We split the raw dataset into a set of training data and a set of
       test data (80/20)
15 print("Now %d rawData, %d trainData, %d testData" % (len(rawData),
      len(trainData), len(testData)),
16        "Preparing training and test data...",sep='\n')
17 splitData(0.8)
18 # We print the number of training samples and the number of
      features
19 print("Now %d rawData, %d trainData, %d testData" % (len(rawData),
      len(trainData), len(testData)),
20        "Training Samples: ", len(trainData), "Features: ", len(
      featureDict), sep='\n')
21 print("Mean of cross-validations (Accuracy, Precision, Recall,
      Fscore): ", crossValidate(trainData, 10))
22 Now 6186 rawData, 0 trainData, 0 testData
23 Preparing training and test data...
24 Now 6186 rawData, 4948 trainData, 1238 testData
25 Training Samples:
26 4948
27 Features:
28 178852
29 Training Classifier...
30 Training Classifier...
31 Training Classifier...
32 Training Classifier...
33 Training Classifier...
34 Training Classifier...
35 Training Classifier...
36 Training Classifier...
37 Training Classifier...
38 Training Classifier...
39 Training Classifier...
40 Mean of cross-validations (Accuracy, Precision, Recall, Fscore):
      [0.87246964 0.87217168 0.87268231 0.8720101 ]
```

Listing 2.8: SVM classifier

```
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
def trainClassifier(trainData):
    print("Training Classifier...")
    pipeline =  Pipeline([('scaler',StandardScaler(with_mean=False)),('rf',RandomForestClassifier(n_estimators=100))])
    return SklearnClassifier(pipeline).train(trainData)
#rawData = []           # the filtered data from the dataset file (should be 21000 samples)
#preprocessedData = [] # the preprocessed reviews (just to see how your preprocessing is doing)
trainData = []         # the training data as a percentage of the total dataset (currently 80%, or 16800 samples)
testData = []

#print("Now %d rawData, %d trainData, %d testData" % (len(rawData), len(trainData), len(testData)),
 #      "Preparing the dataset...",sep='\n')

# We split the raw dataset into a set of training data and a set of test data (80/20)
print("Now %d rawData, %d trainData, %d testData" % (len(rawData), len(trainData), len(testData)),
      "Preparing training and test data...",sep='\n')
splitData(0.8)
# We print the number of training samples and the number of features
print("Now %d rawData, %d trainData, %d testData" % (len(rawData), len(trainData), len(testData)),
      "Training Samples: ", len(trainData), "Features: ", len(featureDict), sep='\n')
print("Mean of cross-validations (Accuracy, Precision, Recall, Fscore): ", crossValidate(trainData, 10))
Now 6186 rawData, 0 trainData, 0 testData
Preparing training and test data...
Now 6186 rawData, 4948 trainData, 1238 testData
Training Samples:
```

```
4948
Features:
178852
Training Classifier...
Training Classifier...
Training Classifier...
Training Classifier...
Training Classifier...
Training Classifier...
Training Classifier...
Training Classifier...
Training Classifier...
Training Classifier...
Mean of cross-validations (Accuracy, Precision, Recall, Fscore):
    [0.85645933 0.85711762 0.85526385 0.85542759]
```

LISTING 2.9: RandomForestClassifier

```
from sklearn.linear_model import LogisticRegression
def trainClassifier(trainData):
    print("Training Classifier...")
    pipeline = Pipeline([('scaler', StandardScaler(with_mean=False)
    ), ('classifier',LogisticRegression(random_state=42))])
    return SklearnClassifier(pipeline).train(trainData)
#rawData = []           # the filtered data from the dataset file (
    should be 21000 samples)
#preprocessedData = [] # the preprocessed reviews (just to see how
    your preprocessing is doing)
trainData = []         # the training data as a percentage of the
    total dataset (currently 80%, or 16800 samples)
testData = []

#print("Now %d rawData, %d trainData, %d testData" % (len(rawData),
    len(trainData), len(testData)),
 #     "Preparing the dataset...",sep='\n')

# We split the raw dataset into a set of training data and a set of
    test data (80/20)
print("Now %d rawData, %d trainData, %d testData" % (len(rawData),
    len(trainData), len(testData)),
```

```
16       "Preparing training and test data...",sep='\n')
17 splitData(0.8)
18 # We print the number of training samples and the number of
       features
19 print("Now %d rawData, %d trainData, %d testData" % (len(rawData),
       len(trainData), len(testData)),
20       "Training Samples: ", len(trainData), "Features: ", len(
       featureDict), sep='\n')
21 print("Mean of cross-validations (Accuracy, Precision, Recall,
       Fscore): ", crossValidate(trainData, 10))
22 Now 6186 rawData, 0 trainData, 0 testData
23 Preparing training and test data...
24 Now 6186 rawData, 4948 trainData, 1238 testData
25 Training Samples:
26 4948
27 Features:
28 178852
29 Mean of cross-validations (Accuracy, Precision, Recall, Fscore):
       [0.83925285 0.84162259 0.8347935  0.83584391]
```

Listing 2.10: logistic regression classifier

```
1  from xgboost.sklearn import XGBClassifier
2  def trainClassifier(trainData):
3      print("Training Classifier...")
4      pipeline = Pipeline([('scaler', StandardScaler(with_mean=False)
       ), ('classifier', XGBClassifier())])
5      return SklearnClassifier(pipeline).train(trainData)
6  #rawData = []           # the filtered data from the dataset file (
       should be 21000 samples)
7  #preprocessedData = [] # the preprocessed reviews (just to see how
       your preprocessing is doing)
8  trainData = []         # the training data as a percentage of the
       total dataset (currently 80%, or 16800 samples)
9  testData = []
10
11 #print("Now %d rawData, %d trainData, %d testData" % (len(rawData),
       len(trainData), len(testData)),
12 #      "Preparing the dataset...",sep='\n')
13
```

```
# We split the raw dataset into a set of training data and a set of test data (80/20)
print("Now %d rawData, %d trainData, %d testData" % (len(rawData), len(trainData), len(testData)),
      "Preparing training and test data...",sep='\n')
splitData(0.8)
# We print the number of training samples and the number of features
print("Now %d rawData, %d trainData, %d testData" % (len(rawData), len(trainData), len(testData)),
      "Training Samples: ", len(trainData), "Features: ", len(featureDict), sep='\n')
print("Mean of cross-validations (Accuracy, Precision, Recall, Fscore): ", crossValidate(trainData, 10))
Now 6186 rawData, 0 trainData, 0 testData
Preparing training and test data...
Now 6186 rawData, 4948 trainData, 1238 testData
Training Samples:
4948
Features:
178852
Training Classifier...
Training Classifier...
Training Classifier...
Training Classifier...
Training Classifier...
Training Classifier...
Training Classifier...
Training Classifier...
Training Classifier...
Training Classifier...
Training Classifier...
Mean of cross-validations (Accuracy, Precision, Recall, Fscore):
   [0.78537909 0.78500849 0.78630599 0.78428345]
```

Listing 2.11: XGBoost classifier

```
from sklearn import tree
def trainClassifier(trainData):
    print("Training Classifier...")
```

```
4     pipeline = Pipeline([('scaler', StandardScaler(with_mean=False)
      ), ('classifier',tree.DecisionTreeClassifier(random_state=42))])
5     return SklearnClassifier(pipeline).train(trainData)
6 #rawData = []           # the filtered data from the dataset file (
      should be 21000 samples)
7 #preprocessedData = [] # the preprocessed reviews (just to see how
      your preprocessing is doing)
8 trainData = []         # the training data as a percentage of the
      total dataset (currently 80%, or 16800 samples)
9 testData = []
10
11 #print("Now %d rawData, %d trainData, %d testData" % (len(rawData),
       len(trainData), len(testData)),
12  #      "Preparing the dataset...",sep='\n')
13
14 # We split the raw dataset into a set of training data and a set of
       test data (80/20)
15 print("Now %d rawData, %d trainData, %d testData" % (len(rawData),
      len(trainData), len(testData)),
16       "Preparing training and test data...",sep='\n')
17 splitData(0.8)
18 # We print the number of training samples and the number of
      features
19 print("Now %d rawData, %d trainData, %d testData" % (len(rawData),
      len(trainData), len(testData)),
20       "Training Samples: ", len(trainData), "Features: ", len(
      featureDict), sep='\n')
21 print("Mean of cross-validations (Accuracy, Precision, Recall,
      Fscore): ", crossValidate(trainData, 10))
22 Now 6186 rawData, 0 trainData, 0 testData
23 Preparing training and test data...
24 Now 6186 rawData, 4948 trainData, 1238 testData
25 Training Samples:
26 4948
27 Features:
28 178852
29 Training Classifier...
30 Training Classifier...
31 Training Classifier...
32 Training Classifier...
33 Training Classifier...
```

s.No	classifier	Fscore
1	SVM	87.2
2	RandomForest	85.5
3	logistic-regression	83.5
4	Xgboost	78.4
5	Decision tree	76.05

TABLE 2.3: F-scores of classifiers

```
34 Training Classifier...
35 Training Classifier...
36 Training Classifier...
37 Training Classifier...
38 Training Classifier...
39 Training Classifier...
40 Mean of cross-validations (Accuracy, Precision, Recall, Fscore):
       [0.76380199 0.76297341 0.76083832 0.76058941]
```

LISTING 2.12: decision tree classifier

2.5 end results

the Fscore is highest for SVM 87percent,lowest for decision tree classifier 76 percent

2.6 Future Works

1. working with deep learning techniques on fake review classification,and fake news detection

2. natural language processing is used in for behaviourial identification of the review similarity in case of spam reviews

Bibliography

[1] Barbado, R., Araque, O., and Iglesias, C. A. A framework for fake review detection in online consumer electronics retailers. *Information Processing & Management 56*, 4 (2019), 1234–1244.

[2] Feng, S., Banerjee, R., and Choi, Y. Syntactic stylometry for deception detection. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2* (2012), Association for Computational Linguistics, pp. 171–175.

[3] Feng, S., Xing, L., Gogar, A., and Choi, Y. Distributional footprints of deceptive product reviews. In *Sixth International AAAI Conference on Weblogs and Social Media* (2012).

[4] Garg, P., and Bassi, V. G. *Sentiment analysis of twitter data using NLTK in python.* PhD thesis, 2016.

[5] Lim, E.-P., Nguyen, V.-A., Jindal, N., Liu, B., and Lauw, H. W. Detecting product review spammers using rating behaviors. In *Proceedings of the 19th ACM international conference on Information and knowledge management* (2010), ACM, pp. 939–948.

[6] Mariani, M. M., Borghi, M., and Kazakov, S. The role of language in the online evaluation of hospitality service encounters: An empirical study. *International Journal of Hospitality Management 78* (2019), 50–58.

[7] Martinez-Torres, M., and Toral, S. A machine learning approach for the identification of the deceptive reviews in the hospitality sector using unique attributes and sentiment orientation. *Tourism Management 75* (2019), 393–403.

[8] Mukherjee, A., Kumar, A., Liu, B., Wang, J., Hsu, M., Castellanos, M., and Ghosh, R. Spotting opinion spammers using behavioral footprints. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining* (2013), ACM, pp. 632–640.

[9] Ott, M., Choi, Y., Cardie, C., and Hancock, J. T. Finding deceptive opinion spam by any stretch of the imagination. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1* (2011), Association for Computational Linguistics, pp. 309–319.

[10] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. Scikit-learn: Machine learning in python. *Journal of machine learning research 12*, Oct (2011), 2825–2830.

[11] Rout, J. K., Dalmia, A., Choo, K.-K. R., Bakshi, S., and Jena, S. K. Revisiting semi-supervised learning for online deceptive review detection. *IEEE Access 5* (2017), 1319–1327.

[12] Sánchez-Franco, M. J., Navarro-García, A., and Rondán-Cataluña, F. J. A naive bayes strategy for classifying customer satisfaction: A study based on online reviews of hospitality services. *Journal of Business Research 101* (2019), 499–506.

[13] Xie, S., Wang, G., Lin, S., and Yu, P. S. Review spam detection via temporal pattern discovery. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining* (2012), ACM, pp. 823–831.