VISVESVARAYA TECHNOLOGICAL UNIVERSITY

BELAGAVI – 590018, Karnataka

INTERNSHIP REPORT ON

"PREDICTIVE SENTIMENT ANALYSIS"

Submitted in partial fulfilment for the award of degree(18CSI85)

BACHELOR OF ENGINEERING IN
ELECTRONICS AND COMMUNICATION ENGINEERING

Submitted by:
NAME: SHIVAM .R. CHAVAN
USN: 4BD20EC413

Varcons Technologies Pvt Ltd

Conducted at
Varcons Technologies Pvt Ltd

BAPUJI INSTITUTE OFENGINEERING TECHNOLOGY

Department of Electronics and communication engineering

Accredited by NBA, New Delhi

Shamanur Road, Davanagere

Karnataka 577004 2022-2023

BAPUJI INSTITUTE OFENGINEERING TECHNOLOGY

Department of Electronics and communication engineering

Accredited by NBA, New Delhi

P.O. Box No. 325, Shamanur Road Davanagere
Karnataka 577004

CERTIFICATE

This is to certify that the Internship titled **"PREDICTIVE SENTIMENT ANALYSIS"** carried out by **Mr.SHIVAM R CHAVAN ,** a bonafide student of Bapuji Institute of engineering and Technology, in partial fulfillment for the award of **Bachelor of Engineering**, in **ELECTRONICS AND COMMUNICATION ENGINEERING** under Visvesvaraya Technological University, Belagavi, during the year 2022-2023. It is certified that all corrections/suggestions indicated have been incorporated in the report.

The project report has been approved as it satisfies the academic requirements in respect of Internship prescribed for the course Internship / Professional Practice (18CSI85)

Signature of Guide

Signature of HOD **Signature of Principal**

External Viva:

Name of the Examiner Signature with Date

1) ________________ ___________________

2) ________________ ___________________

2022-2023

D E C L A R A T I O N

I.SHIVAM R CHAVAN, final year student of Eectronics and communication engineering Branch, Bapuji institute of engineering and technology davanagere Karnataka 577004, declare that the Internship has been successfully completed, in **VARCONS TECHNOLOGY Pvt. Ltd**. This report is submitted in partial fulfillment of the requirements for award of Bachelor Degree in Eectronics and communication engineering, during the academic year 2022-2023.

Date :

Place : DAVANAGERE

USN : 4BD20EC413

NAME : SHIVAM R CHAVAN

A C K N O W L E D G E M E N T

This Internship is a result of accumulated guidance, direction and support of several important persons. We take this opportunity to express our gratitude to all who have helped us to complete the Internship.

We express our sincere thanks to our Principal, for providing us adequate facilities to undertake this Internship.

We would like to thank our Head of Dept – **branch code**, for providing us an opportunity to carry out Internship and for his valuable guidance and support.

We would like to thank our (Lab assistant name) Software Services for guiding us during the period of internship.

We express our deep and profound gratitude to our guide, Guide name, Assistant/Associate Prof, for her keen interest and encouragement at every step in completing the Internship.

We would like to thank all the faculty members of our department for the support extended during the course of Internship.

We would like to thank the non-teaching members of our dept, for helping us during the Internship.

Last but not the least, we would like to thank our parents and friends without whose constant help, the completion of Internship would have not been possible.

NAME : SHIVAM R CHAVAN
USN : 4BD20EC413

1. COMPANY PROFILE

A Brief History of Varcons Technologies

Varcons Technologies, was incorporated with a goal ”To provide high quality and optimal Technological Solutions to business requirements of our clients”. Every business is a different and has a unique business model and so are the technological requirements. They understand this and hence the solutions provided to these requirements are different as well. They focus on clients requirements and provide them with tailor made technological solutions. They also understand that Reach of their Product to its targeted market or the automation of the existing process into e-client and simple process are the key features that our clients desire from Technological Solution they are looking for and these are the features that we focus on while designing the solutions for their clients.

Sarvamoola Software Services. is a Technology Organization providing solutions for all web design and development, MYSQL, PYTHON Programming, HTML, CSS, ASP.NET and LINQ. Meeting the ever increasing automation requirements, Sarvamoola Software Services. specialize in ERP, Connectivity, SEO Services, Conference Management, effective web promotion and tailor-made software products, designing solutions best suiting clients requirements.

Varcons Technologies, strive to be the front runner in creativity and innovation in software development through their well-researched expertise and establish it as an out of the box software development company in Bangalore, India. As a software development company, they translate this software development expertise into value for their customers through their professional solutions.

They understand that the best desired output can be achieved only by understanding the clients demand better. Varcons Technologies work with their clients and help them to defiine their exact solution requirement. Sometimes even they wonder that they have completely redefined their solution or new application requirement during the brainstorming session, and here they position themselves as an IT solutions consulting group comprising of high caliber consultants.

They believe that Technology when used properly can help any business to scale and achieve new heights of success. It helps Improve its efficiency, profitability, reliability; to put it in one sentence ” Technology helps you to Delight your Customers” and that is what we want to achieve.

Table of Contents

Page No.

1. Abstract

Sentiment Analysis also known as Opinion Mining refers to the use of natural language processing, text analysis to systematically identify, extract, quantify, and study affective states and subjective information. Sentiment analysis is widely applied to reviews and survey responses, online and social media, and healthcare materials for applications that range from marketing to customer service to clinical medicine.

In this project, we aim to perform Sentiment Analysis of product based reviews. Data used in this project are online product reviews collected from "amazon.com". We expect to do review-level categorization of review data with promising outcomes.

1. Introduction

Sentiment is an attitude, thought, or judgment prompted by feeling. Sentiment analysis, which is also known as opinion mining, studies people's sentiments towards certain entities. From a user's perspective, people are able to post their own content through various social media, such as forums, micro-blogs, or online social networking sites. From a researcher's perspective, many social media sites release their application programming interfaces (APIs), prompting data collection and analysis by researchers and developers. However, those types of online data have several flaws that potentially hinder the process of sentiment analysis. The first flaw is that since people can freely post their own content, the quality of their opinions cannot be guaranteed. he second flaw is that ground truth of such online data is not always available. A ground truth is more like a tag of a certain opinion, indicating whether the opinion is positive, negative, or neutral.

"It is a quite boring movie…….. but the scenes were good enough. "

The given line is a movie review that states that "it" (the movie) is quite boring but the scenes were good. Understanding such sentiments require multiple tasks.

Hence, SENTIMENTAL ANALYSIS is a kind of text classification based on *Sentimental Orientation* (SO) of opinion they contain.

Sentiment analysis of product reviews has recently become very popular in text mining and computational linguistics research.

- Firstly, evaluative terms expressing opinions must be extracted from the review.

- Secondly, the SO, or the polarity, of the opinions must be determined.

- Thirdly, the opinion strength, or the intensity, of an opinion should also be determined.

- Finally, the review is classified with respect to sentiment classes, such as Positive and Negative, based on the SO of the opinions it contains.

2. Review of Literature

The most fundamental problem in sentiment analysis is the sentiment polarity categorization, by considering a dataset containing over 5.1 million product reviews from Amazon.com with the products belonging to four categories.

A max-entropy POS tagger is used in order to classify the words of the sentence, an additional python program to speed up the process. The negation words like no, not, and more are included in the adverbs whereas Negation of Adjective and Negation of Verb are specially used to identify the phrases.

The following are the various classification models which are selected for categorization: Naïve Bayesian, Random Forest, Logistic Regression and Support Vector Machine.

For feature selection, Pang and Lee suggested to remove objective sentences by extracting subjective ones. They proposed a text-categorization technique that is able to identify subjective content using minimum cut. Gann et al. selected 6,799 tokens based on Twitter data, where each token is assigned a sentiment score, namely TSI (Total Sentiment Index), featuring itself as a positive token or a negative token. Specifically, a TSI for a certain token is computed as:

$$TSI = \frac{p - \frac{tp}{tn} \times n}{p + \frac{tp}{tn} * n}$$

where p is the number of times a token appears in positive tweets and *n* is the number of times a token appears in negative tweets is $\frac{tp}{tn}$ the ratio of total number of positive tweets over total number of negative tweets.

3. Objective of the Project

- Scrapping product reviews on various websites featuring various products specifically amazon.com.
- Analyze and categorize review data.
- Analyze sentiment on dataset from document level (review level).
- Categorization or classification of opinion sentiment into-
  Positive
   Negative

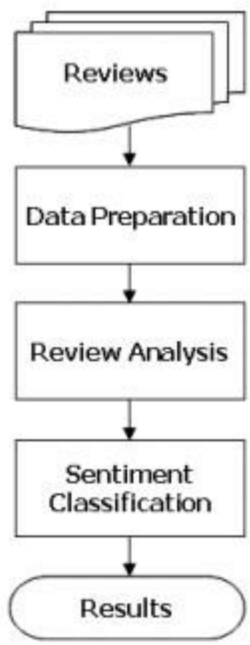


Figure 1: A typical sentiment analysis model.

4. System Design

Hardware Requirements:

- Core i5/i7 processor
- At least 8 GB RAM
- At least 60 GB of Usable Hard Disk Space

Software Requirements:

- Python 3.x
- Anaconda Distribution
- NLTK Toolkit
- UNIX/LINUX Operating System.

Data Information:

- The Amazon reviews dataset consists of reviews from amazon. The data span a period of 18 years, including ~35 million reviews up to March 2013. Reviews include product and user information, ratings, and a plaintext review. For more information, please refer to the following paper: J. McAuley and J. Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. RecSys, 2013.

- The Amazon reviews full score dataset is constructed by Xiang Zhang (xiang.zhang@nyu.edu) from the above dataset. It is used as a text classification benchmark in the following paper: Xiang Zhang, Junbo Zhao, Yann LeCun. Character-level Convolutional Networks for Text Classification. Advances in Neural Information Processing Systems 28 (NIPS 2015).

- The Amazon reviews full score dataset is constructed by randomly taking 200,000 samples for each review score from 1 to 5. In total there are 1,000,000 samples.

Star Level	General Meaning
☆	I hate it.
☆☆	I don't like it.
☆☆☆	It's okay.
☆☆☆☆	I like it.
☆☆☆☆☆	I love it.

Books	reviews (22,507,155 reviews)	metadata (2,370,585 products)	image features
Electronics	reviews (7,824,482 reviews)	metadata (498,196 products)	image features
Movies and TV	reviews (4,607,047 reviews)	metadata (208,321 products)	image features
CDs and Vinyl	reviews (3,749,004 reviews)	metadata (492,799 products)	image features
Clothing, Shoes and Jewelry	reviews (5,748,920 reviews)	metadata (1,503,384 products)	image features
Home and Kitchen	reviews (4,253,926 reviews)	metadata (436,988 products)	image features
Kindle Store	reviews (3,205,467 reviews)	metadata (434,702 products)	image features
Sports and Outdoors	reviews (3,268,695 reviews)	metadata (532,197 products)	image features
Cell Phones and Accessories	reviews (3,447,249 reviews)	metadata (346,793 products)	image features
Health and Personal Care	reviews (2,982,326 reviews)	metadata (263,032 products)	image features
Toys and Games	reviews (2,252,771 reviews)	metadata (336,072 products)	image features
Video Games	reviews (1,324,753 reviews)	metadata (50,953 products)	image features
Tools and Home Improvement	reviews (1,926,047 reviews)	metadata (269,120 products)	image features
Beauty	reviews (2,023,070 reviews)	metadata (259,204 products)	image features
Apps for Android	reviews (2,638,173 reviews)	metadata (61,551 products)	image features
Office Products	reviews (1,243,186 reviews)	metadata (134,838 products)	image features
Pet Supplies	reviews (1,235,316 reviews)	metadata (110,707 products)	image features
Automotive	reviews (1,373,768 reviews)	metadata (331,090 products)	image features
Grocery and Gourmet Food	reviews (1,297,156 reviews)	metadata (171,760 products)	image features
Patio, Lawn and Garden	reviews (993,490 reviews)	metadata (109,094 products)	image features
Baby	reviews (915,446 reviews)	metadata (71,317 products)	image features
Digital Music	reviews (836,006 reviews)	metadata (279,899 products)	image features
Musical Instruments	reviews (500,176 reviews)	metadata (84,901 products)	image features
Amazon Instant Video	reviews (583,933 reviews)	metadata (30,648 products)	image features

Data Format:

The dataset we will use is .json file. The sample of the dataset is given below.

{

"reviewSummary": "Surprisingly delightful",

"reviewText": " This is a first read filled with unexpected humor and profound insights into the art of politics and policy. In brief, it is sly, wry, and wise. ",

"reviewRating": "4",

}

5. Methodology for Implementation (Formulation/Algorithm)

DATA COLLECTION:

Data which means product reviews collected from amazon.com from May 1996 to July 2014. Each review includes the following information: 1) reviewer ID; 2) product ID; 3) rating; 4) time of the review; 5) helpfulness; 6) review text. Every rating is based on a 5-star scale, resulting all the ratings to be ranged from 1-star to 5-star with no existence of a half-star or a quarter-star.

SENTIMENT SENTENCE EXTRACTION & POS TAGGING:

Tokenization of reviews after removal of STOP words which mean nothing related to sentiment is the basic requirement for POS tagging. After proper removal of STOP words like “am, is, are, the, but” and so on the remaining sentences are converted in tokens. These tokens take part in POS tagging

In natural language processing, part-of-speech (POS) taggers have been developed to classify words based on their parts of speech. For sentiment analysis, a POS tagger is very useful because of the following two reasons: 1) Words like nouns and pronouns usually do not contain any sentiment. It is able to filter out such words with the help of a POS tagger; 2) A POS tagger can also be used to distinguish words that can be used in different parts of speech.

NEGETIVE PHRASE IDENTIFICATION:

Words such as adjectives and verbs are able to convey opposite sentiment with the help of negative prefixes. For instance, consider the following sentence that was found in an electronic device’s review: “The built in speaker also has its uses but so far nothing revolutionary." The word, “revolutionary" is a positive word according to the list in. However, the phrase “nothing revolutionary" gives more or less negative feelings. Therefore, it is crucial to identify such phrases. In this work, there are two types of phrases have been identified, namely negation-of-adjective (NOA) and negation-of-verb (NOV).

SENTIMENT CLASSIFICATION ALGORITHMS:

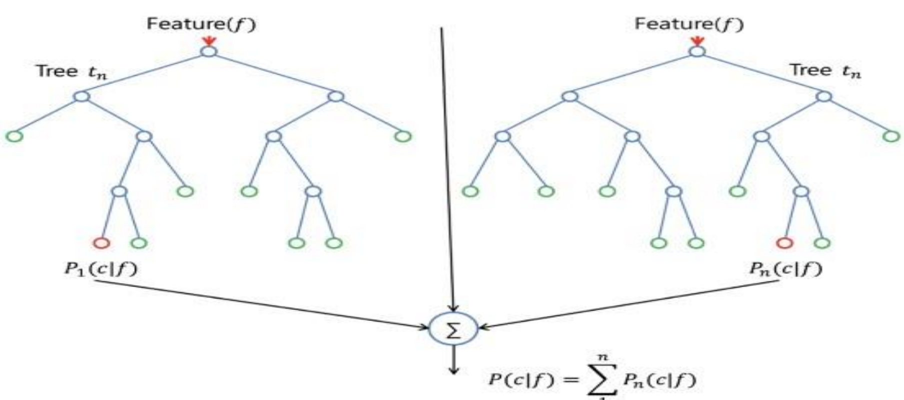
Naïve Bayesian classifier:

The Naïve Bayesian classifier works as follows: Suppose that there exist a set of training data, *D*, in which each tuple is represented by an *n*-dimensional feature vector, $X=x_1,x_2,..,x_n$, indicating *n* measurements made on the tuple from *n* attributes or features. Assume that there are *m* classes, $C_1,C_2,...,C_m$. Given a tuple *X*, the classifier will predict that *X* belongs to C_i if and only if: $P(C_i|X)>P(C_j|X)$, where $i,j\in[1,m]$ *a n d* $i\neq j$. $P(C_i|X)$ is computed as:

$$P(C_i|X) = \prod_{k=1}^{n} P(x_k|C_i)$$

Random forest

The random forest classifier was chosen due to its superior performance over a single decision tree with respect to accuracy. It is essentially an ensemble method based on bagging. The classifier works as follows: Given *D*, the classifier firstly creates *k* bootstrap samples of *D*, with each of the samples denoting as D_i. A D_i has the same number of tuples as *D* that are sampled with replacement from *D*. By sampling with replacement, it means that some of the original tuples of *D* may not be included in D_i, whereas others may occur more than once. The classifier then constructs a decision tree based on each D_i. As a result,



a “forest" that consists of *k* decision trees is formed.

To classify an unknown tuple, *X*, each tree returns its class prediction counting as one vote. The final decision of *X*'s class is assigned to the one that has the most votes.

The decision tree algorithm implemented in scikit-learn is CART (Classification and Regression Trees). CART uses Gini index for its tree induction. For *D*, the Gini index is computed as:

$$Gini(D) = 1 - \sum_{i=1}^{m} p_i^2$$

Where p_i is the probability that a tuple in *D* belongs to class C_i. The Gini index measures the impurity of *D*. The lower the index value is, the better *D* was partitioned.

Support vector machine

Support vector machine (SVM) is a method for the classification of both linear and nonlinear data. If the data is linearly separable, the SVM searches for the linear optimal separating hyperplane (the linear kernel), which is a decision boundary that separates data of one class from another. Mathematically, a separating hyper plane can be written as: $W \cdot X + b = 0$, where *W* is a weight vector and $W = w1, w2, ..., w\ n$. *X* is a training tuple. *b* is a scalar. In order to optimize the hyperplane, the problem essentially transforms to the minimization of $\|W\|$, which is eventually computed as:

$$\sum_{i=1}^{n} \alpha_i y_i x_i ,$$

where α_i are numeric parameters, and y_i are labels based on support vectors, X_i .

That is: if $y_i = 1$ then

$$\sum_{i=1}^{n} w_i x_i \geq 1;$$

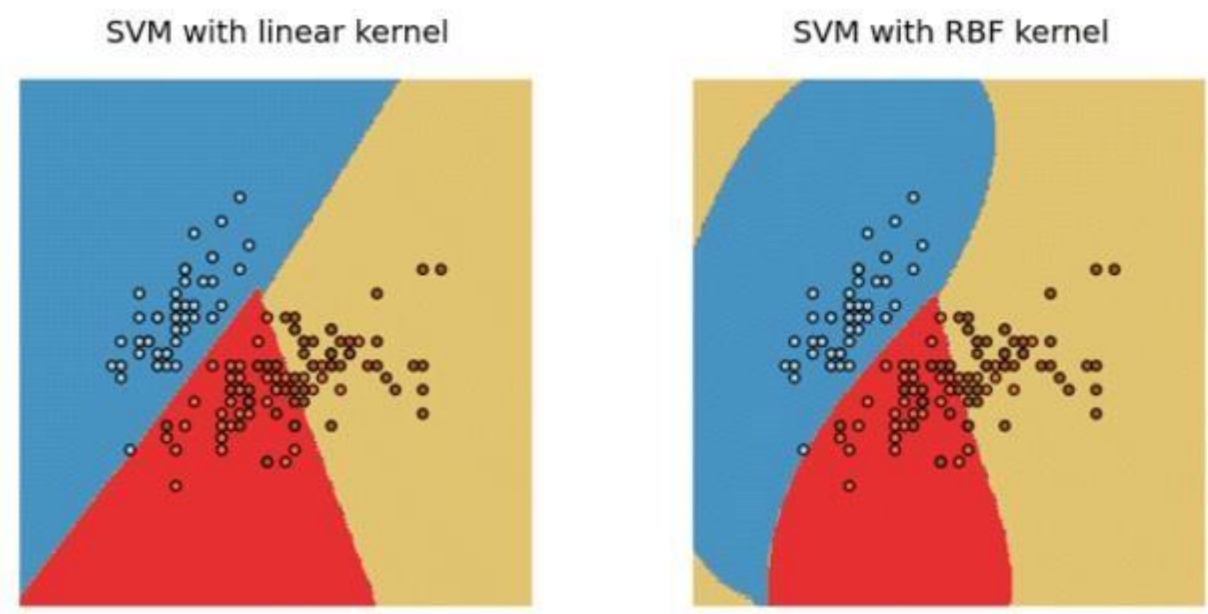
if $y_i = -1$ then

$$\sum_{i=1}^{n} w_i x_i \geq -1.$$

If the data is linearly inseparable, the SVM uses nonlinear mapping to transform the data into a higher dimension. It then solve the problem by finding a linear hyperplane. Functions to perform such transformations are called kernel functions. The kernel function selected for our experiment is the Gaussian Radial Basis Function (RBF):

$$K(X_i, X_j) = e^{-\gamma\|X_i - X_j\|^2/2}$$

where X_i are support vectors, X_j are testing tuples, and γ is a free parameter that uses the default value from scikit-learn in our experiment. Figure shows a classification example of SVM based on the linear kernel and the RBF kernel on the next page-



Logistic Regression

Logistic regression predicts the probability of an outcome that can only have two values (i.e. a dichotomy). The prediction is based on the use of one or several predictors (numerical and categorical). A linear regression is not appropriate for predicting the value of a binary variable for two reasons:

- A linear regression will predict values outside the acceptable range (e.g. predicting probabilities outside the range 0 to 1)
- Since the dichotomous experiments can only have one of two possible values for each experiment, the residuals will not be normally distributed about the predicted line.

On the other hand, a logistic regression produces a logistic curve, which is limited to values between 0 and 1. Logistic regression is similar to a linear regression, but the curve is constructed using the natural logarithm of the "odds" of the target variable, rather than the probability. Moreover, the predictors do not have to be normally distributed or have equal variance in each group.

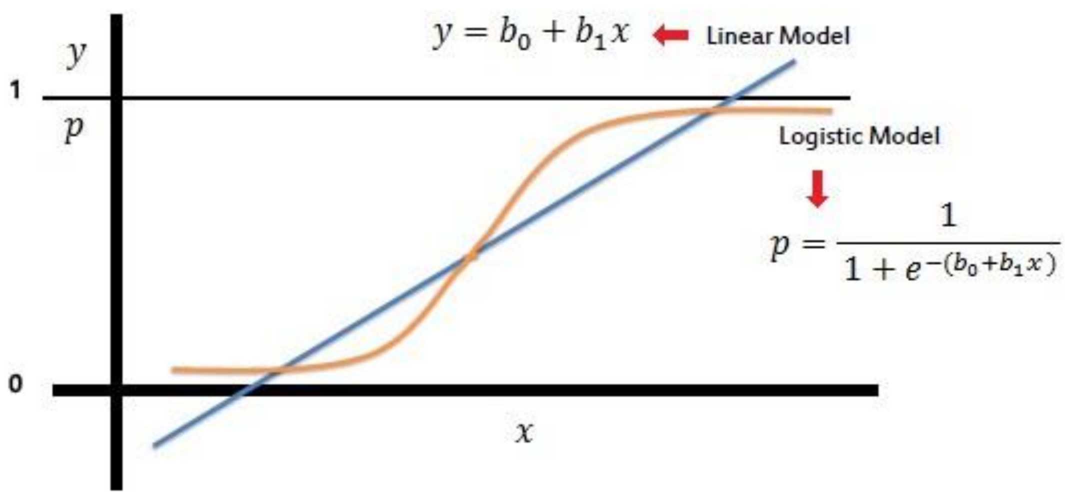
Logistic regression uses maximum likelihood estimation (MLE) to obtain the model coefficients that relate predictors to the target. After this initial function is estimated, the process is repeated until LL (Log Likelihood) does not change significantly.

$$\boldsymbol{\beta}^{1} = \boldsymbol{\beta}^{0} + [\boldsymbol{X}^{T}\boldsymbol{W}\boldsymbol{X}]^{-1}.\boldsymbol{X}^{T}(\boldsymbol{y} - \boldsymbol{\mu})$$

$\boldsymbol{\beta}$ is a vector of the logistic regression coefficients.

$\boldsymbol{W}$ is a square matrix of order N with elements $n_i\pi_i(1-\pi_i)$ on the diagonal and zeros everywhere else.

$\boldsymbol{\mu}$ is a vector of length N with elements $\mu_i = n_i\pi_i$.



6. Implementation Details

The training of dataset consists of the following steps:

- **Unpacking of data:** The huge dataset of reviews obtained from amazon.com comes in a .json file format. A small python code has been implemented in order to read the dataset from those files and dump them in to a pickle file for easier and fastaccess and object serialization.

```python
with open(data_file, 'r') as file_handler:
    for review in file_handler.readlines():
        df[i] = ast.literal_eval(review)
        i += 1

reviews_df = pd.DataFrame.from_dict(df, orient = 'index')
reviews_df.to_pickle('reviews_digital_music.pickle')
```

Hence initial fetching of data is done in this section using Python File Handlers.

- **Preparing Data for Sentiment Analysis:**

i) The pickle file is hence loaded in this step and the data besides the one used for sentiment analysis is removed. As shown in our sample dataset in Page 11, there are a lot of columns in the data out of which only rating and text review is what we require. So, the column, "reviewSummary" is dropped from the data file.

ii) After that, the review ratings which are 3 out of 5 are removed as they signify neutral review, and all we are concerned of is positive and negative reviews.

iii) The entire task of preprocessing the review data is handled by this

```python
reviews_df.drop(columns = ['reviewSummary'], inplace = True)
reviews_df['reviewRating'] = reviews_df.reviewRating.astype('int')
```

```python
reviews_df = reviews_df[reviews_df.reviewRating != 3] # Ignoring 3-star reviews -> neutral
reviews_df = reviews_df.assign(sentiment = np.where(reviews_df['reviewRating'] >= 4, 1, 0)) # 1 -> Positive, 0 -> Negati
```

utility class- "NltkPreprocessor".

```python

class NltkPreprocessor:

    def __init__(self, stopwords = None, punct = None, lower = True, strip = True):
        self.lower = lower
        self.strip = strip
        self.stopwords = stopwords or set(sw.words('english'))
        self.punct = punct or set(string.punctuation)
        self.lemmatizer = WordNetLemmatizer()

    def tokenize(self, document):
        tokenized_doc = []

        for sent in sent_tokenize(document):
            for token, tag in pos_tag(wordpunct_tokenize(sent)):
                token = token.lower() if self.lower else token
                token = token.strip() if self.strip else token
                token = token.strip('_0123456789') if self.strip else token
                # token = re.sub(r'\d+', '', token)

                if token in self.stopwords:
                    continue

                if all(char in self.punct for char in token):
                    continue

                lemma = self.lemmatize(token, tag)
                tokenized_doc.append(lemma)

        return tokenized_doc

    def lemmatize(self, token, tag):
        tag = {
            'N': wn.NOUN,
            'V': wn.VERB,
            'R': wn.ADV,
            'J': wn.ADJ
        }.get(tag[0], wn.NOUN)

        return self.lemmatizer.lemmatize(token, tag)

```

iv) The time required to prepare the following data is hence displayed.

```
administrator@administrator-OptiPlex-3040:~/Desktop/sentiment_analysis$
Preprocessing data...
Preprocessing data completed!
Preprocessing time:  0.163 s
```

The time taken to preprocess the data is calculated and displayed

- **Preprocessing Data:**This is a vital part of training the dataset. Here Words present in the file are accessed both as a solo word and also as pair of words. Because, for example the word “bad” means negative but when someone writes “not bad” it refers to as positive. In such cases considering single word for training data will work otherwise. So words in pairs are checked to find the occurrence to modifiers before any adjective which if present which might provide a different meaning to the outlook.

```
X = reviews_df_preprocessed.iloc[:, -1].values
y = reviews_df_preprocessed.iloc[:, -2].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)
```

- **Training Data/ Evaluation:** The main chunk of code that does the whole evaluation of sentimental analysis based on the preprocessed data is a part of this. The following are the steps followed:

```
pipeline = Pipeline([
                ('vect', TfidfVectorizer(ngram_range = (1,2), stop_words = 'english', sublinear_tf = True)),
                ('chi', SelectKBest(score_func = chi2, k = 50000)),
                ('clf', LinearSVC(C = 1.0, penalty = 'l1', max_iter = 3000, dual = False, class_weight='balanced'))
            ])

model = pipeline.fit(X_train, y_train)
```

i) The Accuracy, Precision, Recall, and Evaluation time is calculated and displayed.

ii) Navie Bayes, Logistic Regression, Linear SVM and Random forest classifiers are applied on the dataset for evaluation of sentiments.

iii) Prediction of test data is done and Confusion Matrix of prediction isdisplayed. **iv)** Total positive and negative reviews are counted.

v) A review like sentence is taken as input on the console and if positive the console gives 1 as output and 0 for negative input.

7. Results and Sample Output

The ultimate outcome of this Training of Public reviews dataset is that, the machine is capable of judging whether an entered sentence bears positive response or negative response.

Precision (also called positive predictive value) is the fraction of relevant instances among the retrieved instances, while **Recall** (also known as sensitivity) is the fraction of relevant instances that have been retrieved over the total amount of relevant instances. Both precision and recall are therefore based on an understanding and measure of relevance.

$$\text{precision} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{retrieved documents}\}|}$$

$$\text{recall} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{relevant documents}\}|}$$

F_1 score (also **F-score** or **F-measure**) is a measure of a test's accuracy. It considers both the precision p and the recall r of the test to compute the score: p is the number of correct positive results divided by the number of all positive results returned by the classifier, and r is the number of correct positive results divided by the number of all relevant samples (all samples that should have been identified as positive). The F_1 score is the harmonic average of the precision and recall, where an F_1 score reaches its best value at 1 (perfect precision and recall) and worst at 0.

$$F_1 = \frac{2}{\frac{1}{\text{recall}} + \frac{1}{\text{precision}}} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}.$$

In statistics, a **receiver operating characteristic curve**, i.e. **ROC curve**, is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied. The Total Operating Characteristic (TOC) expands on the idea of ROC by showing the total information in the two-by-two contingency table for each threshold. ROC gives only two bits of relative information for each threshold, thus the TOC gives strictly more information than the ROC.

When using normalized units, the area under the curve (often referred to as simply the AUC) is equal to the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one (assuming 'positive' ranks higher than 'negative'). This can be seen as follows: the area under the curve is given by (the integral boundaries are reversed as large T has a lower value on the x-axis).

$$A = \int_{\infty}^{-\infty} \mathrm{TPR}(T)\mathrm{FPR}'(T)\,dT = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} I(T' > T) f_1(T') f_0(T)\,dT'\,dT = P(X_1 > X_0)$$

The machine evaluates the accuracy of training the data along with precision Recall and F_1

The Confusion matrix of evaluation is calculated.

It is thus capable of judging an externally written review as positive or negative.

A positive review will be marked as [1], and a negative review will be hence marked as [0].

Results obtained using Hold-out Strategy(Train-Test split) [values rounded upto 2 decimal places]**.**

Name of classifier	F_1	Accuracy	Precision	Recall	ROC AUC
Multinomial NB	**85.25%**	**85.31%**	**85.56%**	**84.95%**	**85.31%**
Logistic Regression	**88.12%**	**88.05%**	**87.54%**	**88.72%**	**88.05%**
Linear SVC	**88.12%**	**88.11%**	**87.59%**	**88.80%**	**88.11%**
Random Forest	**82.43%**	**81.82%**	**79.74%**	**85.30%**	**81.83%**

The Confusion Matrix Format is as follows:

True Negative	False Positive
False Negative	True Positive

The Confusion Matrix of Each Classifier are as follows:

68556	11470
12032	67942

Classifier 1: Multinomial NB

69928	10098
9023	70951

Classifier 3: Liner SVC

Classifier 2: Logistic Regression

69963	10063
8955	17019

62695	17331
11749	68225

Classifier 4: Random Forest

The following are the images of such sample output after successful dataset training using the classifiers:

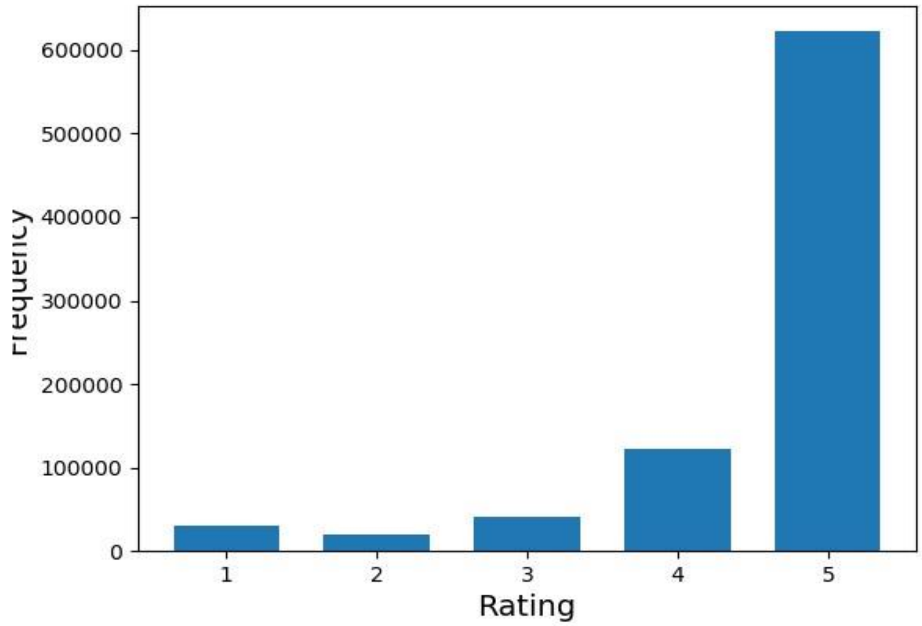
```
~/Projects/machine-learning/sentiment-analysis — -bash    ~/Projects/machine-learning/sentiment-analysis — -bash    ~/Projects/machine-learning/sentiment-analysis — -bash
Pranits-MacBook-Air:sentiment-analysis pranit$ python3 sentiment_analyzer.py

Holdout Strategy...

Splitting data using Train-Test split...
Splitting data completed!
Splitting time:  0.201 s

Training data... Classifier MNB
Training data completed!
Training time:  183.1 s

Training data... Classifier LR
Training data completed!
Training time:  217.264 s

Training data... Classifier SVM
Training data completed!
Training time:  204.015 s

Training data... Classifier RF
Training data completed!
Training time:  719.168 s

Predicting Test data... Classifier MNB
Prediction completed!
Prediction time:  28.198 s

Predicting Test data... Classifier LR
Prediction completed!
Prediction time:  27.013 s

Predicting Test data... Classifier SVM
Prediction completed!
Prediction time:  27.175 s

Predicting Test data... Classifier RF
Prediction completed!
Prediction time:  39.286 s

Evaluating results... Classifier MNB
Results evaluated!
Evaluation time:  0.34 s

Evaluating results... Classifier LR
Results evaluated!
Evaluation time:  0.325 s

Evaluating results... Classifier SVM
Results evaluated!
Evaluation time:  0.318 s

Evaluating results... Classifier RF
Results evaluated!
```

```
~/Projects/machine-learning/sentiment-analysis — -bash    ~/Projects/machine-learning/sentiment-analysis — -bash    ~/Projects/machine-learning/sentiment-analysis — -bash
Evaluating results... Classifier MNB
Results evaluated!
Evaluation time:  0.34 s

Evaluating results... Classifier LR
Results evaluated!
Evaluation time:  0.325 s

Evaluating results... Classifier SVM
Results evaluated!
Evaluation time:  0.318 s

Evaluating results... Classifier RF
Results evaluated!
Evaluation time:  0.315 s

Evaluation metrics of classifier MNB
Accuracy: 0.8531125
Precision: 0.8555633909232861
Recall: 0.8495511041088354
f1: 0.852546647760782
ROC AUC: 0.8531113429223856
Confusion Matrix: [[68556 11470]
 [12032 67942]]
Evaluation metrics of classifier LR
Accuracy: 0.88049375
Precision: 0.8754087033769695
Recall: 0.8871758321454473
f1: 0.8812529887034772
ROC AUC: 0.8804959209711317
Confusion Matrix: [[69928 10098]
 [ 9023 70951]]
Evaluation metrics of classifier SVM
Accuracy: 0.8811375
Precision: 0.8758910732345033
Recall: 0.8880261084852578
f1: 0.8819168487979336
ROC AUC: 0.8811397380703848
Confusion Matrix: [[69963 10063]
 [ 8955 71019]]
Evaluation metrics of classifier RF
Accuracy: 0.81825
Precision: 0.7974309224367666
Recall: 0.8530897541701052
f1: 0.8243218751887875
ROC AUC: 0.8182613192413517
Confusion Matrix: [[62695 17331]
 [11749 68225]]

Total number of observations: 160000
Positives in observation: 79974
Negatives in observation: 80026
Majority class is: 50.01624999999999%
Pranits-MacBook-Air:sentiment-analysis pranit$
```

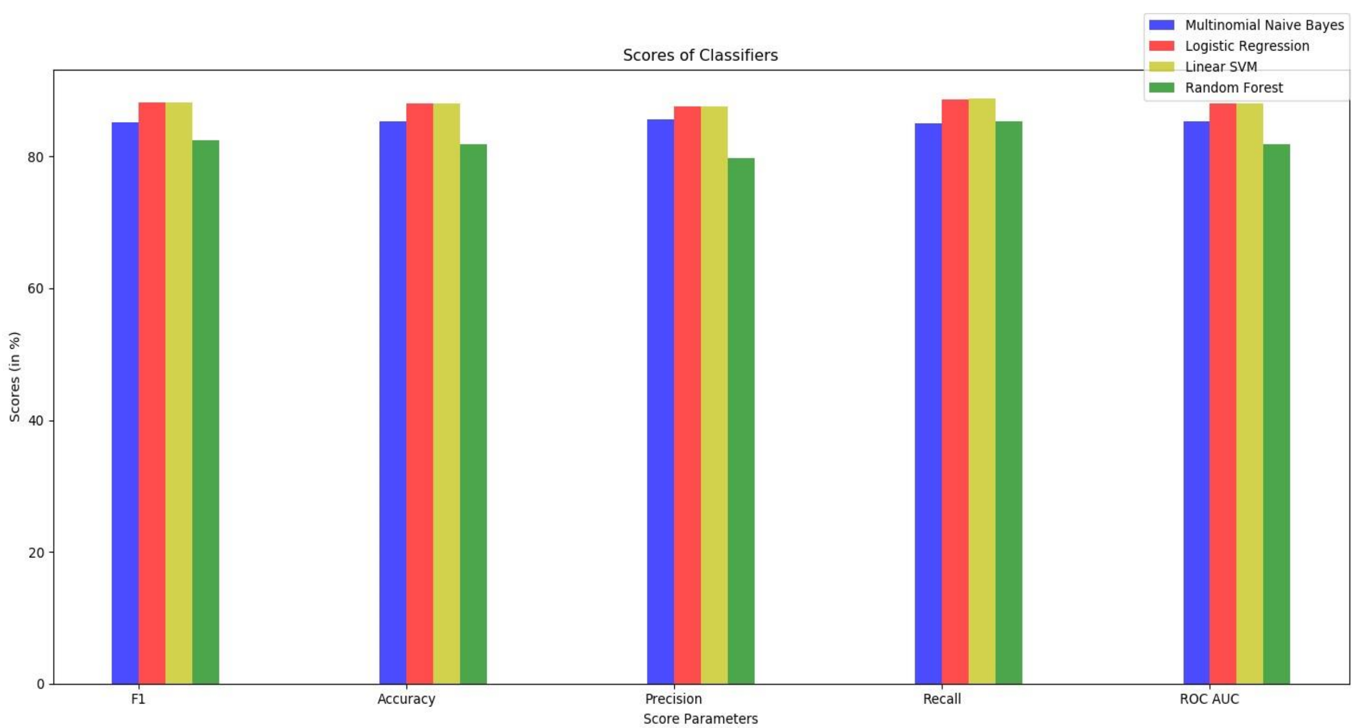
```
administrator@administrator-OptiPlex-3040:~/Desktop/sentiment_analysis$ python3 sentiment_analyzer.py
Preprocessing data...
Preprocessing data completed!
Preprocessing time:  0.131 s

Training data...
Training data completed!
Training time:  244.431 s

Predicting Test data...
Prediction completed!
Prediction time:  11.46 s

Evaluating results...
Accuracy: 0.94855693908754
Precision: 0.983433383243815
Recall: 0.9613014112497147
f1: 0.9722414612616284
Results evaluated!
Evaluation time:  0.084 s

Confusion matrix: [[  7575   2412]
 [  5764 143182]]

Total number of observations: 158933
Positives in observation: 148946
Negatives in observation: 9987
Majority class is: 93.7162200424078%
Worst product ever
[0]
```

```
administrator@administrator-OptiPlex-3040:~/Desktop/sentiment_analysis$ python3 sentiment_analyzer.py
Preprocessing data...
Preprocessing data completed!
Preprocessing time:  0.163 s

Training data...
Training data completed!
Training time:  239.406 s

Predicting Test data...
Prediction completed!
Prediction time:  11.402 s

Evaluating results...
Accuracy: 0.9486261506420944
Precision: 0.983467838868093
Recall: 0.9613416943053187
f1: 0.9722789017488227
Results evaluated!
Evaluation time:  0.086 s

Confusion matrix: [[  7580   2407]
 [  5758 143188]]

Total number of observations: 158933
Positives in observation: 148946
Negatives in observation: 9987
Majority class is: 93.7162200424078%
not a good product
[1]
```

The Bar Graph showing the Frequency of Ratings in the dataset



This Bar graph shows the score of each classifier after successful training. The parameters be: F_1 Score, Accuracy, Precision, Recall and Roc-Auc.



8. Conclusion

Sentiment analysis deals with the classification of texts based on the sentiments they contain. This article focuses on a typical sentiment analysis model consisting of three core steps, namely data preparation, review analysis and sentiment classification, and describes representative techniques involved in those steps.

Sentiment analysis is an emerging research area in text mining and computational linguistics, and has attracted considerable research attention in the past few years. Future research shall explore sophisticated methods for opinion and product feature extraction, as well as new classification models that can address the ordered labels property in rating inference. Applications that utilize results from sentiment analysis is also expected to emerge in the near future.

Appendix

Code:

Loading the dataset:

```
import json import pickle import
numpy as np from matplotlib
import pyplot as plt from textblob
import TextBlob


# fileHandler = open('datasets/reviews_digital_music.json', 'r')
# reviewDatas = fileHandler.read().split('\n')
# reviewText = []
# reviewRating = []


# for review in reviewDatas:
#       if review == "":
#               continue
#       r = json.loads(review)
#       reviewText.append(r['reviewText'])
#       reviewRating.append(r['overall'])

# fileHandler.close()
# saveReviewText = open('review_text.pkl', 'wb')
# saveReviewRating = open('review_rating.pkl','wb')
# pickle.dump(reviewText, saveReviewText) #
pickle.dump(reviewRating, saveReviewRating)
reviewTextFile = open('review_text.pkl', 'rb')
reviewRatingFile = open('review_rating.pkl', 'rb')
reviewText = pickle.load(reviewTextFile)
reviewRating = pickle.load(reviewRatingFile)
# print(len(reviewText))
```

```
# print(reviewText[0])
# print(reviewRating[0]) # ratings
= np.array(reviewRating)
 plt.hist(ratings, bins=np.arange(ratings.min(), ratings.max()+2)-0.5, rwidth=0.7)
plt.xlabel('Rating', fontsize=14)  plt.ylabel('Frequency', fontsize=14)
plt.title('Histogram of Ratings', fontsize=18) plt.show() lang = {} i = 0 for
review in reviewText:
        tb = TextBlob(review)
l = tb.detect_language()
if l != 'en':
                lang.setdefault(l, [])
        lang[l].append(i)
print(i, l)         i += 1 print(lang)
```

Scrapping data:

```
from selenium import webdriver from
selenium.webdriver.chrome.options import Options from
bs4 import BeautifulSoup import openpyxl class
Review():       def __init__(self):
                self.rating=""
                self.info=""
                self.review=""
def scrape():
        options = Options()     options.add_argument("--headless") # Runs Chrome in
headless mode.          options.add_argument('--no-sandbox') # # Bypass OS security
model   options.add_argument('start-maximized')         options.add_argument('disable-
infobars')      options.add_argument("--disable-extensions")
driver=webdriver.Chrome(executable_path=r'C:\chromedriver\chromedriver.exe')
        url='https://www.amazon.com/Moto-PLUS-5th-Generation-Exclusive/product-
```

```
reviews/B0785NN142/ref=cm_cr_arp_d_paging_btm_2?ie=UTF8&reviewerType=all_reviews&pageNumb
er=5'
        driver.get(url)


        soup=BeautifulSoup(driver.page_source,'lxml')
ul=soup.find_all('div',class_='a-section review')
review_list=[]  for d in ul:
                a=d.find('div',class_='a-row')
sib=a.findNextSibling()
                b=d.find('div',class_='a-row a-spacing-medium review-data')
                '''print sib.text'''
                new_r=Review()
new_r.rating=a.text             new_r.info=sib.text
        new_r.review=b.text


                review_list.append(new_r)
driver.quit()   return review_list def
main():
        m = scrape()
        i=1 for r in
        m:


                book = openpyxl.load_workbook('Sample.xlsx')                    sheet =
book.get_sheet_by_name('Sample Sheet')                  sheet.cell(row=i, column=1).value = r.rating
        sheet.cell(row=i, column=1).alignment = openpyxl.styles.Alignment(horizontal='center',
vertical='center', wrap_text=True)
                sheet.cell(row=i, column=3).value = r.info
                sheet.cell(row=i, column=3).alignment =
openpyxl.styles.Alignment(horizontal='center', vertical='center', wrap_text=True)
sheet.cell(row=i, column=5).value = r.review.encode('utf-8')                sheet.cell(row=i,
column=5).alignment = openpyxl.styles.Alignment(horizontal='center', vertical='center',
wrap_text=True)
```

```
book.save('Sample.xlsx')
i=i+1 if
__name__ == '__main__':
 main()
```

Preprocessing Data:

```
import string from nltk.corpus import stopwords as sw from nltk.corpus import wordnet
as wn from nltk import wordpunct_tokenize from nltk import sent_tokenize from nltk
import WordNetLemmatizer from nltk import pos_tag class NltkPreprocessor: def
__init__(self, stopwords = None, punct = None, lower = True, strip = True):
self.lower = lower self.strip = strip
        self.stopwords = stopwords or set(sw.words('english'))
        self.punct = punct or set(string.punctuation)
        self.lemmatizer = WordNetLemmatizer()

    def tokenize(self, document):
        tokenized_doc = []

        for sent in sent_tokenize(document): for token, tag in
pos_tag(wordpunct_tokenize(sent)): token = token.lower() if
self.lower else token token = token.strip() if self.strip else
token token = token.strip('_0123456789') if self.strip else token
            # token = re.sub(r'\d+', '', token)
            if token in self.stopwords:
                continue
            if all(char in self.punct for char in token):
                continue

            lemma = self.lemmatize(token, tag)
tokenized_doc.append(lemma)
```

```
            return tokenized_doc
        def lemmatize(self, token,
tag):
            tag = {
                'N': wn.NOUN,
                'V': wn.VERB,
                'R': wn.ADV,
                'J': wn.ADJ
            }.get(tag[0], wn.NOUN)
            return self.lemmatizer.lemmatize(token, tag)
```

Sentiment Analysis:

```
 import ast import numpy as np import pandas as pd
import re from nltk.corpus import stopwords from
nltk.stem import SnowballStemmer from
sklearn.model_selection import train_test_split
from sklearn.feature_selection import SelectKBest, chi2, SelectPercentile, f_classif from
sklearn.feature_extraction.text import TfidfVectorizer from sklearn.pipeline import Pipeline from
sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score,
confusion_matrix from sklearn.svm import LinearSVC # from textblob import TextBlob from time
import time

def getInitialData(data_file):
        print('Fetching initial data...')
        t = time()

        i = 0     df = {}          with
open(data_file, 'r') as file_handler:
            for review in file_handler.readlines():
        df[i] = ast.literal_eval(review)
                i += 1
```

```
reviews_df = pd.DataFrame.from_dict(df, orient = 'index')
reviews_df.to_pickle('reviews_digital_music.pickle')  print('Fetching data completed!')  print('Fetching time: ', round(time()-t, 3), 's\n')



# def filterLanguage(text):
#       text_blob = TextBlob(text)
#       return text_blob.detect_language()

def   prepareData(reviews_df):
print('Preparing   data...')    t   =
time()

      reviews_df.rename(columns = {"overall" : "reviewRating"}, inplace=True)
reviews_df.drop(columns = ['reviewerID', 'asin', 'reviewerName', 'helpful', 'summary', 'unixReviewTime', 'reviewTime'], inplace = True)



      reviews_df = reviews_df[reviews_df.reviewRating != 3.0] # Ignoring 3-star reviews -> neutral
reviews_df = reviews_df.assign(sentiment = np.where(reviews_df['reviewRating'] >= 4.0, 1, 0)) # 1 -> Positive, 0 -> Negative

      stemmer = SnowballStemmer('english')
stop_words = stopwords.words('english')

      # print(len(reviews_df.reviewText))
      # filterLanguage = lambda text: TextBlob(text).detect_language()
      # reviews_df = reviews_df[reviews_df['reviewText'].apply(filterLanguage) == 'en']
# print(len(reviews_df.reviewText))

      reviews_df = reviews_df.assign(cleaned = reviews_df['reviewText'].apply(lambda text: '
```

```
'.join([stemmer.stem(w) for w in re.sub('[^a-z]+|(quot)+', ' ', text.lower()).split() if w not in stop_words])))
reviews_df.to_pickle('reviews_digital_music_preprocessed.pickle')

        print('Preparing data completed!')
print('Preparing time: ', round(time()-t, 3), 's\n')

def preprocessData(reviews_df_preprocessed):
 print('Preprocessing data...')  t =
time()

        X = reviews_df_preprocessed.iloc[:, -1].values
y = reviews_df_preprocessed.iloc[:, -2].values

        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)

        print('Preprocessing data completed!')
print('Preprocessing time: ', round(time()-t, 3), 's\n')

        return X_train, X_test, y_train, y_test

def evaluate(y_test, prediction):
print('Evaluating results...')
        t = time()

        print('Accuracy: {}'.format(accuracy_score(y_test, prediction)))
print('Precision: {}'.format(precision_score(y_test, prediction)))
print('Recall: {}'.format(recall_score(y_test, prediction)))        print('f1:
{}'.format(f1_score(y_test, prediction)))

        print('Results evaluated!')
print('Evaluation time: ', round(time()-t, 3), 's\n')
```

```
# getInitialData('datasets/reviews_digital_music.json')
# reviews_df = pd.read_pickle('reviews_digital_music.pickle')

# prepareData(reviews_df) reviews_df_preprocessed =
pd.read_pickle('reviews_digital_music_preprocessed.pickle')
# print(reviews_df_preprocessed.isnull().values.sum()) # Check for any null values

X_train, X_test, y_train, y_test = preprocessData(reviews_df_preprocessed)

print('Training data...') t
= time()

pipeline = Pipeline([
                        ('vect', TfidfVectorizer(ngram_range = (1,2), stop_words = 'english',
sublinear_tf = True)),
                        ('chi', SelectKBest(score_func = chi2, k = 50000)),
                        ('clf', LinearSVC(C = 1.0, penalty = 'l1', max_iter = 3000, dual = False,
class_weight = 'balanced'))
                ])

model = pipeline.fit(X_train, y_train)

print('Training data completed!') print('Training
time: ', round(time()-t, 3), 's\n')

print('Predicting Test data...') t
= time()

prediction = model.predict(X_test)
```

```
print('Prediction completed!')
print('Prediction time: ', round(time()-t, 3), 's\n')

evaluate(y_test, prediction)

print('Confusion matrix: {}'.format(confusion_matrix(y_test, prediction)))
print() l = (y_test == 0).sum() + (y_test ==
1).sum() s = y_test.sum()
print('Total number of observations: ' + str(l))
print('Positives in observation: ' + str(s)) print('Negatives
in observation: ' + str(l - s))
print('Majority class is: ' + str(s / l * 100) + '%')
```

Graph Plotting Code: import numpy as np import matplotlib.pyplot as plt from matplotlib.ticker import MaxNLocator from collections import namedtuple n_groups = 5

```
score_MNB = (85.25,    85.31, 85.56, 84.95, 85.31)
score_LR = (88.12,     88.05, 87.54, 88.72, 88.05)
score_LSVC=(88.12,     88.11, 87.59, 88.80, 88.11)
score_RF=(82.43,       81.82, 79.74, 85.30, 81.83)

#n1=(score_MNB[0], score_LR[0], score_LSVC[0], score_RF[0])
#n2=(score_MNB[1], score_LR[1], score_LSVC[1], score_RF[1])
#n3=(score_MNB[2], score_LR[2], score_LSVC[2], score_RF[2])
#n4=(score_MNB[3], score_LR[3], score_LSVC[3], score_RF[3])
#n5=(score_MNB[4], score_LR[4], score_LSVC[4], score_RF[4])
fig, ax = plt.subplots() index = np.arange(n_groups) bar_width =
0.1 opacity = 0.7 error_config = {'ecolor': '0.3'} rects1 =
ax.bar(index,score_MNB, bar_width,          alpha=opacity,
color='b',
```

```
        error_kw=error_config,
label='Multinomial Naive Bayes') z=index
+ bar_width rects2 = ax.bar(z, score_LR,
bar_width,          alpha=opacity,
color='r',          error_kw=error_config,
label='Logistic Regression') z=z+
bar_width
rects3 = ax.bar(z, score_LSVC, bar_width,
alpha=opacity, color='y',
error_kw=error_config,
label='Linear SVM') z=z+ bar_width
rects4 = ax.bar(z, score_RF, bar_width,
alpha=opacity, color='g',
error_kw=error_config,
label='Random Forest') ax.set_xlabel('Score
Parameters') ax.set_ylabel('Scores (in %)')
ax.set_title('Scores of Classifiers')
ax.set_xticks(index + bar_width / 2)
ax.set_xticklabels(('F1', 'Accuracy', 'Precision', 'Recall', 'ROC AUC'))
ax.legend(bbox_to_anchor=(1, 1.02), loc=5, borderaxespad=0)
fig.tight_layout() plt.show()
```

References

- S. ChandraKala1 and C. Sindhu2, “OPINION MINING AND SENTIMENT CLASSIFICATION: A SURVEY,”.Vol .3(1),Oct 2012,420-427
- G.Angulakshmi , Dr.R.ManickaChezian ,”An Analysis on Opinion Mining: Techniques and Tools”. Vol 3(7), 2014 www.iarcce.com.
- Callen Rain,”Sentiment Analysis in Amazon Reviews Using Probabilistic Machine Learning” Swarthmore College, Department of Computer Science.
- Padmani P .Tribhuvan,S.G. Bhirud,Amrapali P. Tribhuvan,” A Peer Review of Feature Based Opinion Mining and Summarization”(IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5 (1), 2014, 247-250 ,www.ijcsit.com.
- Carenini, G., Ng, R. and Zwart, E. Extracting Knowledge from Evaluative Text. Proceedings of the Third International Conference on Knowledge Capture (K-CAP’05), 2005.
- Dave, D., Lawrence, A., and Pennock, D. Mining the Peanut Gallery: Opinion Extraction and Semantic Classification of Product Reviews. Proceedings of International World Wide Web Conference (WWW’03), 2003.
- Zhu, Jingbo, et al. "Aspect-based opinion polling from customer reviews." IEEE Transactions on Affective Computing, Volume 2.1,pp.37-49, 2011.
- Na, Jin-Cheon, Haiyang Sui, Christopher Khoo, Syin Chan, and Yunyun Zhou. "Effectiveness of simple linguistic processing in automatic sentiment classification of product reviews." Advances in Knowledge Organization Volume9, pp. 49-54, 2004.
- Nasukawa, Tetsuya, and Jeonghee Yi. "Sentiment analysis: Capturing favorability using natural language processing." In Proceedings of the 2nd international conference on Knowledge capture, ACM, pp. 70-77, 2003.
- Li, Shoushan, Zhongqing Wang, Sophia Yat Mei Lee, and Chu-Ren Huang. "Sentiment Classification with Polarity Shifting Detection." In Asian Language Processing (IALP), 2013 International Conference on, pp. 129-132. IEEE, 2013.