

**MINI PROJECT
(2020-21)**

SENTIMENTAL ANALYSIS USING SCIKIT LEARN

REPORT



Institute of Engineering & Technology

**Submitted by
Akanksha Mishra
(181500051)
Aman Kumar
(181500073)**

***Supervised By: -*
Mr. Piyush Vasistha
Asst. Professor
Department of Computer Engineering & Applications**

ABSTRACT

Sentiment analysis or opinion mining is the computational study of people's opinions, sentiments, attitudes, and emotions expressed in written language. It is one of the most active research areas in natural language processing and text mining in recent years. Its popularity is mainly due to two reasons. First, it has a wide range of applications because opinions are central to almost all human activities and are key influencers of our behaviors. Whenever we need to make a decision, we want to hear other's opinions. Second, it presents many challenging research problems, which had never been attempted before the year 2000. Part of the reason for the lack of study before was that there was little opinionated text in digital forms. It is thus no surprise that the inception and the rapid growth of the field coincide with those of the social media on the Web. In fact, the research has also spread outside of computer science to management sciences and social sciences due to its importance to business and society as a whole. In this talk, I will start with the discussion of the mainstream sentiment analysis research and then move on to describe some recent work on modeling comments, discussions, and debates, which represents another kind of analysis of sentiments and opinions.

Sentiment classification is a way to analyze the subjective information in the text and then mine the opinion. Sentiment analysis is the procedure by which information is extracted from the opinions, appraisals and emotions of people in regards to entities, events and their attributes. In decision making, the opinions of others have a significant effect on customers ease, making choices with regards to online shopping, choosing events, products, entities. The approaches of text sentiment analysis typically work at a particular level like phrase, sentence or document level. This paper aims at analyzing a solution for the sentiment classification at a fine-grained level, namely the sentence level in which polarity of the sentence can be given by three categories as positive, negative and neutral.

TABLE OF CONTENTS

1 INTRODUCTION

1.1 Objective	4
1.2 Proposed Approach and Methods to be Employed	4

2 LITERATURE SURVEY

2.1 Models	6
2.1.1 Naïve Bayes	6
2.1.2 Bag Of Words	10
2.1.3 Support Vector Machine	14
2.1.4 Principal Component Analysis	21

3 SYSTEM ANALYSIS AND DESIGN

3.1 Software and Hardware Requirements	30
	30
3.3 Data Flow Diagrams	32

4 IMPLEMENTATION

4.1 Elimination of Special Characters and Conversion to Lower Case	38
4.2 Word Count	38
4.3 Testing and Training	39
4.3.1 Naïve Bayes	39
4.3.2 Bag of Words	39
4.3.3 Support Vector Machine	40
4.4 Sample Code	40

5 TESTING

5.1 Testing Strategies	49
5.1.1 Unit Testing	49
5.1.2 Integration Testing	49
5.1.2.1 Top Down Integration Testing	49
5.1.2.2 Bottom Up Integration Testing	50
5.1.3 System Testing	50
5.1.4 Accepting Testing	50
5.1.4.1 Alpha Testing	50
5.1.4.2 Beta Testing	50
5.2 Testing Methods	50
5.2.1 White Box Testing	50
5.2.2 Black Box Testing	51
5.3 Validation	51
5.4 Limitations	51
5.5 Test Results	51

6 SCREEN SHOTS

6.1 Naïve Bayes	53
6.2 Bag of Words	56
6.3 Support Vector Machine	59

CONCLUSION	63
-------------------	----

REFERENCES	64
-------------------	----

LIST OF FIGURES

Figure 1 : Example of Bag Of Words

Figure 2 : Hyperplane separating two classes

Figure 3: Geometrical Representation of the SVM Margin

Figure 4 : A Graph plotted with 3 Support Vectors

Figure 5 : Hyperplane separating different classes with an intercept

Figure 7: Level 0 and Level 1 Data Flow Diagram

Figure 8: Level 2 Data Flow Diagram for the process 1 (Naïve Bayes)

Figure 9 : Level 2 Data Flow Diagram for the process 2 (Bag of Words)

Figure 10 : Level 2 Data Flow Diagram for the process 3 (Support Vector Machine)

Figure 11: Phases of Software Development

Figure 12: Naïve Bayes Main Screen for Input

Figure 13: Naïve Bayes Screen with Test Data

Figure 14: Naïve Bayes Screen with output

Figure 15: Bag Of Words Main Screen for Input

Figure 16 : Bag Of Words screen with Test Data

Figure 17: Bag Of Words screen with output

Figure 18 : Support Vector Machine Main Screen for Input

Figure 19: Support Vector Machine screen with Test Data

Figure 20: Support Vector Machine Screen with Output



Aug 11, 2020

AKANKSHA MISHRA

has successfully completed

Introduction to Data Science in Python

an online non-credit course authorized by University of Michigan and offered through Coursera

A handwritten signature in black ink, appearing to read 'Chris Brooks', positioned above a horizontal line.

Christopher Brooks
Research Assistant Professor
School of Information

COURSE
CERTIFICATE



Verify at coursera.org/verify/JSE65JD2WDP2
Coursera has confirmed the identity of this individual and their participation in the course.



Aug 31, 2020

Aman Kumar

has successfully completed

Crash Course on Python

an online non-credit course authorized by Google and offered through Coursera

Google

Google

COURSE
CERTIFICATE



Verify at coursera.org/verify/RZDV2WJJEXQC
Coursera has confirmed the identity of this individual and their participation in the course.

1.INTRODUCTION

Sentiment analysis refers to the use of natural language processing, text analysis and computational linguistics to identify and extract subjective information in source materials. Generally speaking, sentiment analysis aims to determine the attitude of a speaker or a writer with respect to some topic or the overall contextual polarity of a document .The attitude may be his or her judgment or evaluation affective state, or the intended emotional communication. Sentiment analysis is the process of detecting a piece of writing for positive, negative, or neutral feelings bound to it .Humans have the innate ability to determine sentiment; however, this process is time consuming, inconsistent, and costly in a business context It's just not realistic to have people individually read tens of thousands of user customer reviews and score them for sentiment .

For example if we consider Semantria's cloud based sentiment analysis software .Semantria's cloud-based sentiment analysis software extracts the sentiment of a document and its components through the following steps:

- A document is broken in its basic parts of speech, called POS tags, which identify the structural elements of a document, paragraph, or sentence (ie Nouns, adjectives, verbs, and adverbs) .
- Sentiment-bearing phrases, such as “terrible service”, are identified through the use of specifically designed algorithms .
- Each sentiment-bearing phrase in a document is given a score based on a logarithmic scale that ranges between -10 and 10 .
- Finally, the scores are combined to determine the overall sentiment of the document or sentence Document scores range between -2 and 2 .

Semantria's cloud-based sentiment analysis software is based on Natural Language Processing and delivers you more consistent results than two humans. Using automated sentiment analysis, Semantria analyzes each document and its components based on sophisticated algorithms developed to extract sentiment from your content in a similar manner as a human – only 60,000 times faster.

Existing approaches to sentiment analysis can be grouped into three main categories:

- Keyword spotting
- Lexical affinity
- Statistical methods

Keyword spotting is the most naive approach and probably also the most popular because of its accessibility and economy. Text is classified into affect categories based on the presence of fairly unambiguous affect words like ‘happy’, ‘sad’, ‘afraid’, and ‘bored’. The weaknesses of this approach lie in two areas: poor recognition of affect when negation is involved and reliance on surface features. About its first weakness, while the approach can correctly classify the sentence “today was a happy day” as being happy, it is likely to fail on a sentence like “today wasn’t a happy day at all”. About its second weakness, the approach relies on the presence of obvious affect words that are only surface features of the prose.

In practice, a lot of sentences convey affect through underlying meaning rather than affect adjectives. For example, the text “My husband just filed for divorce and he wants to take custody of my children away from me” certainly evokes strong emotions, but uses no affect keywords, and therefore, cannot be classified using a keyword spotting approach.

Lexical affinity is slightly more sophisticated than keyword spotting as, rather than simply detecting obvious affect words, it assigns arbitrary words a probabilistic ‘affinity’ for a particular emotion. For example, ‘accident’ might be assigned a 75% probability of being indicating a negative affect, as in ‘car accident’ or ‘hurt by accident’. These probabilities are usually trained from linguistic corpora. Though often outperforming pure keyword spotting, there are two main problems with the approach. First, lexical affinity, operating solely on the word-level, can easily be tricked by sentences like “I avoided an accident” (negation) and “I met my girlfriend by accident” (other word senses). Second, lexical affinity probabilities are often biased toward text of a particular genre, dictated by the source of the linguistic corpora. This makes it difficult to develop a reusable, domain-independent model.

Statistical methods, such as Bayesian inference and support vector machines, have been popular for affect classification of texts. By feeding a machine learning algorithm a large training corpus of affectively annotated texts, it is possible for the system to not only learn the affective valence of affect keywords (as in the keyword spotting approach), but also to take into account the valence of other arbitrary keywords (like lexical affinity), punctuation, and word co-occurrence frequencies. However, traditional statistical methods are generally semantically weak, meaning that, with the exception of obvious affect keywords, other lexical or co-occurrence elements in a statistical model have little predictive value individually. As a result, statistical text classifiers only work with acceptable accuracy when given a sufficiently large text input. So, while these methods may be able to affectively classify user’s text on the

page- or paragraph- level, they do not work well on smaller text units such as sentences or clauses .

1.1 Objective

Sentiment classification is a way to analyze the subjective information in the text and then mine the opinion .Sentiment analysis is the procedure by which information is extracted from the opinions, appraisals and emotions of people in regards to entities, events and their attributes. In decision making, the opinions of others have a significant effect on customers ease, making choices with regards to online shopping, choosing events, products, entities .

1.2 Proposed Approach and methods to be Employed

Sentiment Analysis or Opinion Mining is a study that attempts to identify and analyze emotions and subjective information from text Since early 2001, the advancement of internet technology and machine learning techniques in information retrieval make Sentiment Analysis becomes popular among researchers. Besides, the emergent of social networking and blogs as a communication medium also contributes to the development of research in this area Sentiment analysis or mining refers to the application of Natural Language Processing, Computational Linguistics, and Text Analytics to identify and extract subjective information in source materials .Sentiment mining extracts attitude of a writer in a document includes writer's judgement and evaluation towards the discussed issue.

Sentiment analysis allows us to identify the emotional state of the writer during writing, and the intended emotional effect that the author wishes to give to the reader .In recent years, sentiment analysis becomes a hotspot in numerous research fields, including natural language processing (NLP), data mining (DM) and information retrieval (IR) This is due to the increasing of subjective texts appearing on the internet .Machine Learning is commonly used to classify sentiment from text .This technique involves with statistical model such ad Support Vector Machine (SVM) , Bag of Words and N  ive Bayes (NB) .The most commonly used in sentiment mining were taken from blog, twitter and web review which focusing on sentences that expressed sentiment directly .The main aim of this problem is to develop a sentiment mining model that can process the text in the mobile reviews .

CHAPTER 2
LITERATURE SURVEY

2.LITERATURE SURVEY

Sentiment analysis or opinion mining is the computational study of people's opinions, sentiments, attitudes, and emotions expressed in written language .It is one of the most active research areas in natural language processing and text mining in recent years .

2.1 Models

2.1.1 Naïve bayes

A naïve bayes classifier is a simple probability based algorithm. It uses the bayes theorem but assumes that the instances are independent of each other which is an unrealistic assumption in practical world naïve bayes classifier works well in complex real world situations .

The naïve bayes classifier algorithm can be trained very efficiently in supervised learning for example an insurance company which intends to promote a new policy to reduce the promotion costs the company wants to target the most likely prospects the company can collect the historical data for its customers ,including income range ,number of current insurance policies ,number of vehicles owned ,money invested ,and information on whether a customer has recently switched insurance companies .Using naïve bayes classifier the company can predict how likely a customer is to respond positively to a policy offering. With this information,the company can reduce its promotion costs by restricting the promotion to the most likely customers .

The naïve bayes algorithm offers fast model building and scoring both binary and multiclass situations for relatively low volumes of data this algorithm makes prediction using bayes theorem which incorporates evidence or prior knowledge in its prediction bayes theorem relates the conditional and marginal probabilities of stochastic events H and X which is mathematically stated as

$$P(H|X) = \frac{P(X|H) P(H)}{P(X)}$$

P stands for the probability of the variables within parenthesis .

$P(H)$ is the prior probability of marginal probability of H it's prior in the sense that it has not yet accounted for the information available in X .

$P(H/X)$ is the conditional probability of H, given X it is also called the posterior probability because it has already incorporated the outcome of event X .

$P(X/H)$ is the conditional probability of X given H .

$P(X)$ is the prior or marginal probability of X, which is normally the evidence .

It can also be represented as

$$\text{Posterior} = \text{likelihood} * \text{prior} / \text{normalising constant}$$

The ratio of $P(X/H)/P(X)$ is also called as standardised likelihood .

The naive Bayesian classifier works as follows:

Let T be a training set of samples, each with their class labels . There are k classes $[C_1, C_2, \dots, C_k]$ Each sample is represented by an n-dimensional vector, $X = \{x_1, x_2, \dots, x_n\}$ depicting n measured values of the n attributes $[A_1, A_2, \dots, A_n]$ respectively .

Given a sample X, the classifier will predict that X belongs to the class having the highest a posteriori probability, conditioned on X That is X is predicted to belong to the class C_i if and only if

$$P(C_i / X) > P(C_j / X) \text{ for } 1 \leq j \leq m, j \neq i$$

Thus we find the class that maximizes $P(C_i/X)$ The class C_i for which $P(C_i / X)$ is maximized is called the maximum posteriori hypothesis . By Baye's theorem

$$P(C_i/X) = (P(X/C_i) * P(C_i)) / P(X)$$

As $P(X)$ is the same for all classes, only $P(X/C_i) * P(C_i)$ need be maximized If the class a priori probabilities, $P(C_i)$, are not known, then it is commonly assumed that the classes are equally likely $[P(C_1) = P(C_2) = \dots = P(C_k)]$ and we would therefore maximize $P(X/C_i)$ Otherwise we maximize $P(X/C_i) * P(C_i)$.

Given data sets with many attributes, it would be computationally expensive to compute $P(X/C_i)$. In order to reduce computation in evaluating $P(X/C_i) * P(C_i)$, the naive assumption of class conditional independence is made This presumes that the values of the attributes are conditionally independent of one another, given the class label of the sample .

Mathematically this means that

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

The probabilities $[P(x_1/C_i), P(x_2 / C_i) \dots P(x_n / C_i)]$ can easily be estimated from the training set .Recall that here x_k refers to the value of attribute A_k for sample X .If A_k is categorical, then $P(x_k / C_i)$ is the A_k number of samples of class C_i in T having the value x_k for attribute , divided by $\text{freq}(C_i, T)$, the number of sample of class C_i in T .

In order to predict the class label of X , $P(X/C_i) * P(C_i)$ is evaluated for each class C_i .The classifier predicts that the class label of X is C_i if and only if it is the class that maximizes $P(X/C_i) * P(C_i)$.

The naïve bayes example for text classification

The training set consists of 10 Positive Reviews and 10 negative reviews and considered word counts are as follows

Positive Reviews Database	Negative Reviews Database
I = 5	I=4
Love= 20	Love=6
This= 5	This=5
Film=4	Film=3

Given test set as “I love this film” Find the sentiment for the given test set

Given training set consists of the following information

positive reviews =10

Negative reviews=10

Total no of Reviews=positive reviews+ negative reviews=20

Prior probability:

The prior probability for the positive reviews is $P(\text{positive})=10/20=0.5$

The prior probability for the negative reviews is $P(\text{negative})=10/20=0.5$

Conditional probability

The conditional probability is the probability that a random variable will take on a particular value given that the outcome for another random variable is known

The conditional probability for the word ‘I’ in positive review is

$$P(I/\text{positive})=5/10=0.5$$

The conditional probability for the word 'LOVE' in positive review is

$$P(\text{Love}/\text{positive})=20/100=0.2$$

The conditional probability for the word 'THIS' in positive review is

$$P(\text{This}/\text{positive})=5/10=0.5$$

The conditional probability for the word 'FILM' in positive review is

$$P(\text{Film}/\text{positive})=4/10=0.4$$

The conditional probability for the word 'I' in negative review is

$$P(I/\text{negative})=4/10=0.4$$

The conditional probability for the word 'LOVE' in negative review is

$$P(\text{Love}/\text{negative})=6/10=0.6$$

The conditional probability for the word 'THIS' in negative review is

$$P(\text{This}/\text{negative})=5/10=0.5$$

The conditional probability for the word 'FILM' in negative review is

$$P(\text{Film}/\text{negative})=3/10=0.3$$

Posterior probability

The posterior probabilities is the product of prior probability and conditional probabilities

$$\text{Posterior probability} = \text{prior probability} * \text{conditional probability}$$

The posterior probability for the positive review is

$$P(\text{positive})=0.5*0.5*0.5*0.4*2=0.1$$

The posterior probability for the negative review is

$$P(\text{negative})=0.5*0.6*0.3*0.5*0.4=0.018$$

The posterior probability for the positive reviews is greater than the posterior probability of the negative review

$$P(\text{positive}) > P(\text{negative})$$

The given test set "I Love This Film" is predicted by naïve bayes as a positive Sentiment

2.1.2 Bag of words

In Multinomial document model a document is represented by a feature vector with integer elements whose value is the frequency of that word in the document .Text classifiers often don't use any kind of deep representation about language often a document is represented as a bag of words (A bag is like a set that allows repeating elements) .This is an extremely simple representation it only knows which words are included in the document (and how many time search word occurs), and throws away the word order In the multinomial document model, the document feature vectors capture the frequency of words, not just their presence or absence .

Let x_i be the multinomial model feature vector for the i^{th} document D^i The t^{th} element of x_i , written x_{it} , is the count of the number of times word w_t , occurs in document D^i Let $n_i = \sum_t x_{it}$ be the total number of words in document D^i .

Let $P(w_t|C)$ again be the probability of word w_t occurring in class C , this time estimated using the word frequency information from the document feature vectors .We again make the naive Bayes assumption, that the probability of each word occurring in the document is independent of the occurrences of the other words .We can then write the document likelihood $P(D^i|C)$ as a multinomial distribution where the number of draws corresponds to the length of the document, and the proportion of drawing item t is the probability of word type t occurring in a document of class C , $P(w_t|C)$

$$P(D^i|C) \sim P(x_i|C) = (n_i! / \prod_{t=1}^{|V|} x_{it}!) \prod_{t=1}^{|V|} P(w_t|C)^{x_{it}}$$

$$\propto \prod_{t=1}^{|V|} P(w_t|C)^{x_{it}}$$

We often won't need the normalisation term $(n_i! / \prod_{t=1}^{|V|} x_{it}!)$ because it does not depend on the class, C .The numerator of the right hand side of this expression can be interpreted as the product of word likelihoods for each word in the document, with repeated words taking part for each repetition .

As for the Bernoulli model, the parameters of the likelihood are the probabilities of each word given the document class $P(w_t|C)$, and the model parameters also include the prior probabilities $P(C)$. To estimate these parameters from a training set of documents labelled with class $C = k$, let z_{ik} be an indicator variable which equals 1 when D^i has class $C=k$, and equals 0 otherwise If N is again the total number of documents, then we have:

$$P(w_t|C=k) = \sum_{i=1}^N x_{it} z_{ik} / (\sum_{s=1}^{|V|} \sum_{i=1}^N x_{is} z_{ik})$$

An estimate of the probability $P(w_t|C=k)$ as the relative frequency of w_t in documents of class $C=k$ with respect to the total number of words in documents of that class .

The prior probability of class $C=k$ is estimated as

$$P(C=k) = N_t / N$$

Thus given a training set of documents (each labelled with a class) and a set of K classes, we can estimate a multinomial text classification model as follows:

- Define the vocabulary V the number of words in the vocabulary defines the dimension of the feature vectors .
- Count the following in the training set:
 - N the total number of documents .
 - N_k the number of documents labelled with class $C=k$, for each class $k=1,...,K$.
 - x_{it} the frequency of word w_t in document D_i , computed for every word w_t in V .
- Estimate the priors $P(C=k)$.
- Estimate the likelihoods $P(w_t | C=k)$.

To classify an unlabelled document D_j , we estimate the posterior probability for each class in terms of words u which occur in our document as

$$P(C|D^i) \propto P(C) \prod_{h=1}^{len(D^i)} p(u_h/C)$$

Where u_h is the i^{th} word in document D^i

The Zero Probability Problem

A drawback of relative frequency estimates for the multinomial model is that zero counts result in estimates of zero probability .This is a bad thing because the Naive Bayes equation for the likelihood involves taking a product of probabilities if any one of the terms of the product is zero, then the whole product is zero .This means that the probability of the document belonging to that particular class is zero which is impossible .

Just because a word does not occur in a document class in the training data does not mean that it cannot occur in any document of that class .The problem is that equation of likelihood underestimates the likelihoods of words that do not occur in the data .Even if word w is not observed for class $C=k$ in the training set, we would still like $P(w | C=k) > 0$. Since probabilities must sum to 1, if unobserved words have underestimated probabilities, then those words that

are observed must have overestimated probabilities .Therefore, one way to alleviate the problem is to remove a small amount of probability allocated to observed events and distribute this across the unobserved events .A simple way to do this, sometimes called Laplace’s law of succession or add one smoothing, adds a count of one to each word type .If there are W word types in total, then instead of previous likelihood formula replaced with:

$$P(w_t | C=k) = \frac{1 + \sum_{i=1}^n x_{it} z_{ik}}{(|V| + \sum_{s=1}^{|V|} \sum_{i=1}^N x_{is} z_{ik})}$$

$$P(w_t | C) = \frac{\text{count}(w_t, c) + 1}{\sum_{w \in V} \text{count}(w, c) + |V|}$$

The denominator was increased to take account of the |V| extra “observations” arising from the “add 1” term, ensuring that the probabilities are still normalised .

Bag of words example on text classification

	Doc	Words	Class
Training	1	Chinese Beijing Chinese	c
	2	Chinese Chinese Shanghai	c
	3	Chinese Macao	c
	4	Tokyo Japan Chinese	j
Test	5	Chinese Chinese Chinese Tokyo Japan	?

Figure 1 : Example for Bag Of Words

Given dataset consists of

Total no of positive classes=3

Total no of negative classes=1

Total no of classes =positive classes + negative classes = 4

Prior probability:

It is defined as the ratio of no of objects in that class to total no of objects

$$P = N_c / N$$

P= (no of objects in that class/total no of objects)

The prior probability for the class c is

$$P(c) = 3/4$$

The prior probability for the class j is

$$P(j) = 1/4$$

Conditional Probability:

$$\hat{P}(w | c) = \frac{\text{count}(w, c) + 1}{\text{count}(c) + |V|}$$

The conditional probability for the word “Chinese” in c class is

$$P(\text{Chinese} | c) = (5+1)/(8+6) = 6/14 = 3/7$$

The conditional probability for the word “Tokyo” in c class is

$$P(\text{Tokyo} | c) = (0+1)/(8+6) = 1/14$$

The conditional probability for the word “Japan” in c class is

$$P(\text{Japan} | c) = (0+1)/(8+6) = 1/14$$

The conditional probability for the word “Chinese” in c class is

$$P(\text{Chinese} | c) = (1+1)/(3+6) = 2/9$$

The conditional probability for the word “Tokyo” in c class is

$$P(\text{Tokyo} | j) = (1+1)/(3+6) = 2/9$$

The conditional probability for the word “Japan” in c class is

$$P(\text{Japan} | j) = (1+1)/(3+6) = 2/9$$

Posterior probability

The posterior probability for the class c is

$$\begin{aligned} P(c) &= 3/4 * 3/7 * 3/7 * 3/7 * 1/14 * 1/14 \\ &= 0.0003 \end{aligned}$$

The posterior probability for the class j is

$$\begin{aligned} P(j) &= 1/4 * 2/9 * 2/9 * 2/9 * 2/9 * 2/9 \\ &= 0.0001 \end{aligned}$$

The posterior probability of the class c is greater than the posterior probability of the class j

$$P(c) > P(j)$$

Hence the given test data belongs to Class c

2.1.3 Support vector machine

Support vector machines (SVMs) are a set of related supervised learning methods used for classification and regression they belong to a family of generalized linear classifiers. In another terms, Support Vector Machine (SVM) is a classification and regression prediction tool that uses machine learning theory to maximize predictive accuracy while automatically avoiding over-fit to the data. Support Vector machines can be defined as systems which use hypothesis space of a linear functions in a high dimensional feature space, trained with a learning algorithm from optimization theory that implements a learning bias derived from statistical learning theory. Support vector machine was initially popular with the NIPS community and now is an active part of the machine learning research around the world. SVM becomes famous when, using pixel maps as input; it gives accuracy comparable to sophisticated neural networks with elaborated features in a handwriting recognition task. It is also being used for many applications, such as hand writing analysis, face analysis and so forth, especially for pattern classification and regression based applications. The foundations of Support Vector Machines (SVM) has gained popularity due to many promising features such as better empirical performance. The formulation uses the Structural Risk Minimization (SRM) principle, which has been shown to be superior to traditional Empirical Risk Minimization (ERM) principle, used by conventional neural networks. SRM minimizes an upper bound on the expected risk, where as ERM minimizes the error on the training data. It is this difference which equips SVM with a greater ability to generalize, which is the goal in statistical learning. SVMs were developed to solve the classification problem, but recently they have been extended to solve regression problems. A support vector machine (SVM) is preferred when data has exactly two classes. An SVM classifies data by finding the best hyperplane that separates all data points of one class from those of the other class. The best hyperplane for an SVM means the one with the largest margin between the two classes. Margin means the maximal width of the slab parallel to the hyperplane that has no interior data points. The support vectors are the data points that are closest to the separating hyperplane; these points are on the boundary of the slab.

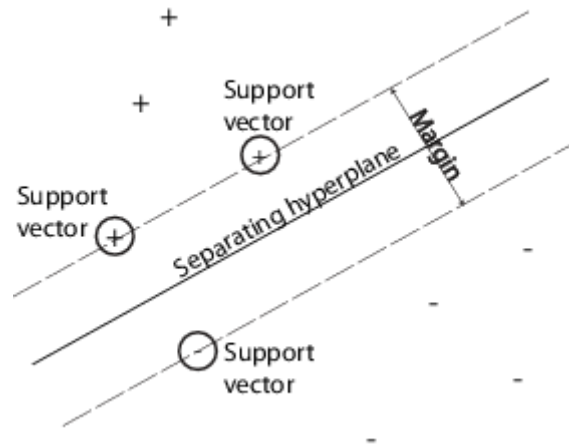


Figure 2: Hyperplane separating two classes

Assume, there is a new company j which has to be classified as solvent or insolvent according to the SVM score. In the case of a linear SVM the score looks like a DA or Logit score, which is a linear combination of relevant financial ratios $x_j = (x_{j1}, x_{j2}, \dots, x_{jd})$, where x_j is a vector with d financial ratios and x_{jk} is the value of the financial ratio number k for company j ($k=1, \dots, d$). So z_j the score of company j , can be expressed as:

$$z_j = x_j^T w + b$$

Where w is a vector which contains the weights of the d financial ratios and b is a constant. The comparison of the score with a benchmark value (which is equal to zero for a balanced sample) delivers the “forecast” of the class – solvent or insolvent – for company j .

To use this decision rule for the classification of company j , the SVM has to learn the values of the score parameters w and b on a training sample. Assume this consists of a set of n companies ($i=1, 2, \dots, n$). From a geometric point of view, calculating the value of the parameters w and b means looking for a hyperplane that best separates solvent from insolvent companies according to some criterion.

The criterion used by SVMs is based on margin maximization between the two data classes of solvent and insolvent companies. The margin is the distance between the hyperplanes bounding each class, where in the hypothetical perfectly separable case no observation may lie. By maximizing the margin, we search for the classification function that can most safely separate the classes of solvent and insolvent companies. The graph below represents a binary space with two input variables. Here crosses represent the solvent companies of the training sample and circles the insolvent ones. The threshold separating solvent and insolvent companies is the line in the middle between the two margin boundaries, which are canonically

represented as $x^T w + b = 1$ and $x^T w + b = -1$. Then the margin is $2 / \|w\|$ where $\|w\|$ is the norm of the vector w .

In a non-perfectly separable case the margin is “soft”. This means that in-sample classification errors occur and also have to be minimized. Let ε_i be a non-negative slack variable for in-sample misclassifications.

In most cases $\varepsilon_i = 0$, that means companies are being correctly classified. In the case of a positive ε_i the company i of the training sample is being misclassified. A further criterion used by SVMs for calculating w and b is that all misclassifications of the training sample have to be minimized.

Let y_i be an indicator of the state of the company, where in the case of solvency $y_i = -1$ and in the case of insolvency $y_i = 1$. By imposing the constraint that no observation may lie within the margin except some classification errors, SVMs require that either

$$\begin{aligned} x_i^T w + b &\geq 1 - \varepsilon_i \quad \text{or} \\ x_i^T w + b &\leq -1 + \varepsilon_i \end{aligned}$$

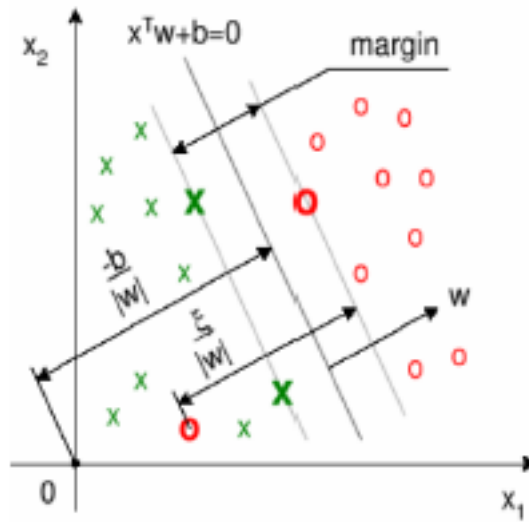


Figure 3: Geometrical Representation of the SVM Margin

The optimization problem for the calculation of w and b can thus be expressed by:

$$\begin{aligned} \min_w \quad & \frac{1}{2} \|w\|^2 + C \sum_{i=1}^n \varepsilon_i \\ & x_i^T w + b \geq -1 + \varepsilon_i \end{aligned}$$

We maximize the margin $2 / \|w\|$ by minimizing $\|w\|^2 / 2$, where the square in the form of w comes from the second term, which originally is the sum of in-sample misclassification errors $2 / \|w\|$ times the parameter C . Thus SVMs maximize the margin width while minimizing

errors .This problem is quadratic i e convex C = “capacity” is a tuning parameter, which weights in-sample classification errors and thus controls the generalization ability of an SVM .The higher is C , the higher is the weight given to in-sample misclassifications , the lower is the generalization of the machine . Low generalization means that the machine may work well on the training set but would perform miserably on a new sample .Bad generalization may be a result of overfitting on the training sample, for example, in the case that this sample shows some untypical and non-repeating data structure .By choosing a low C , the risk of overfitting an SVM on the training sample is reduced .It can be demonstrated that C is linked to the width of the margin .The smaller is C , the wider is the margin, the more and larger in-sample classification errors are permitted .

Solving the above mentioned constrained optimization problem of calibrating an SVM means searching for the minimum Lagrange function and considering $\alpha_i \geq 0$ are the Lagrange multipliers for the inequality constraint and $v_i \geq 0$ are the Lagrange multipliers for the condition $\varepsilon_i \geq 0$.This is a convex optimization problem with inequality constraints, which is solved by means of classical non-linear programming tools and the application of the Kuhn-Tucker Sufficiency Theorem .The solution of this optimisation problem is given by the saddle-point of the Lagrangian, minimized with respect to w , b , and ε and maximized with respect to α and v . The entire task can be reduced to a convex quadratic programming problem in α_i .Thus, by calculating α_i we solve our classifier construction problem and are able to calculate the parameters of the linear SVM model using the formulas

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

$$b = (x_{+1}^T, x_{-1}^T)/2$$

α_i must be non-negative, weighs different companies of the training sample

The companies, whose α_i are not equal to zero, are called support vectors and are the relevant ones for the calculation of w .Support vectors lie on the margin boundaries or, for non-perfectly separable data, within the margin .By this way, the complexity of calculations does not depend on the dimension of the input space but on the number of support vectors .Here x_{+1} and x_{-1} are any two support vectors belonging to different classes, which lie on the margin boundaries. After simplifying the equations we obtain the score z_j as a function of the scalar product of the financial ratios of the company to be classified and the financial ratios of the support vectors in the training sample, of α_i and of y_i .By comparing z_j with a benchmark value, we are able to estimate if a company has to be classified as solvent or insolvent

$$z_j = \sum_{i=1}^n y_i \alpha_i (x_i, x_j) + b$$

the support vector machine returns one class as output which is our result .

Example :

Given 3 support vectors as

$$S_1 = \begin{pmatrix} 2 \\ 1 \end{pmatrix} \quad S_2 = \begin{pmatrix} 2 \\ -1 \end{pmatrix} \quad S_3 = \begin{pmatrix} 4 \\ 0 \end{pmatrix}$$

Where S_1 and S_2 belongs to Negative class , S_3 belongs to Positive class .

Find $S_4 = \begin{pmatrix} 6 \\ 2 \end{pmatrix}$ belongs to which class either positive class or negative class using support vector machine

Given 3 support vectors as S_1, S_2, S_3 where S_1, S_2 belongs to Negative class and S_3 belongs to Positive class.

$$S_1 = \begin{pmatrix} 2 \\ 1 \end{pmatrix} \quad S_2 = \begin{pmatrix} 2 \\ -1 \end{pmatrix} \quad S_3 = \begin{pmatrix} 4 \\ 0 \end{pmatrix}$$

A graph is plotted for these 3 support vectors

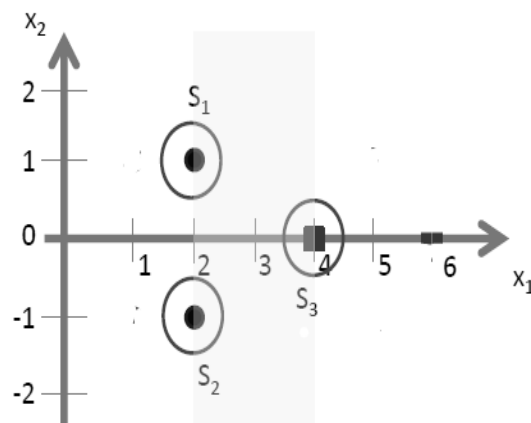


Figure 4: A graph plotted with 3 support vectors

we will use vectors augmented with a 1 as a bias input .we will differentiate these with an over-tilde.

$$S_1 = \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} \quad \tilde{S}_1 = \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix}$$

$$S_2 = \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix} \quad \tilde{S}_2 = \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix}$$

$$S_3 = \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix} \quad \tilde{S}_3 = \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix}$$

Find the 3 parameters $\alpha_1, \alpha_2, \alpha_3$ from the 3 linear equations

$$\alpha_1 \widetilde{S_1} \cdot \widetilde{S_1} + \alpha_2 \widetilde{S_2} \cdot \widetilde{S_1} + \alpha_3 \widetilde{S_3} \cdot \widetilde{S_1} = -1 \text{ (-ve class)}$$

$$\alpha_1 \widetilde{S_1} \cdot \widetilde{S_2} + \alpha_2 \widetilde{S_2} \cdot \widetilde{S_2} + \alpha_3 \widetilde{S_3} \cdot \widetilde{S_2} = -1 \text{ (-ve class)}$$

$$\alpha_1 \widetilde{S_1} \cdot \widetilde{S_3} + \alpha_2 \widetilde{S_2} \cdot \widetilde{S_3} + \alpha_3 \widetilde{S_3} \cdot \widetilde{S_3} = +1 \text{ (+ve class)}$$

Substitute the values of $\tilde{S}_1, \tilde{S}_2, \tilde{S}_3$ in the above 3 Linear equations

$$\tilde{S}_1 = \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} \quad \tilde{S}_2 = \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix} \quad \tilde{S}_3 = \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix}$$

$$\alpha_1 \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} + \alpha_2 \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} + \alpha_3 \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} = -1$$

$$\alpha_1 \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix} + \alpha_2 \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix} + \alpha_3 \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix} = -1$$

$$\alpha_1 \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix} + \alpha_2 \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix} + \alpha_3 \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix} = +1$$

After substituting the values of $\tilde{S}_1, \tilde{S}_2, \tilde{S}_3$ in 3 linear equations and simplifying it reduces into

$$6\alpha_1 + 4\alpha_2 + 9\alpha_3 = -1$$

$$4\alpha_1 + 6\alpha_2 + 9\alpha_3 = -1$$

$$9\alpha_1 + 9\alpha_2 + 17\alpha_3 = +1$$

By solving the above 3 equations we will get the $\alpha_1, \alpha_2, \alpha_3$ values

$$\alpha_1 = -3.25$$

$$\alpha_2 = -3.25$$

$$\alpha_3 = 3.5$$

The Hyperplane that separates the positive class from negative class is given by formula

$$\tilde{w} = \sum_i \alpha_i \tilde{S}_i$$

Now substituting the values of $\hat{S}_1, \hat{S}_2, \hat{S}_3$ in the above formula

$$\begin{aligned} \tilde{w} &= \alpha_1 \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} + \alpha_2 \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix} + \alpha_3 \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix} \\ \tilde{w} &= (-3.25) \cdot \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} + (-3.25) \cdot \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix} + (3.5) \cdot \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ -3 \end{pmatrix} \end{aligned}$$

After simplifying

$$\tilde{w} = (-3.25) \cdot \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} + (-3.25) \cdot \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix} + (3.5) \cdot \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ -3 \end{pmatrix}$$

Our vectors are augmented with a bias. Hence we can equate the entry in as the hyper plane with an offset b . The separating hyper plane equation $y = wx + b$ with $w = \frac{1}{0}$ and offset $b = -3$. Hence the positive class is separated from the negative class at offset $b = -3$.

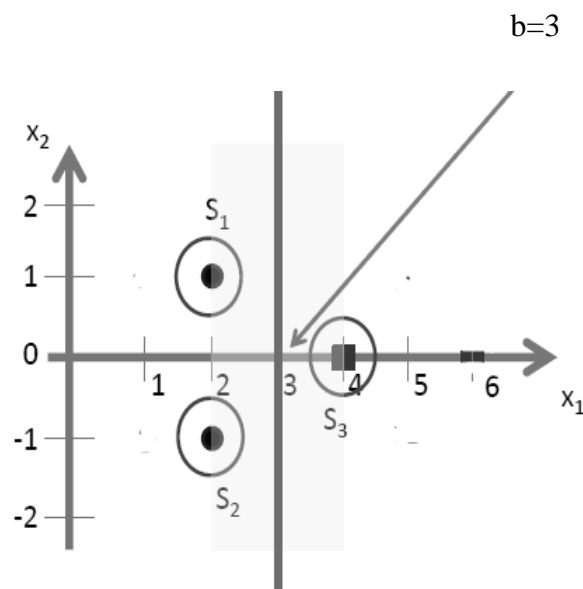


Figure 5 : Hyperplane separating two different classes with an intercept

Given $S4 = \frac{6}{2}$ for finding the class for this new support vector we use the formula

$= w x$ in this case if $w x > \text{offset}$ positive class
 $w x < \text{offset}$ negative class

we know $w = \frac{1}{0}$ and $x = \frac{6}{2}$ and $\text{offset} = 3$

$$w x = \frac{1}{0} * \frac{6}{2} = 6$$

$$w x > \text{offset} = 6 > 3$$

hence support vector machine classifies this newly added point belongs to the positive class.

2.1.4 Principle component analysis

PCA is a dimensionality reduction method in which a covariance analysis between factors takes place. The original data is remapped into a new coordinate system based on the variance within the data. PCA applies a mathematical procedure for transforming a number of (possibly) correlated variables into a (smaller) number of uncorrelated variables called principal components. The first principal component accounts for as much of the variability in the data as possible, and each succeeding component accounts for as much of the remaining variability as possible.

PCA is useful when there is data on a large number of variables, and (possibly) there is some redundancy in those variables. In this case, redundancy means that some of the variables are correlated with one another and because of this redundancy, PCA can be used to reduce the observed variables into a smaller number of principal components that will account for most of the variance in the observed variables.

PCA is recommended as an exploratory tool to uncover unknown trends in the data. The technique has found application in fields such as face recognition and image compression, and is a common technique for finding patterns in data of high dimension.

For a given data set with n no of observations. Each observation consists of x variants for calculating the principal components the PCA algorithm follows the following 5 main steps

Subtract the mean from each of the data dimensions:

The mean subtracted is the average across each dimension. This produces a data set whose mean is zero

$$\text{Mean } \bar{x} = \sum_{i=1}^n x_i / n$$

Calculate the covariance matrix:

Variance and Covariance are a measure of the spread of a set of points around their center of mass (mean). Variance is the measure of the deviation from the mean for points in one dimension. Variance is calculated with the formula

$$\sigma^2 = \sum (x - \bar{x})^2 / n - 1$$

Covariance as a measure of how much each of the dimensions vary from the mean with respect to each other. Covariance is measured between 2 dimensions to see if there is a relationship between the 2 dimensions. The covariance between one dimension and itself is the variance. For example a 3-dimensional data set (x,y,z), then you could measure the covariance between the x and y dimensions, the y and z dimensions, and the x and z dimensions. Measuring the covariance between x and x, or y and y, or z and z would give you the variance of the x, y and z dimensions respectively

$$C = \begin{bmatrix} \text{var}(x, x) & \text{cov}(x, y) & \text{cov}(x, z) \\ \text{cov}(y, x) & \text{var}(y, y) & \text{cov}(y, z) \\ \text{cov}(z, x) & \text{cov}(z, y) & \text{var}(z, z) \end{bmatrix}$$

Diagonal is the variances of x, y and z $\text{cov}(x, y) = \text{cov}(y, x)$ hence matrix is symmetrical about the diagonal. For a N-dimensional data will result in NxN covariance matrix

$$\text{Var}[X] = \begin{bmatrix} \text{Var}[X_1] & \text{Cov}[X_1, X_2] & \dots & \text{Cov}[X_1, X_K] \\ \text{Cov}[X_1, X_2] & \text{Var}[X_2] & \dots & \text{Cov}[X_2, X_K] \\ \vdots & \vdots & \ddots & \vdots \\ \text{Cov}[X_1, X_K] & \text{Cov}[X_2, X_K] & \dots & \text{Var}[X_K] \end{bmatrix}$$

Exact value is not as important as it's sign. A positive value of covariance indicates both dimensions increase or decrease together. Example: as the number of hours studied increases, the marks in that subject increase. A negative value indicates while one increases the other decreases, or vice-versa. If covariance is zero: the two dimensions are independent of each other.

Calculate the eigenvectors and eigenvalues of the covariance matrix:

The covariance matrix is a square, the calculation of eigenvectors and eigenvalues are possible for this matrix. These are rather important, as they tell us useful information about our data.

Eigen values are calculated with the formula $|C - \lambda I| = 0$

Where I is the identity matrix

Eigen vectors are calculated with the formula $[C - \lambda I][k] = 0$

It is important to notice that these eigenvectors are both unit eigenvectors ie Their lengths are both 1 which is very important for PCA .

Choose components and form a feature vector:

The eigenvectors and eigenvalues obtained from the covariance matrix will have quite different values .In fact, it turns out that the eigenvector with the highest eigenvalue is the principle component of the data set .

In general, once eigenvectors are found from the covariance matrix, the next step is to order them by eigenvalue, highest to lowest This gives you the components in order of significance Now, if you like, you can decide to ignore the components of lesser significance .You do lose some information, but if the eigenvalues are small, you don't lose much If you leave out some components, the final data set will have less n dimensions than the original .To be precise, if you originally have dimensions in your data, and so you calculate n eigenvectors and eigenvalues, and then you choose only the first p eigenvectors, then the final data set has only p dimensions .

To form a feature vector, which is just a fancy name for a matrix of vectors . This is constructed by taking the eigenvectors that you want to keep from the list of eigenvectors, and forming a matrix with these eigenvectors in the columns

$$\text{FeatureVector} = (\text{eigenvector1}, \text{eigenvector2}, \dots, \text{eigenvector}_n)$$

Deriving the new data set:

$$\text{FinalData} = \text{RowFeatureVector} \times \text{RowDataAdjusted}$$

Where RowFeatureVector is the matrix with the eigenvectors in the columns transposed so the eigenvectors are now in the rows and the most significant are in the top.

RowDataAdjusted is the mean-adjusted data transposed i.e the data items are in each column, with each row holding a separate dimension. It will give us the original data solely in terms of the vectors we chose .

Principle component analysis example

X1	X2
1 4000	1 6500
1 6000	1 9700
-1 4000	-1 7750
-2 0000	-2 5250
-3 0000	-3 9500
2 4000	3 0750
1 5000	2 0250
2 3000	2 7500
-3 2000	-4 0500
-4 1000	-4 8500

Table 1 : Principle Component Analysis Data Set

Given a two dimensional data as x1,x2 Dimensionality reduction using Principle component analysis can be done in 5 steps:

- To obtain covariance matrix .
- To obtain eigen values .
- To obtain eigen vectors .
- To obtain coordinates of data point in the direction of eigen vectors .

Covariance matrix:

The covariance matrix is obtained by using the formula

$$cov(x, y) = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{(n - 1)}$$

$$= \begin{matrix} Cov(x1, x1) & Cov(x1, x2) \\ Cov(x2, x1) & Cov(x2, x2) \end{matrix}$$

X1	X2	C= X1-X1bar	D= X2-X2bar	C*D
1 4000	1 6500	1 8500	2 2175	4 1024
1 6000	1 9750	2 0500	2 5425	5 3121
-1 4000	-1 7750	-0 9500	-1 2075	1 1471
-2 0000	-2 5250	-1 5500	-1 9575	3 0341
-3 0000	-3 9500	-2 5500	-3 3825	8 6254
2 4000	3 0750	2 8500	3 6425	10 3811
1 5000	2 0250	1 9500	2 5925	5 0554
2 3000	2 7500	2 7500	3 3175	9 1231
-3 2000	-4 0500	-2 7500	-3 4825	9 5769
-4 1000	-4 8500	-3 6500	-4 2825	15 6311
A=-0 4500	B= -0 5675			

Table 2 : Principle Component Analysis Mean and Variance calculation

$$\text{Cov}(x_1, x_1) = 6.4228$$

$$\text{Cov}(x_1, x_2) = 7.9876$$

$$\text{Cov}(x_2, x_1) = 7.9876$$

$$\text{Cov}(x_2, x_2) = 9.9528$$

$$\text{Covariance matrix} = \begin{bmatrix} 6.4228 & 7.9876 \\ 7.9876 & 9.9528 \end{bmatrix}$$

Eigen values

The eigen values are calculated by using the formula

$$|\text{Covariance matrix} - \lambda I| = 0$$

$$\text{Where Covariance matrix} = \begin{bmatrix} 6.4228 & 7.9876 \\ 7.9876 & 9.9528 \end{bmatrix}$$

$$\text{Identity matrix } I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

After substituting the covariance matrix and identity matrix in the formula

$$\begin{bmatrix} 6.4228 & 7.9876 \\ 7.9876 & 9.9528 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = 0$$

After simplifying we get Eigen values as

$$\lambda_1 = 16.36809984$$

$$\lambda_2 = 0.007462657$$

Eigen vector

The eigen vectors for the given two dimension data set are obtained by using the formula

$$(A - \lambda I) \mathbf{x} = \mathbf{0}$$

If we consider $\lambda = 16.36809984$

$$(A - \lambda I) = \begin{pmatrix} 6.4228 & 7.9876 \\ 7.9876 & 9.9528 \end{pmatrix} - (16.36809984) \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} -9.9453 & 7.9876 \\ 7.9876 & -6.4153 \end{pmatrix}$$

$$(A - \lambda I) \mathbf{x} = \mathbf{0}$$

$$= \begin{pmatrix} -9.9453 & 7.9876 \\ 7.9876 & -6.4153 \end{pmatrix} * \begin{pmatrix} a \\ b \end{pmatrix} = 0$$

By using row reduction and simplifying we get eigen vector as

$$\text{Eigen vector} = \begin{pmatrix} 0.6262 \\ 0.7797 \end{pmatrix}$$

If we consider $\lambda = 0.007462657$

$$(A - \lambda I) = \begin{pmatrix} 6.4228 & 7.9876 \\ 7.9876 & 9.9528 \end{pmatrix} - (0.007462657) \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} 6.4153 & 7.9876 \\ 7.9876 & 8.9925 \end{pmatrix}$$

$$(A - \lambda I) \mathbf{x} = \mathbf{0}$$

$$= \begin{pmatrix} 6.4153 & 7.9876 \\ 7.9876 & 8.9925 \end{pmatrix} * \begin{pmatrix} a \\ b \end{pmatrix} = 0$$

By using row reduction and simplifying we get eigen vector as

$$\text{Eigen vector} = \begin{pmatrix} 0.7797 \\ -0.6262 \end{pmatrix}$$

coordinates of data point in the direction of eigen vectors

This is obtained by multiplying centered data matrix to the eigen vector matrix

$$\text{Eigen vector matrix} = \begin{pmatrix} 0.6262 & 0.7797 \\ 0.7797 & -0.6262 \end{pmatrix}$$

X1-X1bar	X2-X2bar
1 8500	2 2175
2 0500	2 5425
-0 9500	-1 2075
-1 5500	-1 9575
-2 5500	-3 3825
2 8500	3 6425
1 9500	2 5925
2 7500	3 3175
-2 7500	-3 4825
-3 6500	-4 2825

Table 3 : Centered data matrix

X1-X1bar	X2-X2bar	$\begin{pmatrix} 0.6262 & 0.7797 \\ 0.7797 & -0.6262 \end{pmatrix} =$	Projection on the line of 1 st principle component	Projection on the line of 2 nd principle component
1 8500	2 2175		2 88737	0 05380
2 0500	2 5425		3 26600	00622
-0 9500	-1 2075		-1 53633	0 01545
-1 5500	-1 9575		-2 49680	0 01729
-2 5500	-3 3825		-4 23402	0 12995
2 8500	3 6425		4 62459	-0 05886
1 9500	2 5925		3 2237	-0 10306
2 7500	3 3175		4 30858	0 06669
-2 7500	-3 4825		-4 43722	0 03664
-3 6500	-4 2825		-5 62453	-0 16411
		variance	16 3680977	0 007462657

Table 4 : Principle Component Projection Calculation

The variance of projections on the line of first principle component is 16 36809775 .

The variance of projections on the line of first principle component is 0 007462657 .

The variances of projections in the line of principle component is equal to the eigen values of the principle components .First eigen vector is able to explain around 99% of total variance .

- **Data Flow:** The flow of data from process to process.

Data flow diagram :

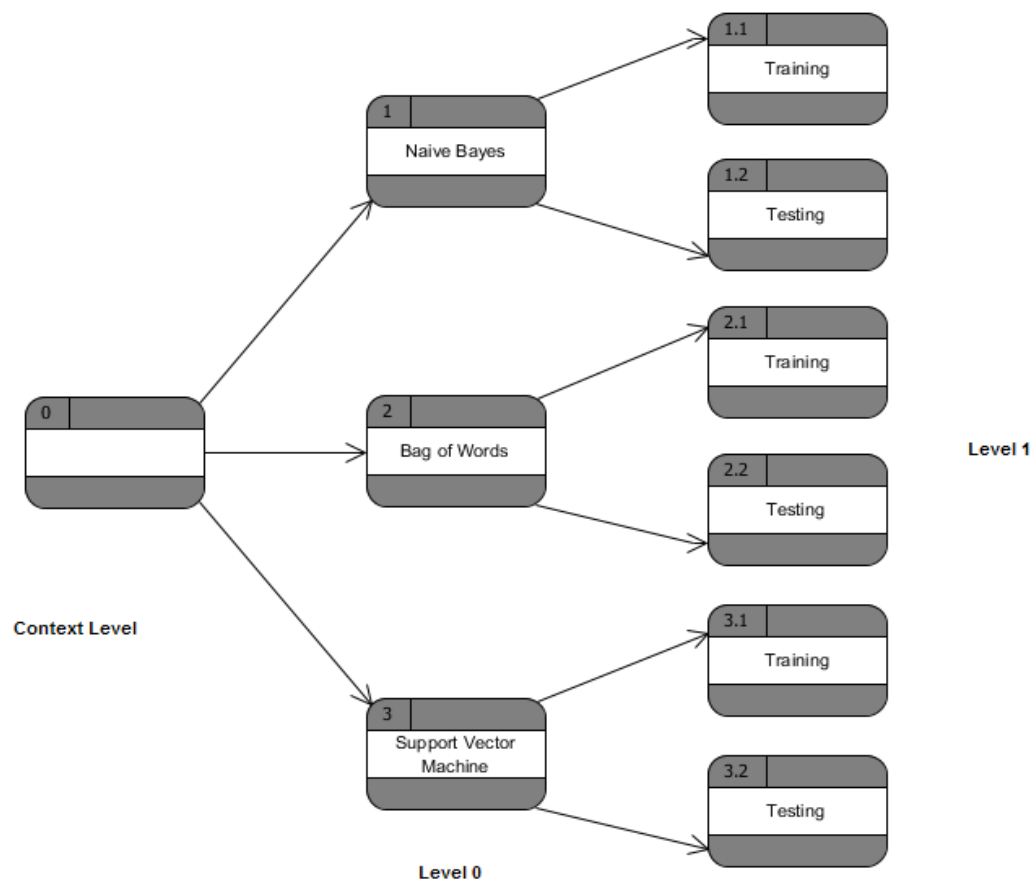


Figure 7 : Level 0 and Level 1 Data Flow Diagram

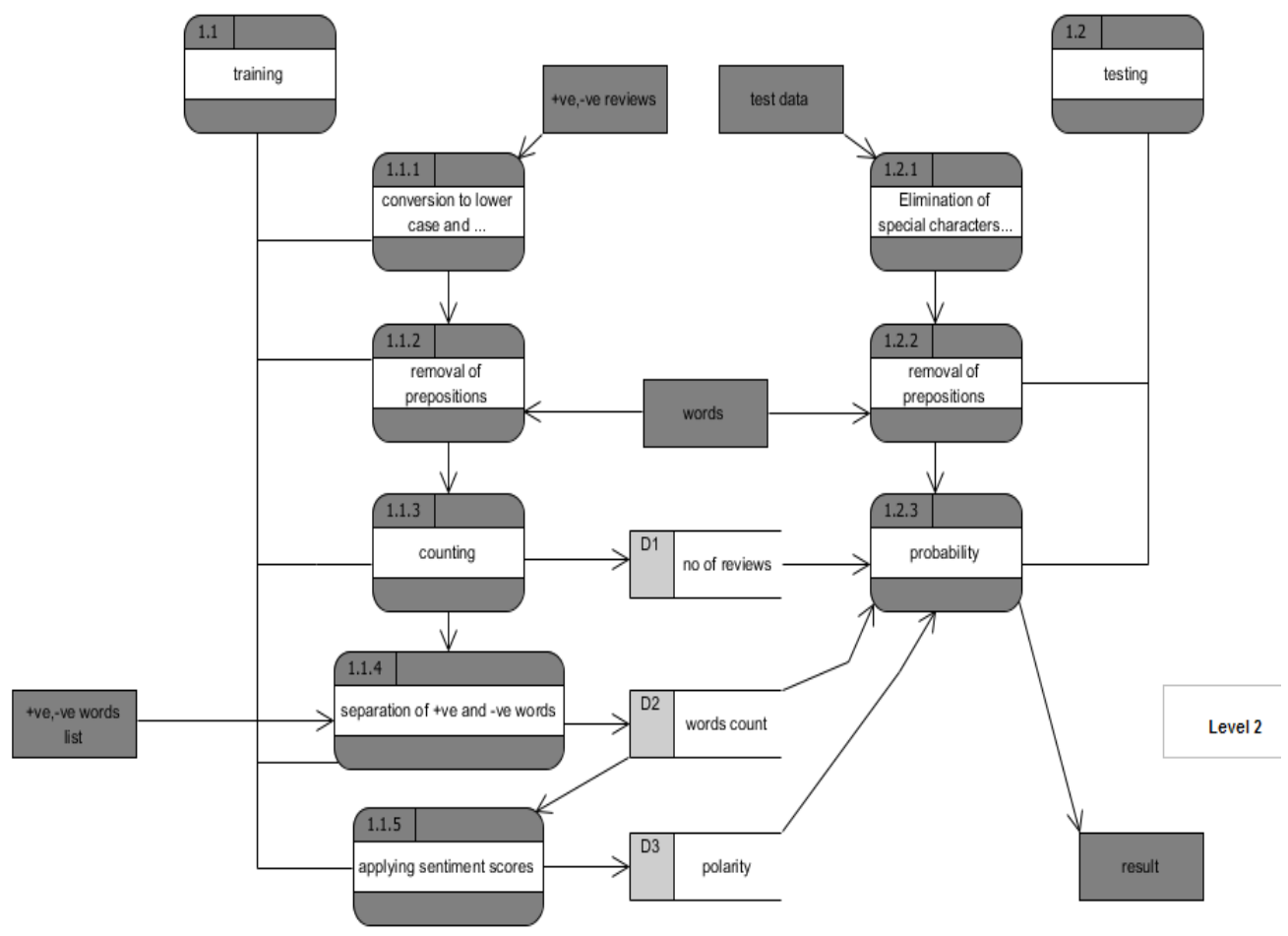


Figure 8: Level 2 Data Flow Diagram for the process 1 (Naïve Bayes)

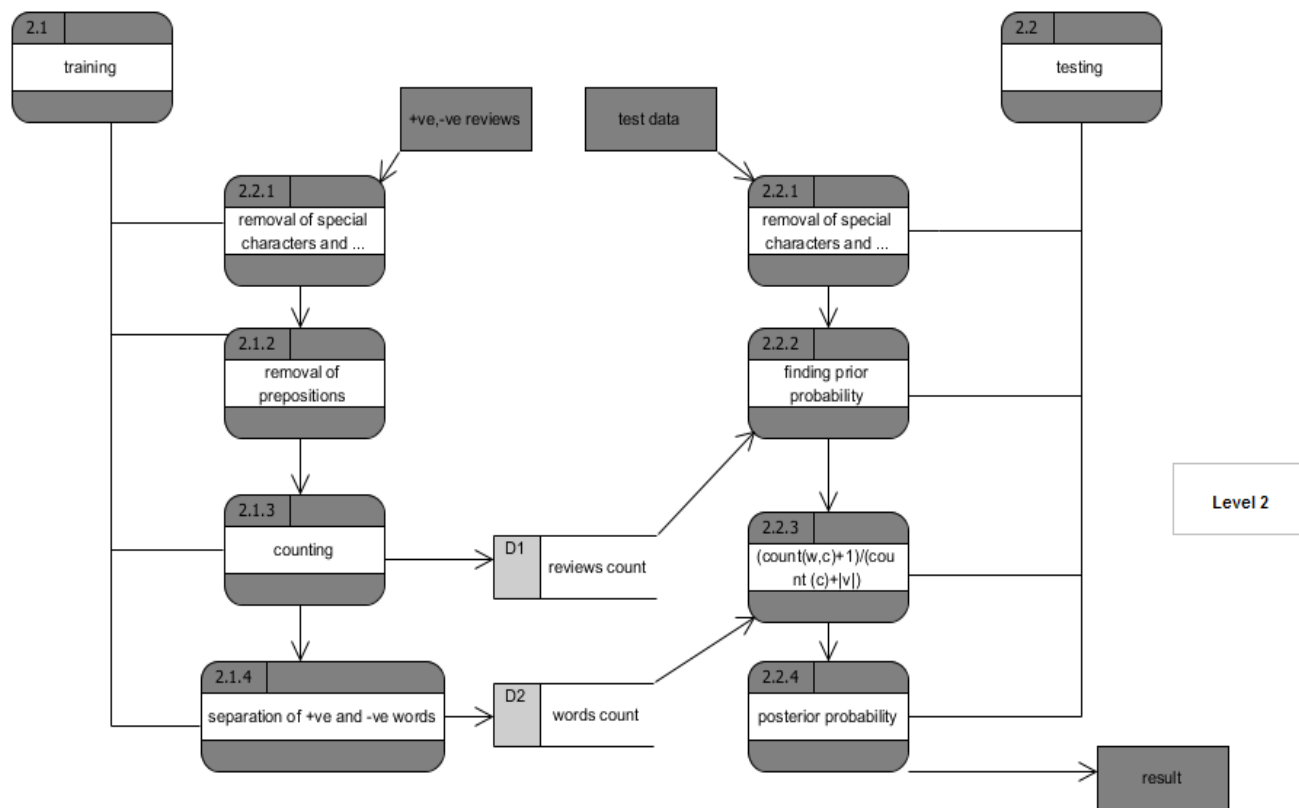


Figure 9: Level 2 Data Flow Diagram for the process 2 (Bag of Words)

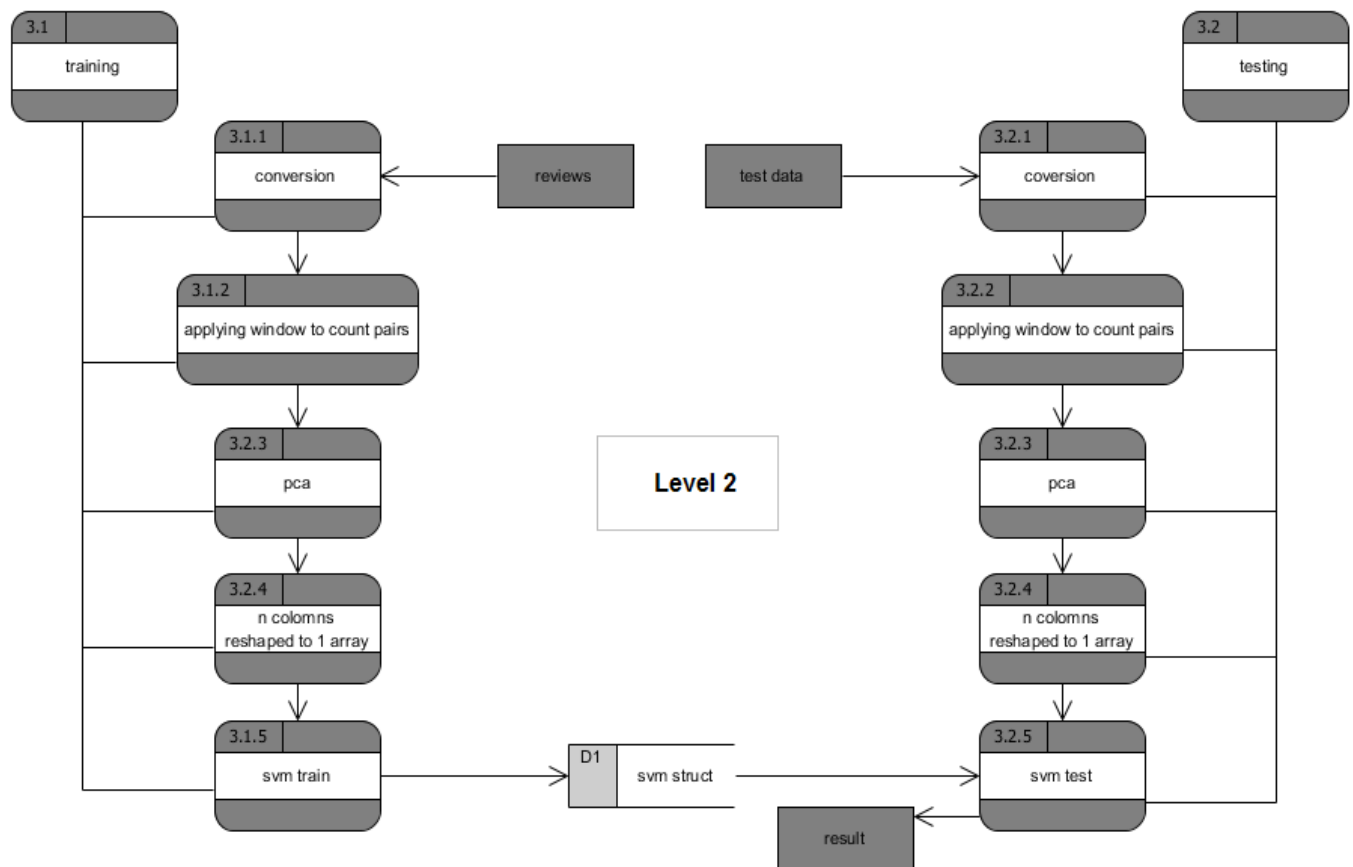


Figure 11: Level 2 Data Flow Diagram for the process 3 (Support Vector Machine)

CHAPTER 5

TESTING

5.TESTING

Testing is the process of evaluating a system or its component's with the intent to find that whether it satisfies the specified requirements or not .This activity results in the actual, expected and difference between their results i.e testing is executing a system in order to identify any gaps, errors or missing requirements in contrary to the actual desire or requirements.

5.1 Testing Strategies

In order to make sure that system does not have any errors, the different levels of testing strategies that are applied at different phases of software development are

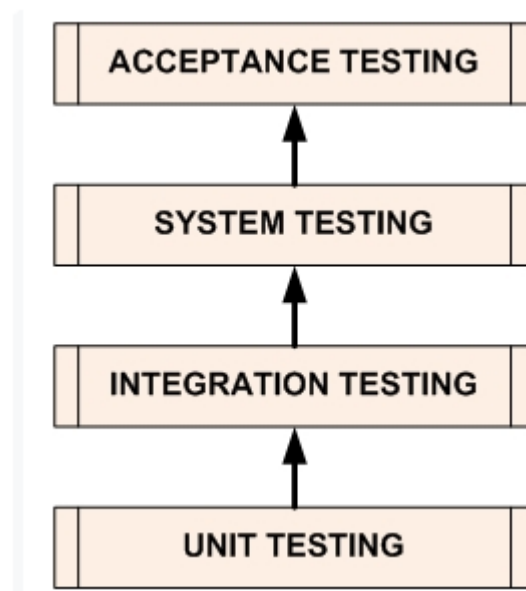


Figure 11 : Phases of Software Development

5.1.1 Unit Testing

The goal of unit testing is to isolate each part of the program and show that individual parts are correct in terms of requirements and functionality.

5.1.2 Integration Testing

The testing of combined parts of an application to determine if they function correctly together is Integration testing .This testing can be done by using two different methods

5.1.2.1 Top Down Integration testing

In Top-Down integration testing, the highest-level modules are tested first and then progressively lower-level modules are tested.

5.1.2.2 Bottom-up Integration testing

Testing can be performed starting from smallest and lowest level modules and proceeding one at a time .When bottom level modules are tested attention turns to those on the next level that use the lower level ones they are tested individually and then linked with the previously examined lower level modules.In a comprehensive software development environment, bottom-up testing is usually done first, followed by top-down testing.

5.1.3 System Testing

This is the next level in the testing and tests the system as a whole .Once all the components are integrated, the application as a whole is tested rigorously to see that it meets Quality Standards.

5.1.4 Acceptance Testing

The main purpose of this Testing is to find whether application meets the intended specifications and satisfies the client's requirements .We will follow two different methods in this testing.

5.1.4.1 Alpha Testing

This test is the first stage of testing and will be performed amongst the teams .Unit testing, integration testing and system testing when combined are known as alpha testing. During this phase, the following will be tested in the application:

- Spelling Mistakes.
- Broken Links.
- The Application will be tested on machines with the lowest specification to test loading times and any latency problems.

5.1.4.2 Beta Testing

In beta testing, a sample of the intended audience tests the application and send their feedback to the project team .Getting the feedback, the project team can fix the problems before releasing the software to the actual users.

5.2 Testing Methods

5.2.1 White Box Testing

White box testing is the detailed investigation of internal logic and structure of the Code. To perform white box testing on an application, the tester needs to possess knowledge of the internal working of the code .The tester needs to have a look inside the source code and find out which unit/chunk of the code is behaving inappropriately.

5.2.2 Black Box Testing

The technique of testing without having any knowledge of the interior workings of the application is Black Box testing .The tester is oblivious to the system architecture and does not have access to the source code.Typically, when performing a black box test, a tester will interact with the system's user interface by providing inputs and examining outputs without knowing how and where the inputs are worked upon.

5.3 Validation

All the levels in the testing (unit,integration,system) and methods (black box,white box)are implemented on our application successfully and the results obtained as expected .

5.4 Limitations

The execution time for support vector machine is more so that the user may not receive the result fast.

5 5 Test Results

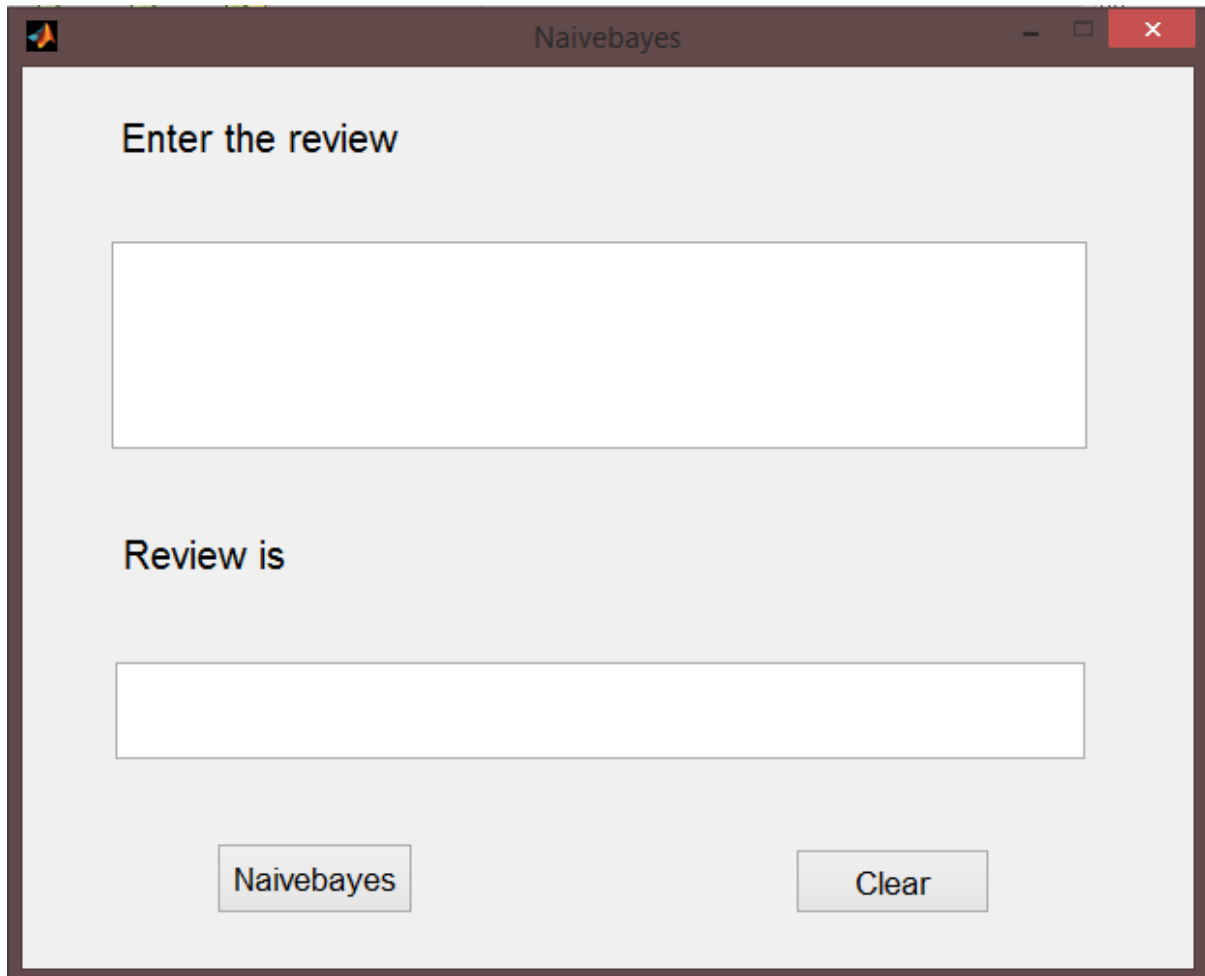
The testing is done among the team members and by the end users. It satisfies the specified requirements and finally we obtained the results as expected.

CHAPTER 6

SCREEN SHOTS

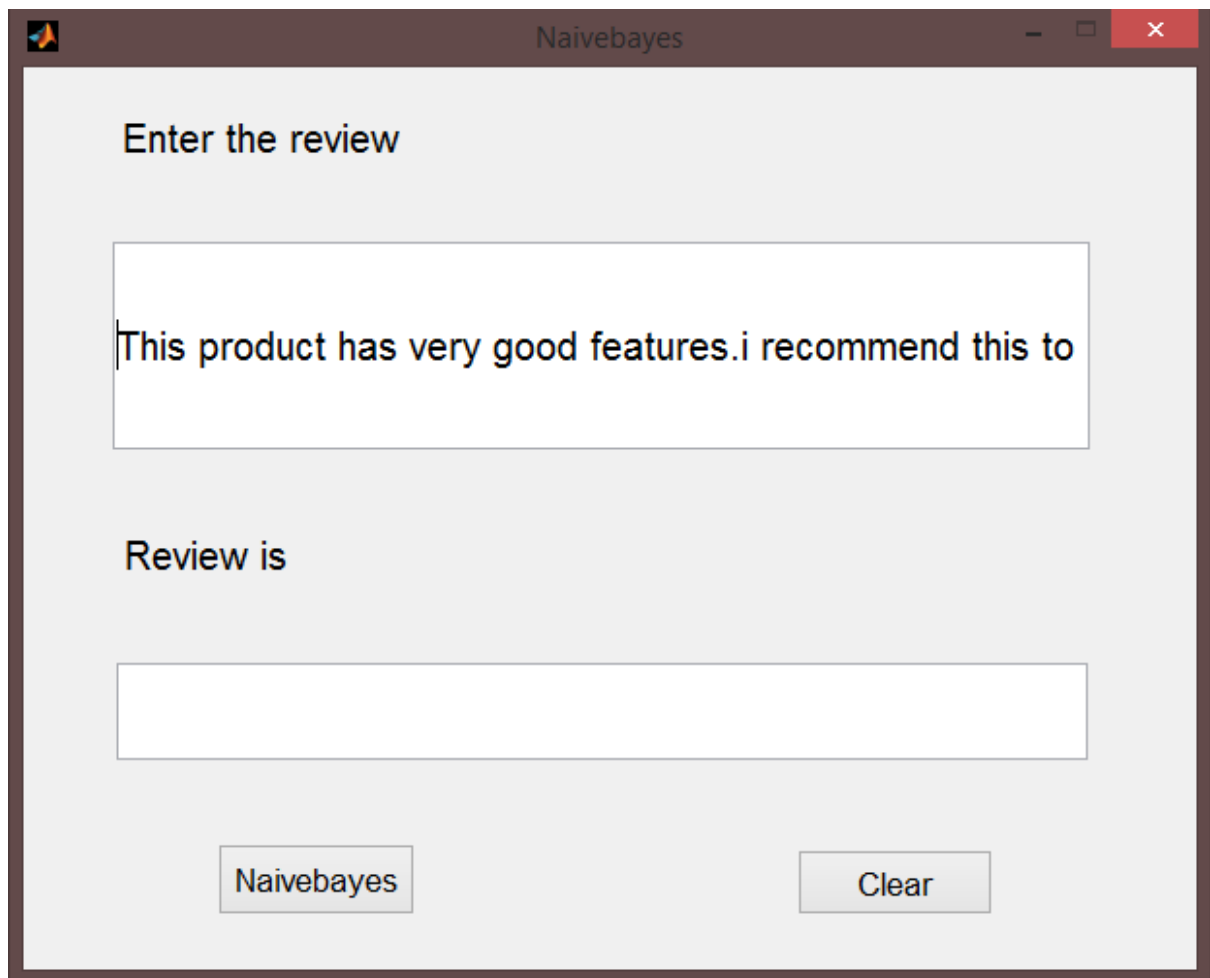
6.SCREEN SHOTS

6.1 Naïve Bayes



The image shows a software window titled "Naivebayes" with a standard Windows-style title bar (minimize, maximize, close buttons). The window has a light gray background. At the top, the text "Enter the review" is displayed. Below it is a large, empty white rectangular text input field. Further down, the text "Review is" is displayed. Below this is another empty white rectangular text input field. At the bottom of the window, there are two buttons: "Naivebayes" on the left and "Clear" on the right. Both buttons have a light gray background and a thin black border.

Figure 12 : Naïve Bayes Main Screen for Input



A screenshot of a Java Swing window titled "Naivebayes". The window has a light gray background and a dark gray border. At the top, there is a label "Enter the review". Below it is a large text area containing the text "This product has very good features.i recommend this to". Underneath the text area is a label "Review is". Below this label is an empty text area. At the bottom of the window, there are two buttons: "Naivebayes" on the left and "Clear" on the right.

Naivebayes

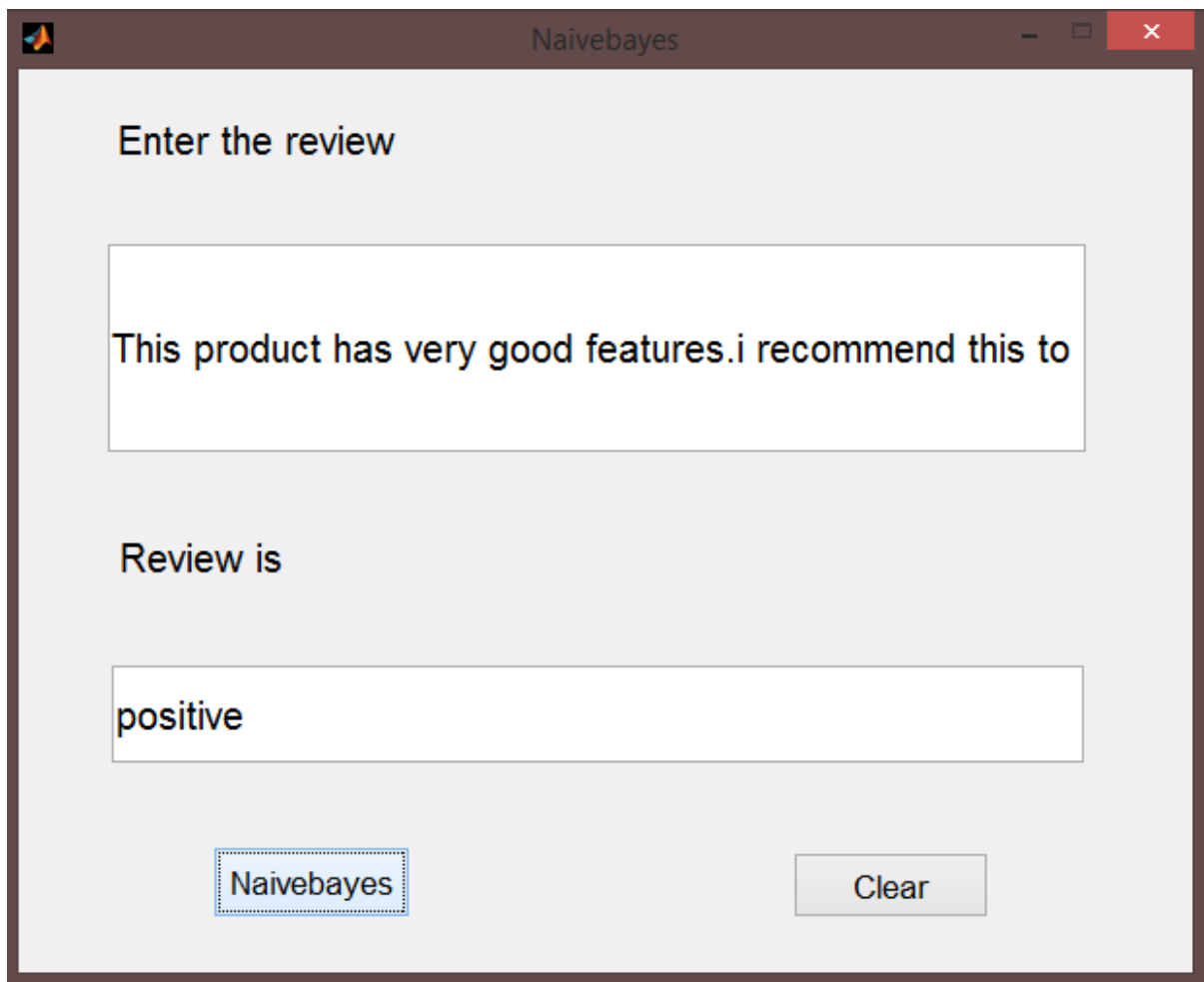
Enter the review

This product has very good features.i recommend this to

Review is

Naivebayes Clear

Figure 13 : Naïve Bayes Screen with Test Data



A screenshot of a Java Swing window titled "Naivebayes". The window has a light gray background and a dark gray title bar with standard window controls. It contains two text input fields and two buttons. The first input field, labeled "Enter the review", contains the text "This product has very good features.i recommend this to". The second input field, labeled "Review is", contains the text "positive". At the bottom, there is a button labeled "Naivebayes" with a dotted border and a disabled button labeled "Clear".

Naivebayes

Enter the review

This product has very good features.i recommend this to

Review is

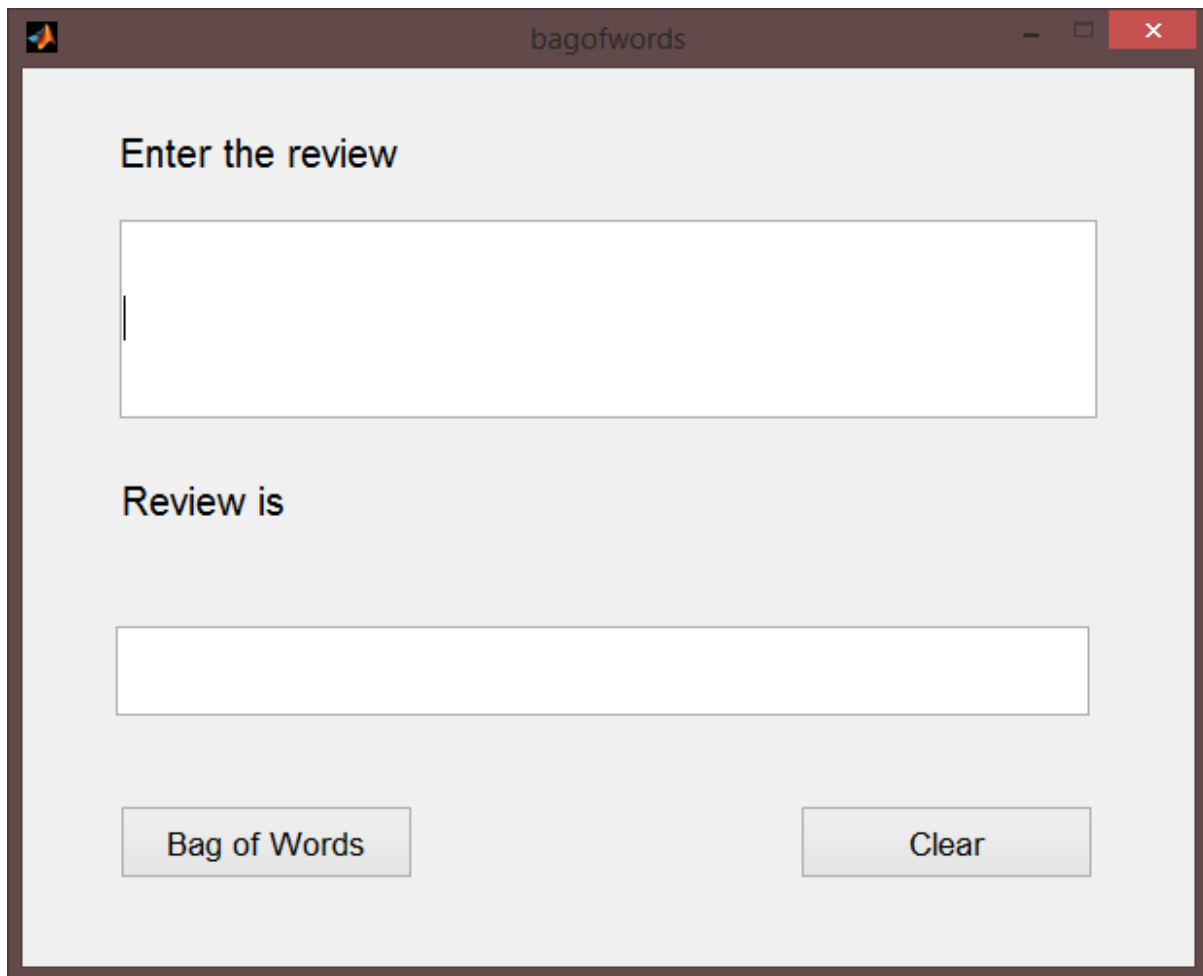
positive

Naivebayes

Clear

Figure 14 : Naïve Bayes Screen with output

6.2 Bag of words



The image shows a window titled "bagofwords" with a standard operating system title bar (minimize, maximize, close buttons). The window has a light gray background. At the top, the text "Enter the review" is displayed. Below it is a large, empty text input field. Further down, the text "Review is" is displayed. Below that is another empty text input field. At the bottom of the window, there are two buttons: "Bag of Words" on the left and "Clear" on the right. Both buttons have a light gray background and a thin border.

Figure 15 : Bag Of Words Main Screen for Input

The image shows a Java Swing window titled "bagofwords". Inside the window, there is a label "Enter the review" above a text area containing the text "This product has very good features. I recommend this to every one.". Below this, there is a label "Review is" above an empty text field. At the bottom of the window, there are two buttons: "Bag of Words" on the left and "Clear" on the right.

Figure 16 : Bag Of Words screen with Test Data

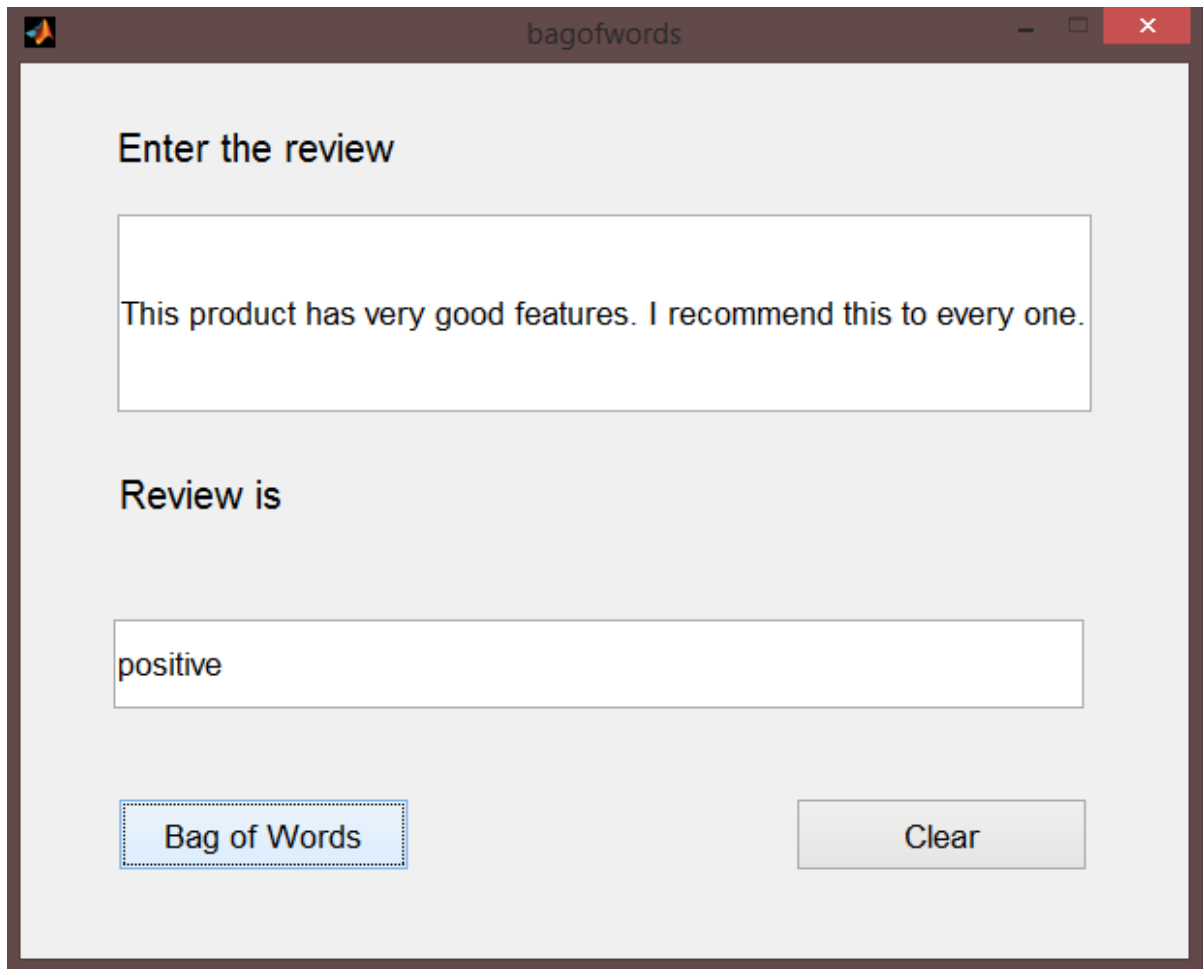
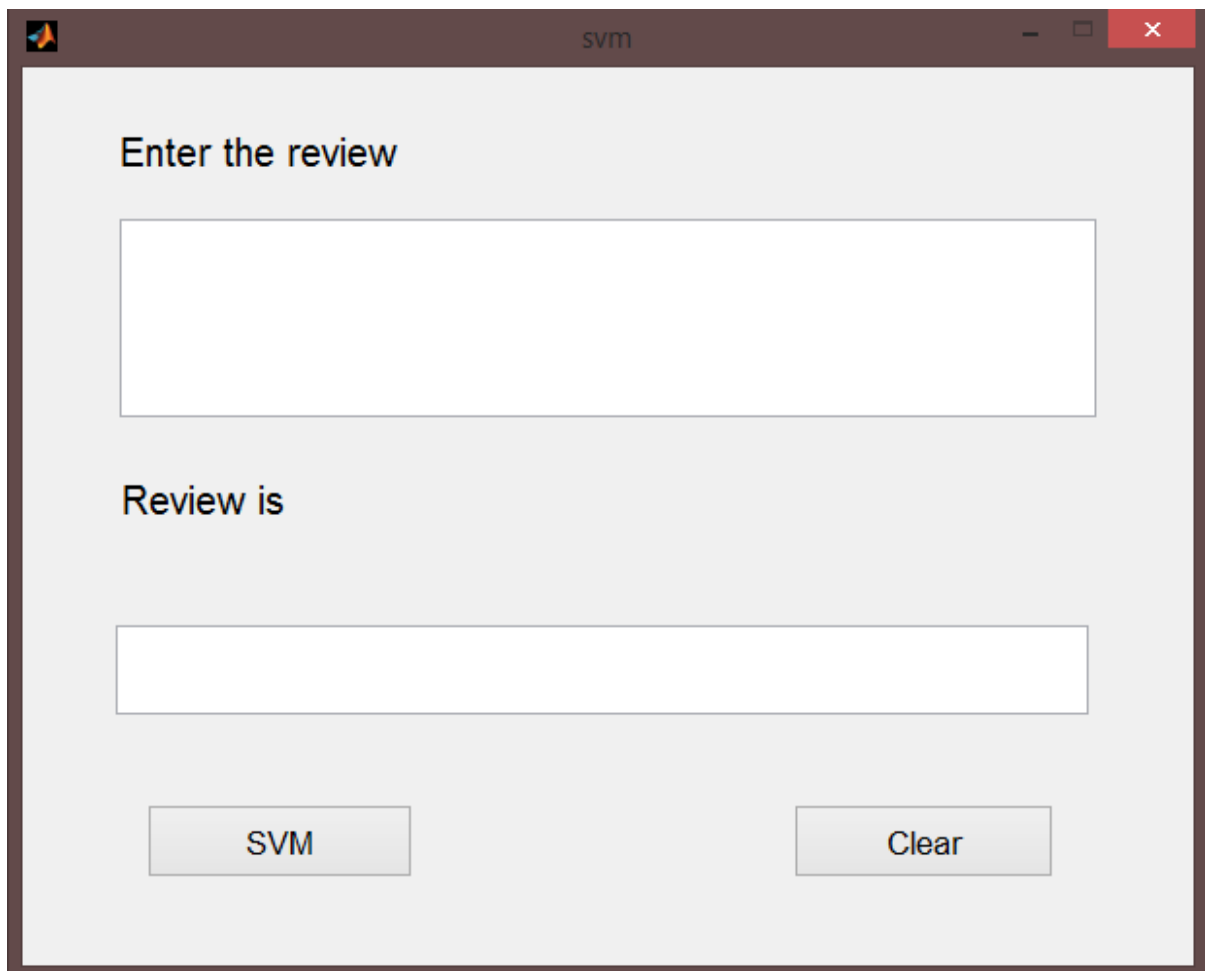


Figure 17 : Bag Of Words screen with output

6.3 Support Vector Machine using Principle Component Analysis



The image shows a software window titled "svm" with a dark brown border. Inside the window, the background is light gray. At the top, the text "Enter the review" is displayed in a black font. Below this text is a large, empty white rectangular text input field. Further down, the text "Review is" is displayed in a black font. Below this text is another empty white rectangular text input field. At the bottom of the window, there are two buttons: "SVM" on the left and "Clear" on the right. Both buttons have a light gray background and a thin black border.

Figure 18 : Support Vector Machine Main Screen for Input

svm

Enter the review

This product has good features.I recommend this to every one.

Review is

SVM Clear

Figure 19 : Support Vector Machine screen with Test Data

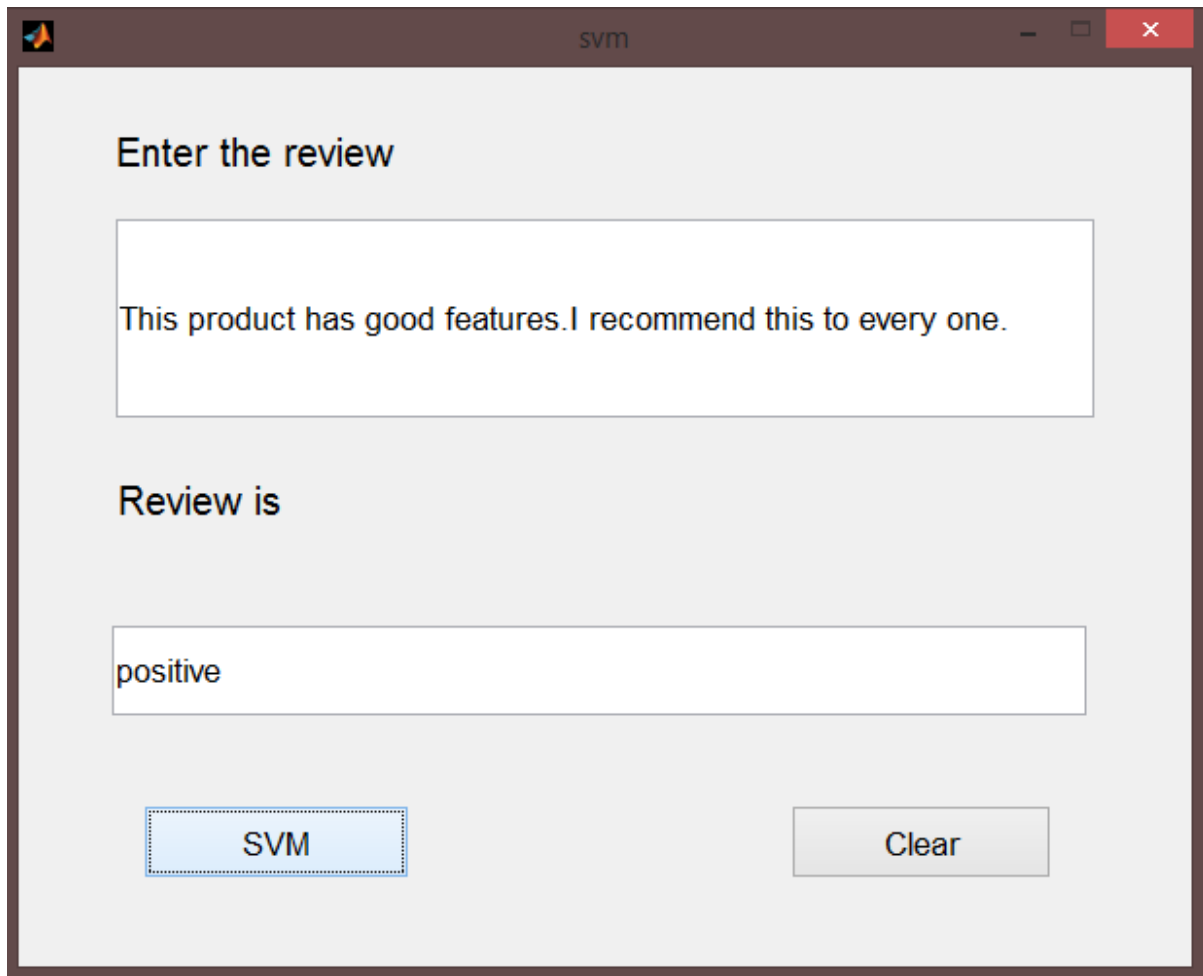


Figure 20 : Support Vector Machine Screen with Output

References

- <https://www.courseera.com>
- <https://www.google.com>
- <https://www.kaggle.com>