RESTAURANT REVIEW CLASSIFICATION AND RECOMMENDER SYSTEM

A project report submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

submitted by

G.S.C. SIRISHA (16A31A0511)

R. SHAMA SUDHEER (16A31A0555)

J. ADITHYA KIRAN (16A31A0540)

Under the Guidance of

A. LAKSHMAN RAO,
ASSOC. PROFESSOR,
M.Tech (CSE)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PRAGATI ENGINEERING COLLEGE
(AUTONOMOUS)

(Approved by AICTE & Permanently Affiliated to JNTUK, Kakinada & Accredited by NAAC)

1-378, ADB Road, Surampalem, E.G.Dist., A.P, Pin-533437.

2018-2019

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PRAGATI ENGINEERING COLLEGE
(AUTONOMOUS)

(Approved by AICTE & Permanently Affiliated to JNTUK & Accredited by NAAC)
1-378, ADB Road, Surampalem, E.G.Dist., A.P, Pin-533437.

CERTIFICATE

This is to certify that the report entitled **"RESTAURANT REVIEW CLASSIFICATION AND RECOMMENDER SYSTEM"** *that is being submitted by* ***G. S .C. Sirisha bearing roll number 16A31A0511***, ***R. Shama Sudheer bearing roll number 16A31A0555, J. Adithya Kiran bearing roll number 16A31A0540 of III Year II Semester*** *in partial fulfillment for the award of the Degree of Bachelor of Technology in Computer Science and Engineering, Pragati Engineering College is a record of bonafide work carried out by them.*

Supervisor	**Head of the Department**
Mr. A. Lakshman Rao	**Dr. M. Radhika Mani,**
Assoc. Professor, M. Tech, Dept of CSE	**Dept of CSE**

ACKNOWLEDGEMENT

It gives us an immense pleasure to express deep sense of gratitude to our guide, Mr. **A. Lakshman Rao**, Associate Professor, Department of Computer Science & Engineering, because of his whole hearted and valuable guidance throughout the report. Without his sustained and sincere effort, this report would not have taken this shape. He encouraged and helped us to overcome various difficulties that we have faced at various stages of our report.

We would like to sincerely thank Prof **Dr. M. Radhika Mani**, Head of the Department, Computer Science & Engineering, for providing all the necessary facilities that led to the successful completion of our report.

We would like to take this opportunity to thank our beloved Principal and Vice- principal **Dr. S. Sambhu Prasad**, **Dr. K. Satyanarayana** for providing a great support for us in completing our project and making us to submit a detailed report of it.

We would like to thank all the faculty members of the Department of Computer Science and Engineering for their direct or indirect support for helping us in completion of this report.

Finally, we would like to thank all of our friends and family members for their continuous help and encouragement.

G. S. C. Sirisha (16A31A0511)

R. Shama Sudheer (16A31A0555)

J. Adithya Kiran (16A31A0540)

ABSTRACT

One of the most effective tools any restaurant has is the ability to track food and beverage sales on a daily basis. A daily business review, as this type of report is often called, allows for a place to build a history of business. Recommender systems (RS) have attracted attention in both academia and industry. Such systems help to manage information overload.

Therefore, in this project, we particularly emphasize user reviews and provide a comprehensive survey of recent attempts to use the valuable information in reviews to solve the rating sparsity issue. The system can capture the multi-faceted nature of a user's opinions from his/her reviews and hence build a fine-grained preference model for the user, which however cannot be obtained from overall ratings. Empirical findings from marketing and consumer behavior studies have also documented the positive influence of product reviews on the decision processes of new users.

We make use of machine learning algorithms for classifying reviews and recommend the best restaurant.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ML	Machine Learning
PY	Python
NLP	Natural Language Processing
CSV	Comma Separated Values
TSV	Tab Separated Values
GUI	Graphical User Interface
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative
RE	Regular Expression
NB	Naive Bayes
NLTK	Natural Language Tool Kit

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION TO NATURAL LANGUAGE PROCESSING (NLP)

Natural language processing (NLP) is the ability of a computer program to understand human language as it is spoken. NLP is a component of artificial intelligence (AI). NLP can be used to interpret free text and make it analyzable. Current approaches to NLP are based on deep learning, a type of AI that examines and uses patterns in data to improve a program's understanding.

1.2 INTRODUCTION TO MACHINE LEARNING

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it learn for themselves.

The process of learning begins with observations or data, such as examples, direct experience, or instruction, in order to look for patterns in data and make better decisions in the future based on the examples that we provide. The primary aim is to allow the computers learn automatically without human intervention or assistance and adjust actions accordingly.

Some machine learning methods

Machine learning algorithms are often categorized as supervised or unsupervised.

Supervised machine learning algorithms can apply what has been learned in the past to new data using labeled examples to predict future events. Starting from the analysis of a known training dataset, the learning algorithm produces an inferred function to make predictions about the output values. The system is able to provide targets for any new input after sufficient training. The learning algorithm can also compare its output with the correct, intended output and find errors in order to modify the model accordingly.

In contrast, **unsupervised machine learning** algorithms are used when the information used to train is neither classified nor labeled. Unsupervised learning studies how systems can infer a function to describe a hidden structure from unlabeled data. The system doesn’t figure out the right output, but it explores the data and can draw inferences from datasets to describe hidden structures from unlabeled data.

Semi-supervised machine learning algorithms fall somewhere in between supervised and unsupervised learning, since they use both labeled and unlabeled data for training – typically a small amount of labeled data and a large amount of unlabeled data. The systems that use this method are able to considerably improve learning accuracy. Usually, semi-supervised learning is chosen when the acquired labeled data requires skilled and relevant resources in order to train it / learn from it. Otherwise, acquiring unlabeled data generally doesn't require additional resources.

Reinforcement machine learning algorithms is a learning method that interacts with its environment by producing actions and discovers errors or rewards. Trial and error search and delayed reward are the most relevant characteristics of reinforcement learning. This method allows machines and software agents to automatically determine the ideal behavior within a specific context in order to maximize its performance. Simple reward feedback is required for the agent to learn which action is best; this is known as the reinforcement signal.

Machine learning enables analysis of massive quantities of data. While it generally delivers faster, more accurate results in order to identify profitable opportunities or dangerous risks, it may also require additional time and resources to train it properly. Combining machine learning with AI and cognitive technologies can make it even more effective in processing large volumes of information.

1.3 INTRODUCTION TO CLASSIFICATION IN MACHINE LEARNING

In machine learning and statistics, **Classification** is a supervised learning approach in which the computer program learns from the data input given to it and then uses this learning to classify new observation. This data set may simply be bi-class (like identifying whether the person is male or female or that the mail is spam or non-spam) or it may be multi-class too. Some examples of classification problems are: speech recognition, handwriting recognition, bio metric identification, document classification etc.

Here we have the types of classification algorithms in Machine Learning:

1.Linear Classifiers: Logistic Regression, Naive Bayes Classifier

2.Support Vector Machines

3.Decision Trees

4.Random Forest

5.Neural Networks

6.Nearest Neighbor

1.3.1 Naive Bayes Classifier (Generative Learning Model)

It is a classification technique based on Bayes' Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. Even if these features depend on each other or upon the existence of the other features, all of these properties independently contribute to the probability. Naive Bayes model is easy to build and particularly useful for very large data sets. Along with simplicity, Naive Bayes is known to outperform even highly sophisticated classification method

1.3.2 Logistic Regression (Predictive Learning Model)

It is a statistical method for analyzing a data set in which there are one or more independent variables that determine an outcome. The outcome is measured with a dichotomous variable (in which there are only two possible outcomes). The goal of logistic regression is to find the best fitting model to describe the relationship between the dichotomous characteristic of interest (dependent variable = response or outcome variable) and a set of independent (predictor or explanatory) variables.

1.3.3 Decision Trees

Decision tree builds classification or regression models in the form of a tree structure. It breaks down a data set into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes. A decision node has two or more branches and a leaf node represents a classification or decision. The topmost decision node in a tree which corresponds to the best predictor called root node. Decision trees can handle both categorical and numerical data.

1.3.4 Random Forest

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of over fitting to their training set.

1.3.5 Neural Network

A neural network consists of units (neurons), arranged in layers, which convert an input vector into some output. Each unit takes an input, applies a (often nonlinear) function to it and then passes the output on to the next layer. Generally, the networks are defined to be feed-forward: a unit feeds its output to all the units on the next layer, but there is no feedback to the previous layer. Weightings are applied to the signals passing from one unit to another, and it is these weightings which are tuned in the training phase to adapt a neural network to the particular problem at hand.

1.3.6 Nearest Neighbor

The k-nearest-neighbors algorithm is a classification algorithm, and it is supervised: it takes a bunch of labelled points and uses them to learn how to label other points. To label a new point, it looks at the labelled points closest to that new point (those are its nearest neighbors), and has those neighbors vote, so whichever label the most of the neighbors have is the label for the new point (the "k" is the number of neighbors it checks).

1.4 OBJECTIVE OF THE PROJECT

The main objective of our project is to classify reviews and recommend the best restaurant for user convenience.

1.4.1 Existing System

Every business process is on paper. Every business information is generated by employee. We may use google or other search engines to know the details of the particular restaurant. We can also get the information by contacting our friends and family members who already visited the restaurant.

1.4.2 Proposed System

We make use of machine learning algorithms for classifying reviews and recommend the best restaurant. This project serves the best way of maintaining the customer's information and cater their needs. From the classification machine learning algorithms mentioned above, we make use of 'NAÏVE BAYES CLASSIFIER (GENERATIVE LEARNING ALGORITHM)'. The machine learning algorithm is supervised machine learning algorithm and 'MULTINOMIAL NAÏVE BAYES CLASSIFIER' is used. The algorithm we used is one of the 'COLLABORATIVE FILTERING MODEL-BASED ALGORITHMS'.

CHAPTER 2
LITERATURE SURVEY

In general, the methods implemented in a recommender system fall within one of the following categories:

(a) Content-based Methods
(b) Collaborative Methods and
(c) Hybrid Methods

2.1 CONTENT-BASED METHODS

Modern information systems embed the ability to monitor and analyze users' actions to determine the best way to interact with them. Ideally, each user's actions are logged separately and analyzed to generate an individual user profile. All the information about a user is extracted either by monitoring his/her actions or by examining the objects the user has evaluated, is stored and utilized to customize services offered. This user modeling approach is known as **content-based learning**. The main assumption behind it is that a user's behavior remains unchanged through time; therefore, the content of past user actions may be used to predict the desired content of their future actions. Therefore, in content-based recommendation methods, the rating R(u,i) of the item i for the user u is typically estimated based on ratings assigned by user u to the items $I_n \in I$ that are "similar" to item i in terms of their content, as defined by their associated features. To be able to search through a collection of items and make observations about the similarity between objects that are not directly comparable, we must transform raw data at a certain level of information granularity. Information granules refer to a collection of data that contain only essential information. Such granulation allows more efficient processing for extracting features and computing numerical representations that characterize an item. As a result, the large amount of detailed information of one item is reduced to a limited set of features. Each feature is a vector of low dimensionality, which captures some aspects of the item and can be used to determine item similarity. Therefore, an item i could be described by a feature vector

$F(i) = [\,feature_1(i), feature_2(i), feature_3(i), \ldots feature_n(i)]$.

2.2 COLLABORATIVE METHODS

Collaborative filtering (CF) methods are based on the assumption that similar users prefer similar items or that a user expresses similar preferences for similar items. Instead of performing content indexing or content analysis, CF systems rely entirely on interest ratings from the members of a participating community. CF methods are categorized into two general classes, namely **model-based and memory-based**. **Model-based** algorithms use the underlying data to learn a probabilistic model, such as a cluster model or a Bayesian network model using statistical and machine learning techniques. Subsequently, they use the model to make predictions. The clustering model works by clustering similar users in the same class and estimating the probability that a particular user is in a particular class. From there, the clustering model computes the conditional probability of ratings. **Memory-based** methods, store raw preference information in computer memory and access it as needed to find similar users or items and to make predictions. CF was formulated as a classification problem. Specifically, based on a set of user ratings about items, they try to induce a model for each user that would allow the classification of unseen items into two or more classes, each of which corresponds to different points in the accepted rating scale. Memory-based CF methods can be further divided into two groups, namely **user-based** and **item-based** methods. On the one hand, **user-based** methods look for users (also called "neighbors") similar to the active user and calculate a predicted rating as a weighted average of the neighbor's ratings on the desired item. On the other hand, **item-based** methods look for similar items for an active user.

2.2.1 User-Based Collaborative Filtering Systems

User-based CF systems are systems that utilize memory-based algorithms, meaning that they operate over the entire user-item matrix R, to make predictions. The majority of such systems mainly deal with user-user similarity calculations, meaning that they utilize user neighborhoods, constructed as collections of similar users. In other words, they deal with the rows of the user-item matrix, R, in order to generate their results. For example, in a personalized music RS called RINGO [, similarities between the tastes of different users are utilized to recommend music items. This user-based CF approach works as follows: A new user is matched against the database to discover neighbors, which are other customers who, in the past, have had a similar taste as the new user, i.e. who have bought similar items as the new user. Items (unknown to the new user) that these neighbors like are then recommended to the new user.

2.2.2 Item-Based Collaborative Filtering Systems

A different approach is based on item relations and not on user relations, as in classic CF. Since the relationships between users are relatively dynamic, as they continuously buy new products, it is computationally hard to calculate the user-to-user matrix online. This causes the user-based CF approach to be relatively expensive in terms of computational load. In the item-based CF algorithm, we look into the set of items, denoted by I_{ua} , that the active user, u_a, has rated and compute how similar they are to the target item i_t . Then, we select the k most similar items $I_k = \{i_1,i_2,...,i_k\}$, based on their corresponding similarities $\{sim(i_t,i_1),sim(i_t,i_2), . . . ,sim(i_t,i_k)\}$. The predictions can then be computed by taking a weighted average of the active user's ratings on these similar items. The main steps in this approach are the same as in user-based CF. The difference in the present approach is that instead of calculating similarities between two users who have provided ratings for a common item, we calculate similarities between two items i_t, i_j which have been rated by a common user u_a.

2.2.3 Personality Diagnosis

Personality diagnosis may be thought of as a **hybrid between memory and model-based** approaches of CF. The main characteristic is that predictions have meaningful probabilistic semantics. Moreover, this approach assumes that preferences constitute a characterization of their underlying personality type for each user. Therefore, taking into consideration the active user's known ratings of items, it is possible to estimate the probability that he/she has the same personality type with another user. The personality type of a given user is taken to be the vector of "true" ratings for items the user has seen. A true rating differs from the actually reported rating given by a user by an amount of (Gaussian) noise. Given the personality type of a user, the personality diagnosis approach estimates the probability that the given user is of the same personality type as other users in the system, and, consequently, estimates the probability that the user will like some new item.

2.3 HYBRID METHODS

Hybrid methods combine two or more recommendation techniques to achieve better performance and to take out drawbacks of each technique separately. Usually, CF methods are combined with content-based methods. Hybrid recommender systems could be classified into the following categories:
• Combining Separate Recommenders
• Adding Content-Based Characteristics to Collaborative Models
• Adding Collaborative Characteristics to Content-Based Models
• A Single Unifying Recommendation Model

2.3.1 Combining Separate Recommenders

The Hybrid recommender system of this category include two separate systems, a collaborative one and a content-based one. There are four different ways of combining these two separate systems, namely the following:
• Weighted Hybridization Method: The outputs (ratings) acquired by individual RS are combined together to produce a single final recommendation using either a linear combination or a voting scheme. The P-Tango system initially gives equal weights to both recommenders, but gradually adjusts the weights as predictions about user ratings are confirmed or not. The system keeps the two filtering approaches separate and this allows the benefit from individual advantages.
• Switched Hybridization Method: The system switches between recommendation techniques selecting the method that gives better recommendations for the current situation depending on some recommendation "quality" metric. A characteristic example of such a recommender is The Daily Learner which selects the recommender sub-system that provides the higher level of confidence. Another example of this method is presented in where either the content-based or the collaborative filtering technique is selected according to which of the two provided better consistency with past ratings of the user.

• **Mixed Hybridization Method:** In this method, the results from different recommender sub-systems are presented simultaneously. An example of such a recommender is given in where they utilize a content-based technique based on textual descriptions of TV shows and collaborative information about users' preferences. Recommendations from both techniques are provided together in the final suggested program.

• **Cascade Hybridization Method:** In this method, one recommendation technique is utilized to produce a coarse ranking of candidates, while the second technique focuses only on those items for which additional refinement is needed. This method is more efficient than the weighted hybridization method which applies all of its techniques on all items. The computational burden of this hybrid approach is relatively small because recommendation candidates in the second level are partially eliminated in the first level. Moreover, this method is more tolerant to noise in the operation of low-priority recommendations, since ratings of the high-level recommender can only be refined, but never over-turned. In other words, cascade hybridization methods can be analyzed into two sequential stages. The first stage (content-based method or knowledge-based/collaborative) selects intermediate recommendations. Then, the second stage (collaborative/content-based method or knowledge-based) selects appropriate items from the recommendations of the first stage. Burke developed a restaurant RS called EntreeC. The system first selects several restaurants that match a user's preferred cuisine (e.g., Italian, Chinese, etc.) with a knowledge-based method. In the knowledge-based method, the authors construct a feature vector according to defined attributes that characterize the restaurants. This method is similar to content-based methods; however, it must be noted that these metadata are content-independent and for this reason the term knowledge-based is utilized. These restaurants are then ranked with a collaborative method.

2.3.2 Adding Content-Based Characteristics to Collaborative Models

Collaboration via content is a method that uses a prediction scheme similar to the standard CF, in which similarity among users is not computed on provided ratings, but rather on the content-based profile of each user. The underlying intuition is that like-minded users are likely to have similar content-based models and that this similarity relation can be detected without requiring overlapping ratings. The main limitation of this approach is that the similarity of users is computed using Pearson's correlation coefficient between content-based weight vectors. On the other hand, the authors proposed the content-boosted collaborative filtering approach, which exploits a content-based predictor to enhance existing user data and then provides personalized suggestions through CF. The content-based predictor is applied to each row of the initial user-item matrix, corresponding to each user, and gradually generates a pseudo user-item matrix that is a full dense matrix. The similarity between the active user, u_a, and another user u_i , is computed with CF using the new pseudo user-item matrix.

2.3.3 Adding Collaborative Characteristics to Content-Based Models

The main technique of this category is to apply dimensionality reduction on a group of content-based profiles. The authors used latent semantic indexing to create a collaborative view of a collection of user profiles represented as term vectors. This technique results in performance improvement in comparison with the pure content-based approach.

2.3.4 A Single Unifying Recommendation Model

A general unifying model that incorporates content-based and collaborative characteristics was proposed where the authors present the use of content-based and collaborative characteristics (e.g., the age or gender of users or the genre of movies) in a single rule-based classifier. Single unifying models were also presented where the authors utilized a unified probabilistic method for combining collaborative and content-based recommendations.

CHAPTER 3

METHODOLOGY

3.1 ARCHITECTURE OF THE SYSTEM

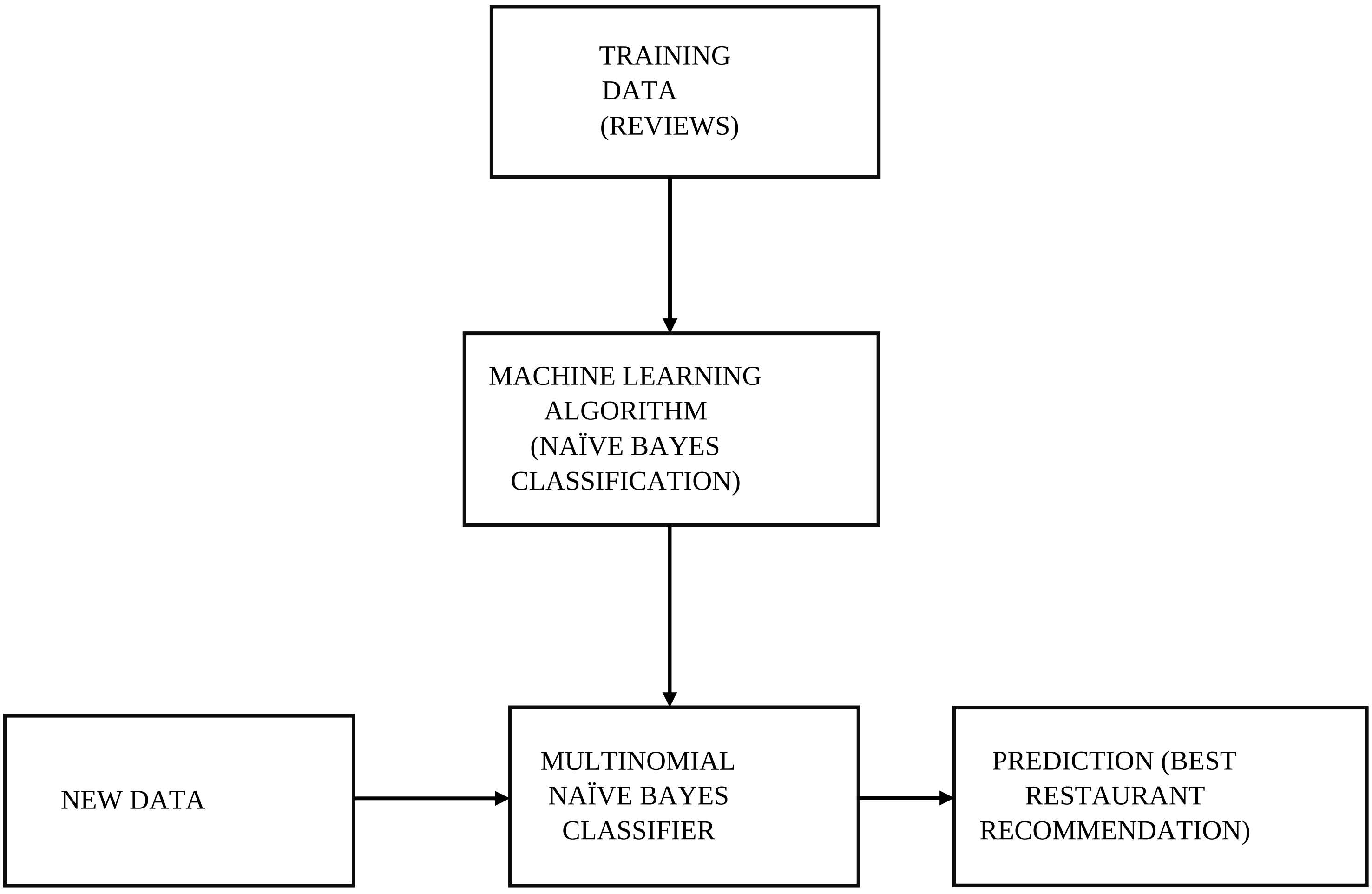

TRAINING DATA (REVIEWS)
MACHINE LEARNING ALGORITHM (NAÏVE BAYES CLASSIFICATION)
NEW DATA
MULTINOMIAL NAÏVE BAYES CLASSIFIER
PREDICTION (BEST RESTAURANT RECOMMENDATION)

Fig 3.1: Architecture of the System

3.2 FLOW CHART OF THE SYSTEM

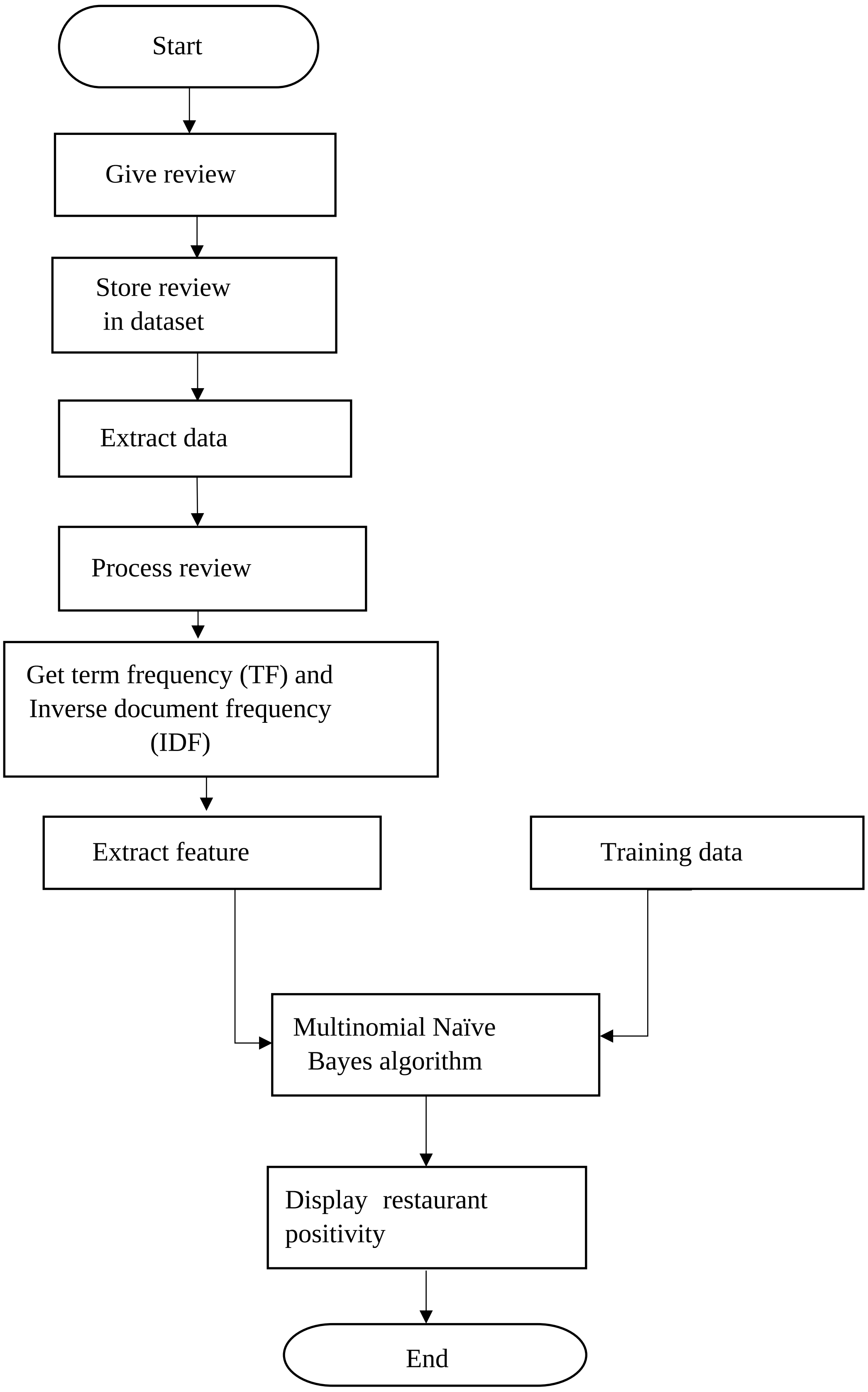

Start
Give review
Store review in dataset
Extract data
Process review
Get term frequency (TF) and Inverse document frequency (IDF)
Extract feature
Training data
Multinomial Naïve Bayes algorithm
Display restaurant positivity
End

Fig 3.2: Flow Chart of the System

3.3 UML DESIGN DIAGRAMS OF THE SYSTEM

3.3.1 CLASS DIAGRAM

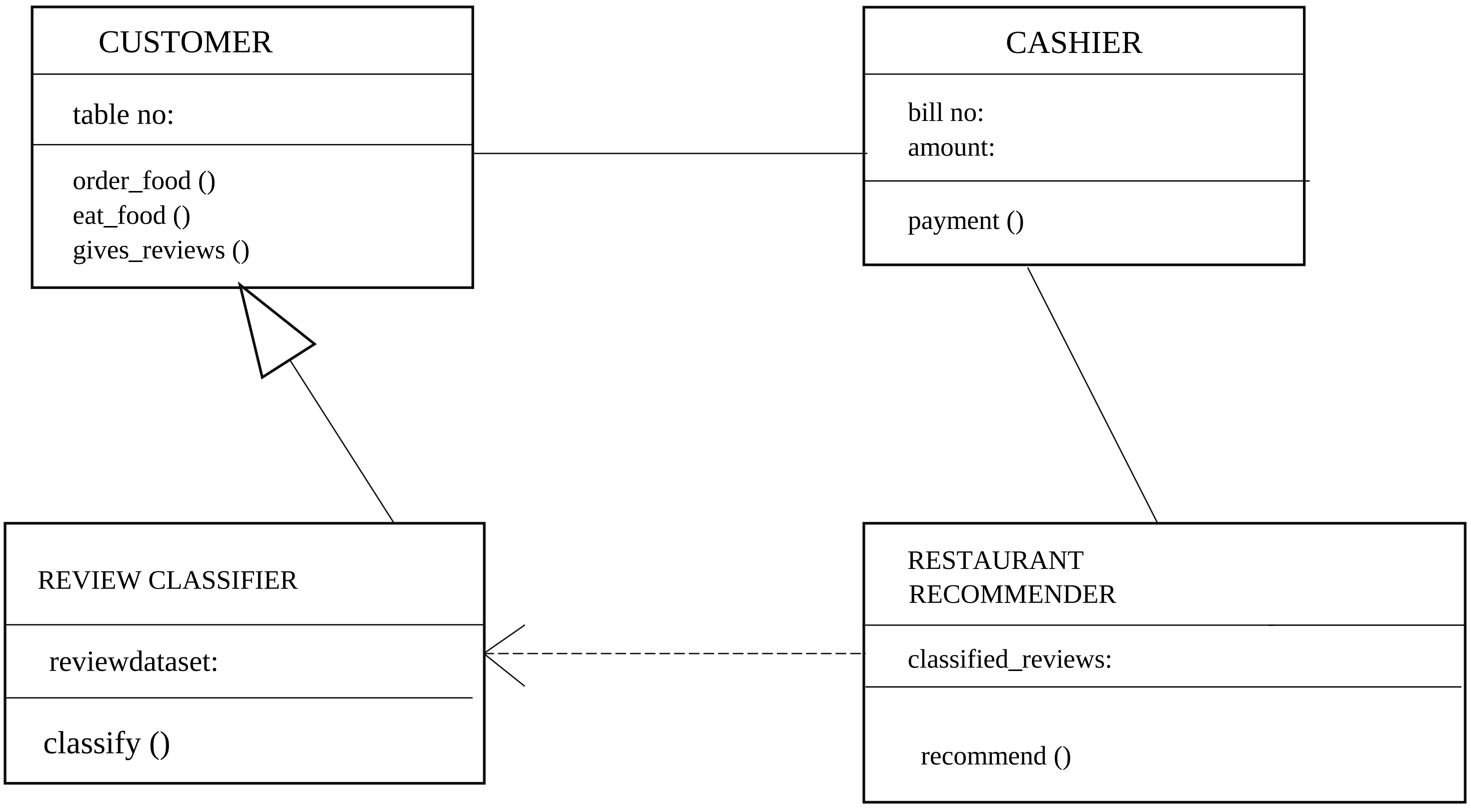


Fig 3.3: Class Diagram

3.3.2 OBJECT DIAGRAM

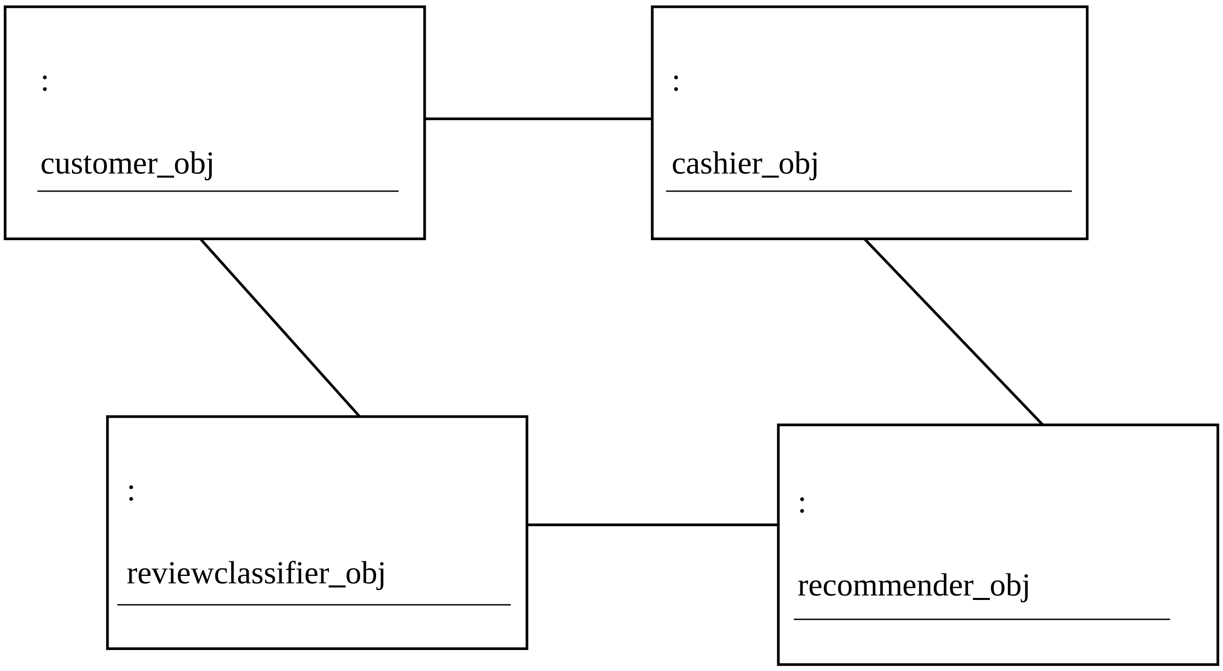


Fig 3.4: Object Diagram

3.3.3 ACTIVITY DIAGRAM

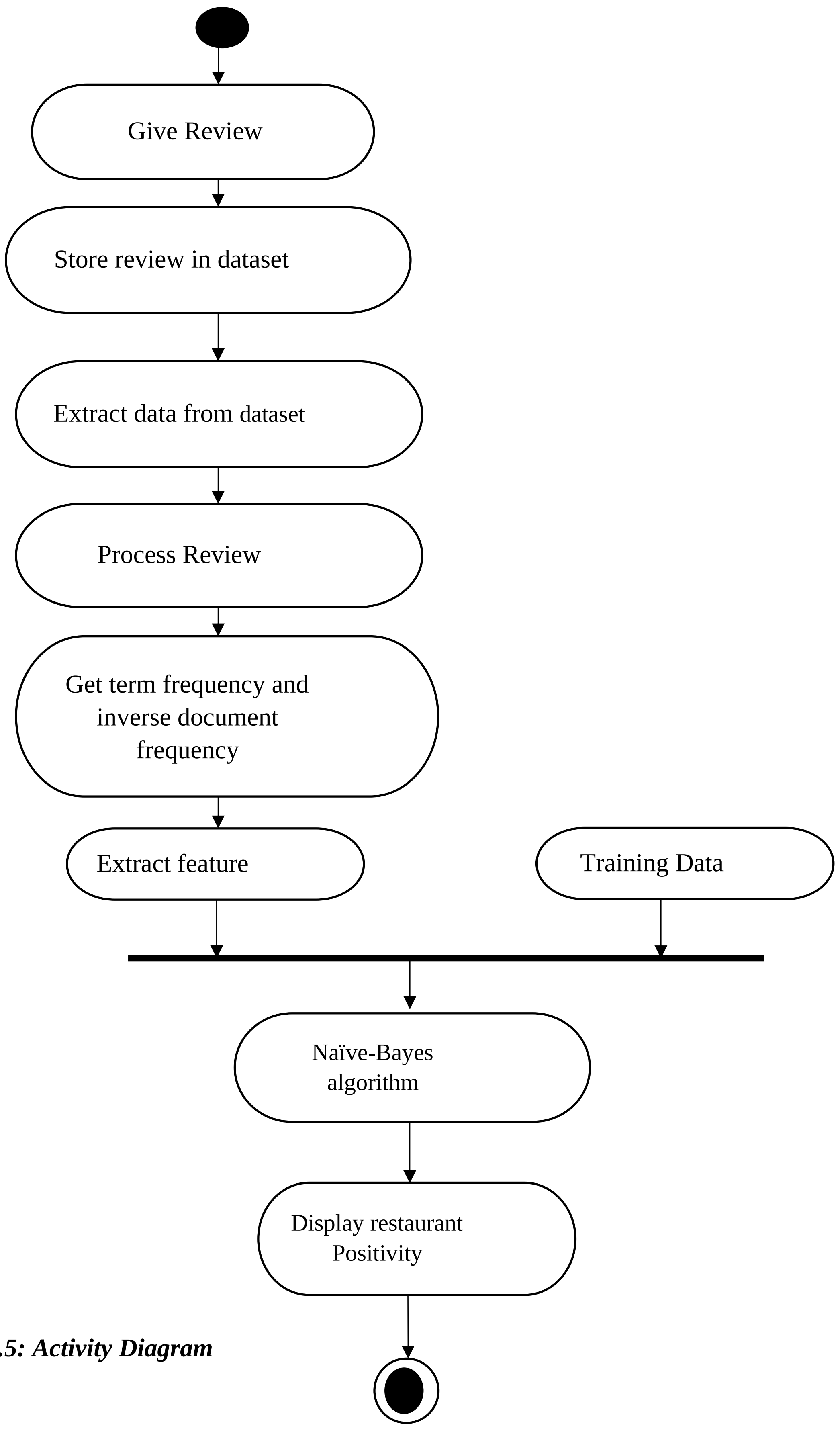
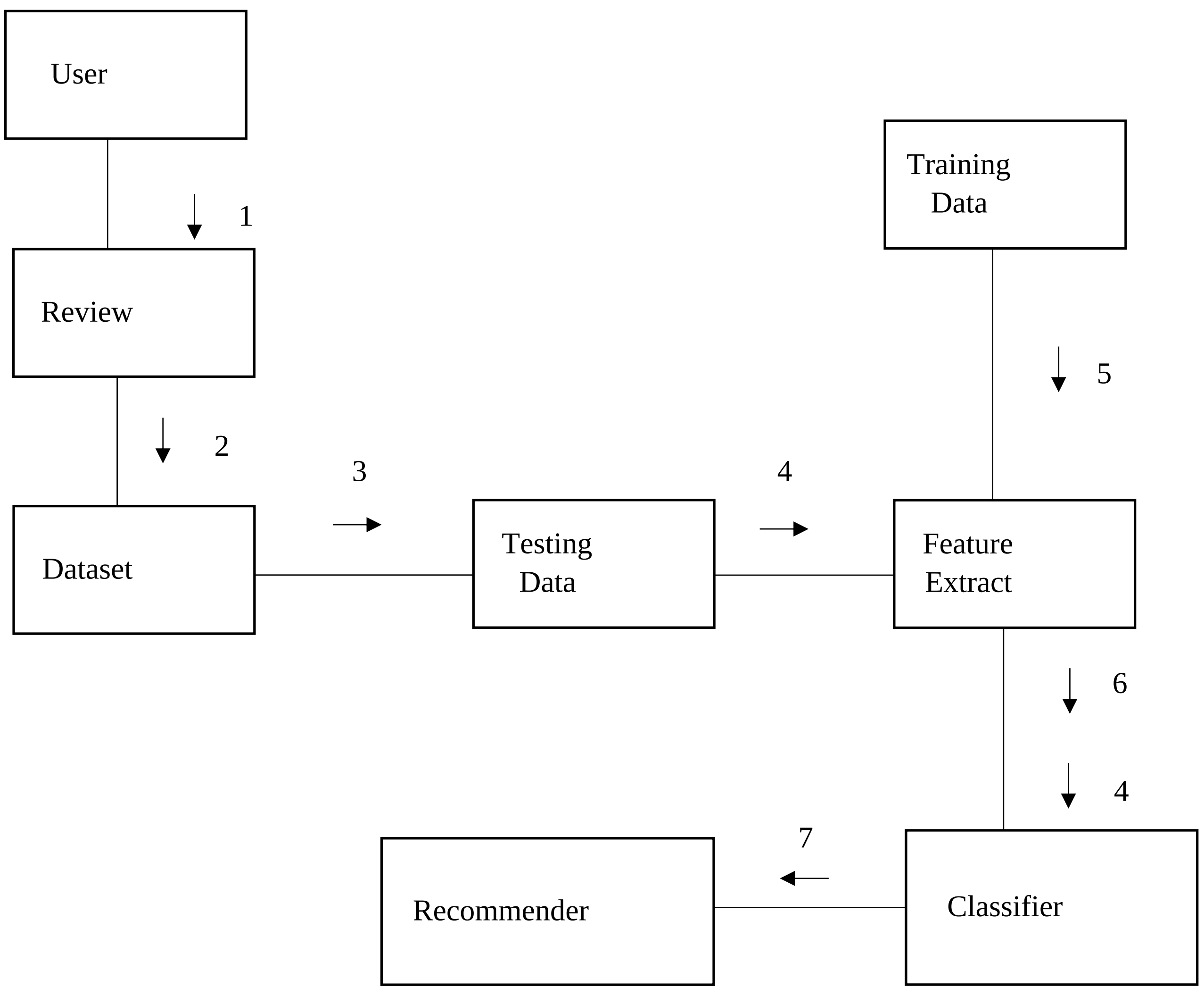

Give Review
Store review in dataset
Extract data from dataset
Process Review
Get term frequency and inverse document frequency
Extract feature
Training Data
Naïve-Bayes algorithm
Display restaurant Positivity

Fig 3.5: Activity Diagram

3.3.4 COLLABORATION DIAGRAM



1: take review
2: store review in dataset
3: extract data from dataset
4: send new review
5: send reviews with result
6: send reviews to classifier
7: send positives to recommender

Fig 3.6: Collaboration Diagram

3.3.5 STATE CHART DIAGRAM

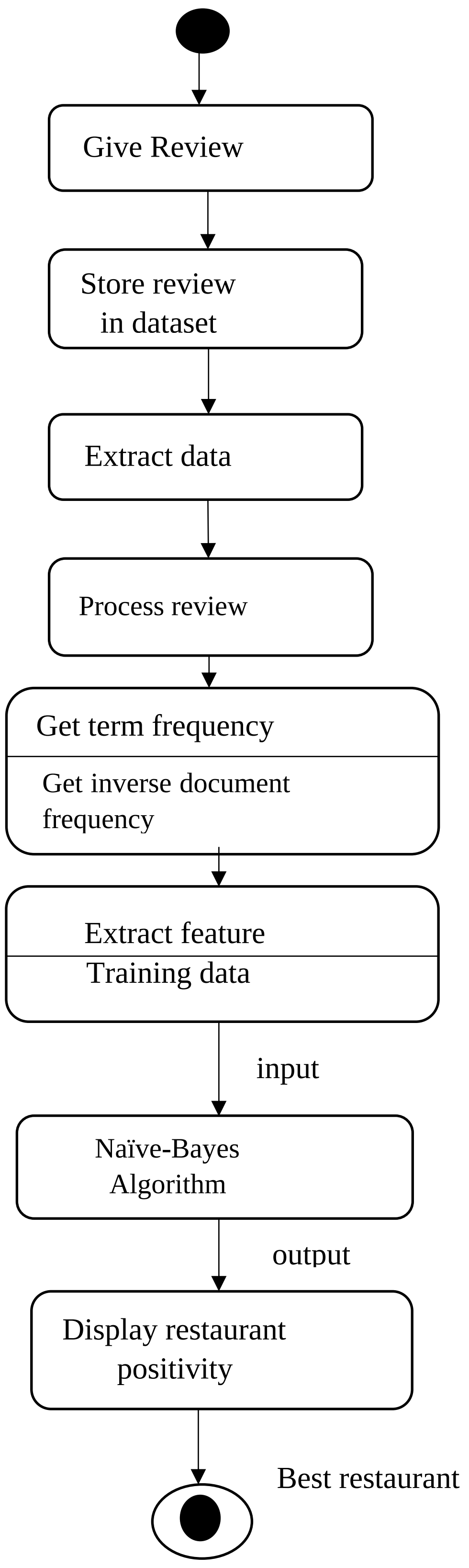
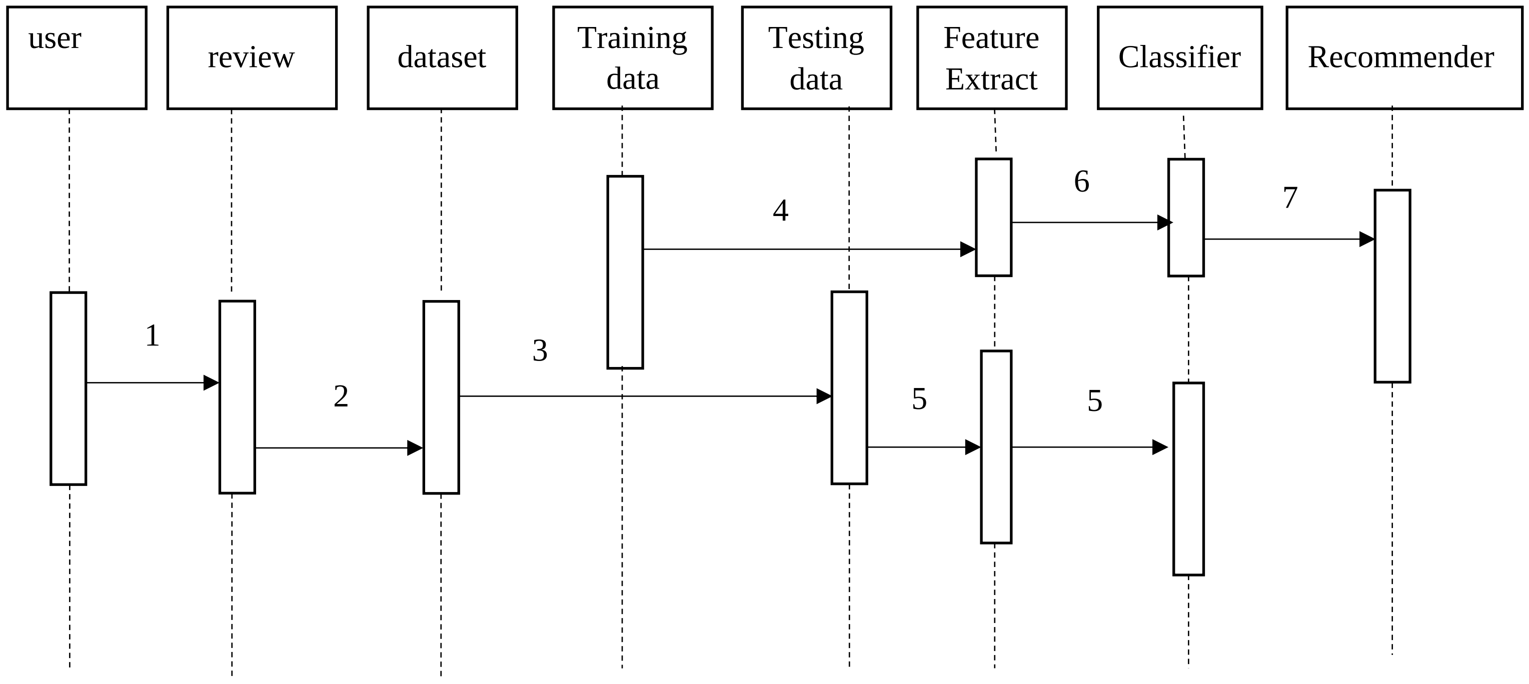


Fig 3.7: State Chart Diagram

3.3.6 SEQUENCE DIAGRAM



1: take
2: store review in dataset
3: extract data from dataset
4: send reviews with result
5: send new review
6: send reviews to classifier
7: send positives to recommender

Fig 3.8: Sequence diagram

3.3.7 DEPLOYMENT DIAGRAM

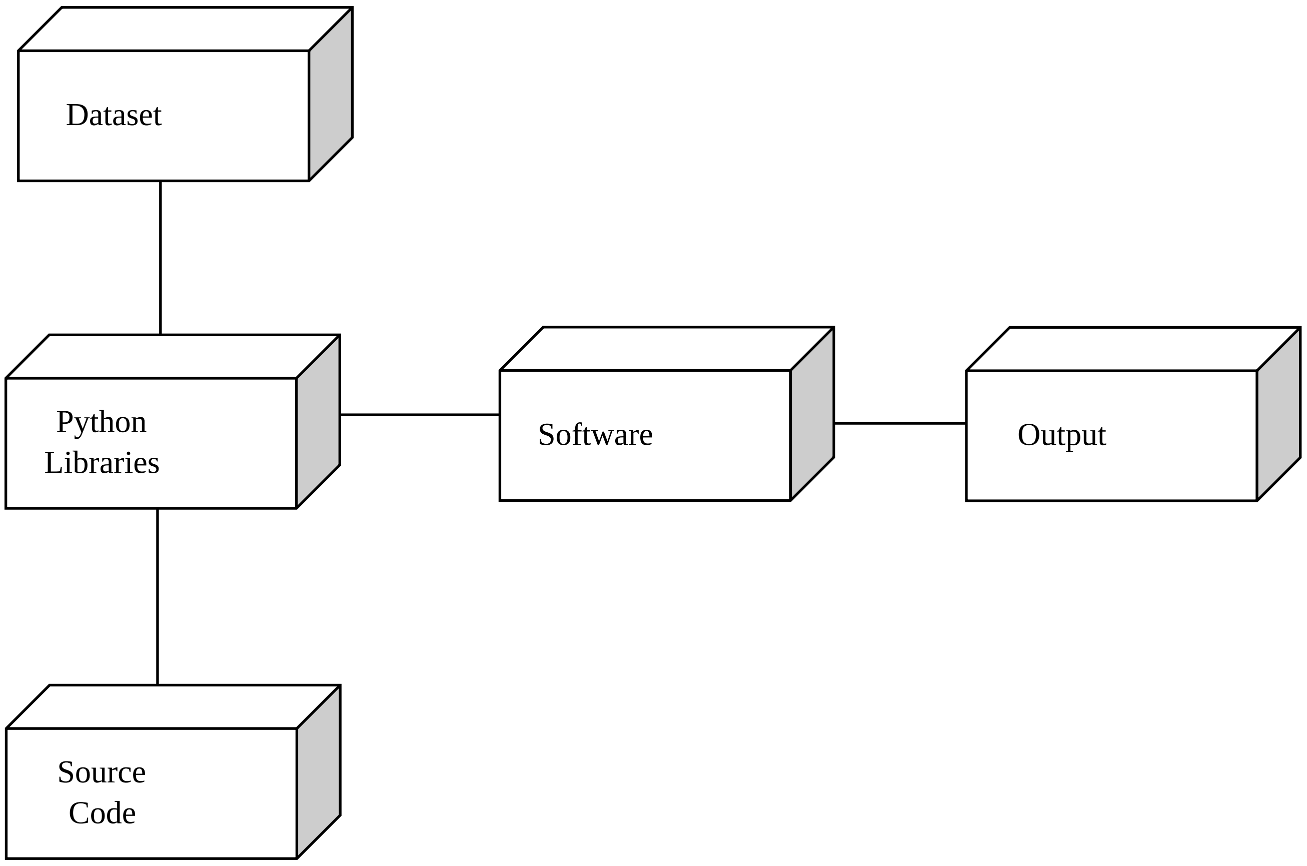


Fig 3.9: Deployment diagram

3.3.8 COMPONENT DIAGRAM

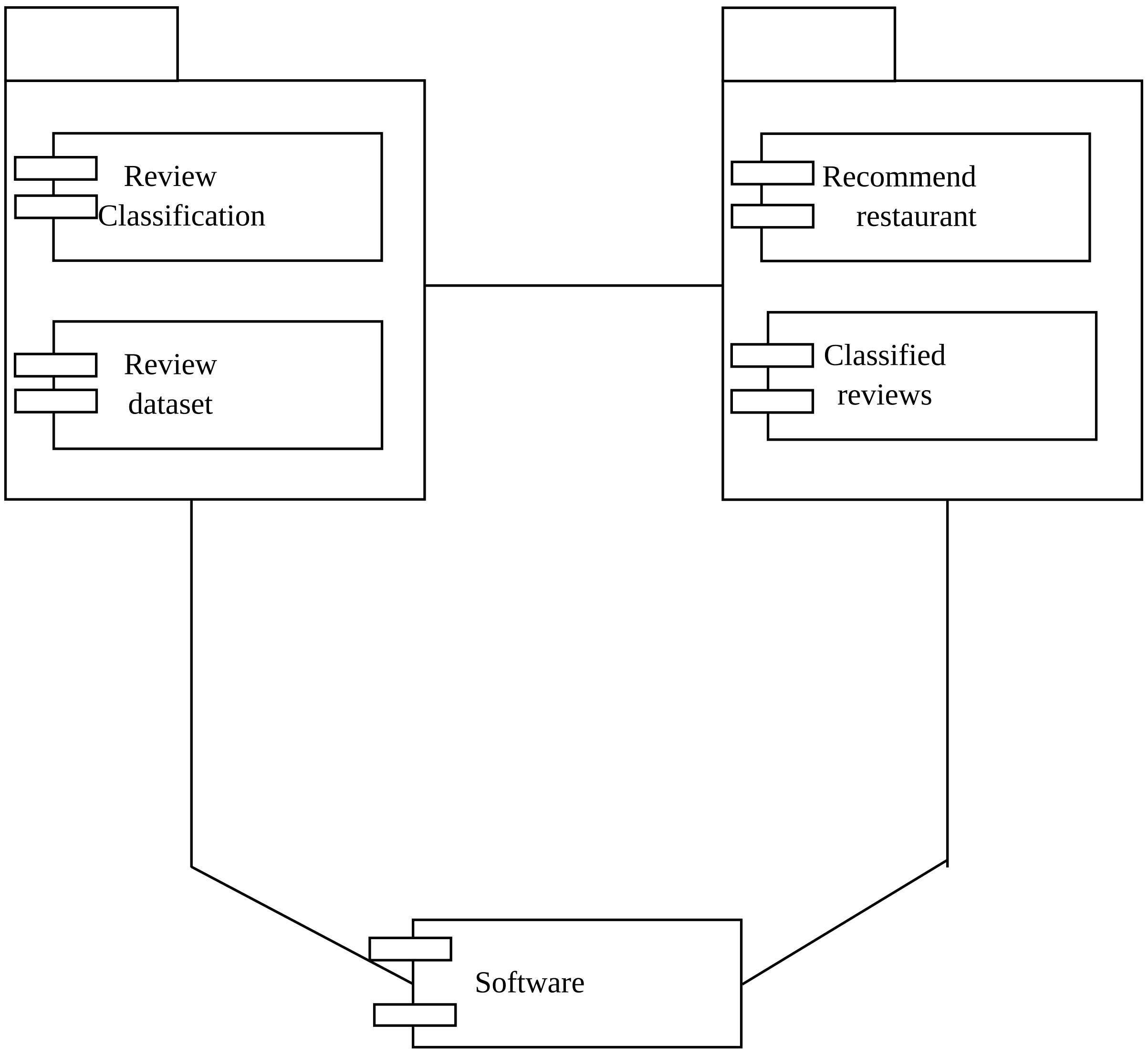


Fig 3.10: Component diagram

3.3.9 USE CASE DIAGRAM

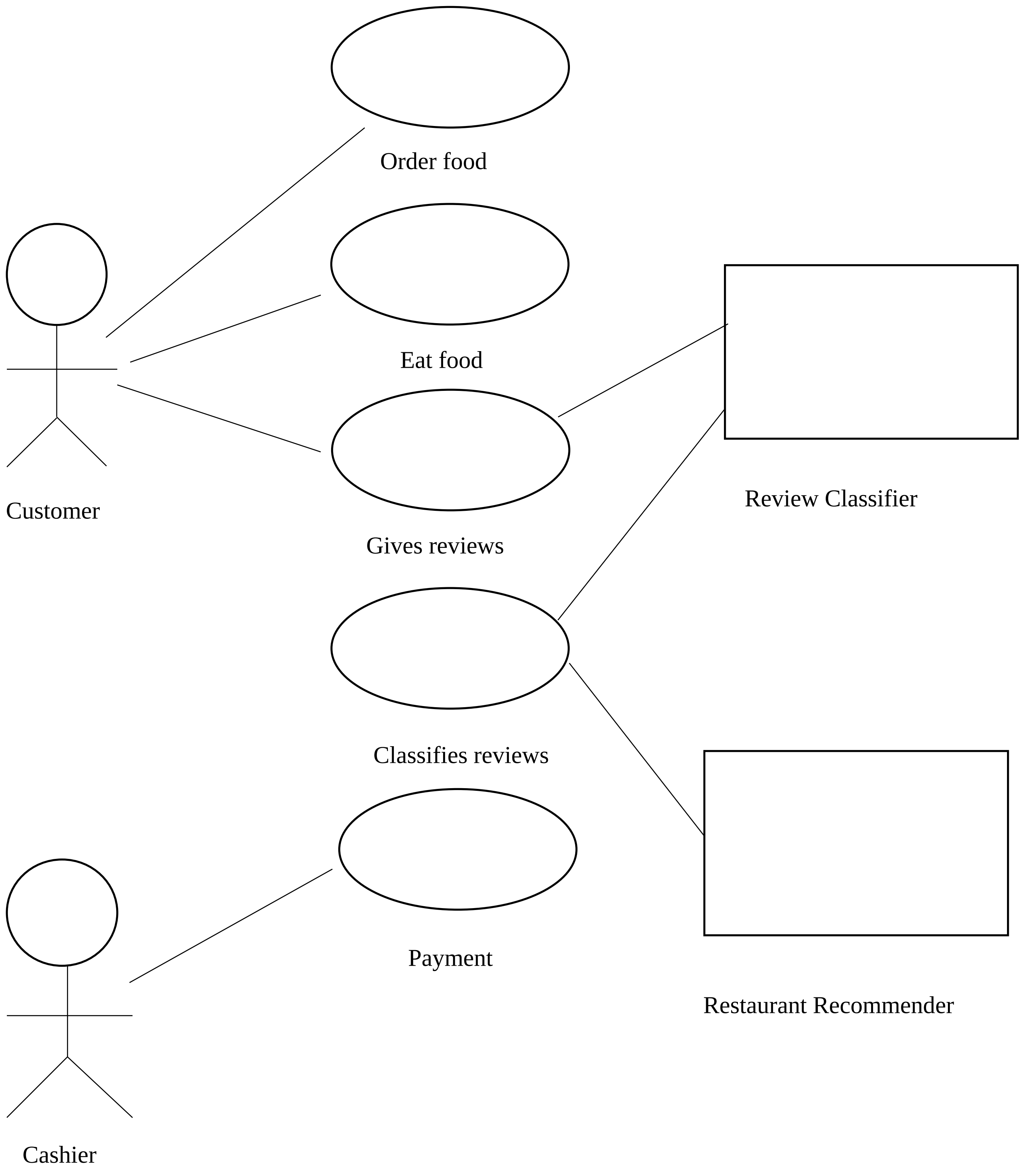


Fig 3.11: Use Case Diagram

3.4 ALGORITHM

3.4.1 INTRODUCTION

The algorithm used for the system is MULTINOMIAL NAÏVE BAYES algorithm. Feature vectors represent the frequencies with which certain events have been generated by a multinomial distribution. This is the event model typically used for document classification**.**

3.4.2 NAÏVE BAYES CLASSIFIER

Naive Bayes is a simple, yet effective and commonly-used, machine learning classifier. It is a probabilistic classifier that makes classifications using the Maximum A Posteriori decision rule in a Bayesian setting. It can also be represented using a very simple Bayesian network. Naive Bayes classifiers have been especially popular for text classification, and are a traditional solution for problems such as spam detection.

Assumption

The fundamental Naive Bayes assumption is that **each feature makes an independent and equal contribution to the outcome.**

We assume that no pair of features are dependent. Secondly, each feature is given the same weight (or importance). None of the attributes is irrelevant and assumed to be contributing equally to the outcome.
Note: The assumptions made by Naive Bayes are not generally correct in real-world situations. In-fact, the independence assumption is never correct but often works well in practice.

Bayes' Theorem

Bayes' Theorem finds the probability of an event occurring given the probability of another event that has already occurred. Bayes' theorem is stated mathematically as the following equation:

$P(A|B) = \{P(B|A)\ P(A)\}/\{P(B)\}$

where A and B are events and $P(B)\neq0$

Basically, we are trying to find probability of event A, given the event B is true. Event B is also termed as evidence.

P(A) is the **priori** of A (the prior probability, i.e. Probability of event before evidence is seen). The evidence is an attribute value of an unknown instance (here, it is event B).

P(A|B) is a **posteriori** probability of B, i.e. probability of event after evidence is seen.

3.4.3 TECHNIQUES FOR IMPROVING NAÏVE BAYES CLASSIFIER

Here are some advanced techniques for improving the basic Naive Bayes classifier:

a. Removing stop words

There are common words that don’t really add anything to the categorization, such as a, able, either, else, ever and so on. So, for our purposes, the election was over would be *election over* and a very close game would be *very close game.*

b. Lemmatizing words

This is grouping together different inflections of the same word. So, election, elected, and so on would be grouped together and counted as appearances of the same word.

c. Using n-grams

Instead of counting single words like we did here, we could together count sequences of words like “*clean match”* and *“close election”.*

d. Using TF-IDF (Term Frequency -Inverse Document Frequency)

Instead of just counting frequency, we could do something more advanced like also penalizing words that appear frequently in most of the samples.

CHAPTER 4
RESULTS AND DISCUSSIONS
4.1 INPUT DATASET

The input dataset required for our project 'RESTAURANT REVIEW CLASSIFICATION AND RECOMMENDER SYSTEM' should consist of customer **review** and **liked/disliked** in the form of 1/0 i.e., 0 for disliked review and 1 for liked review. It should be in '.tsv' format.

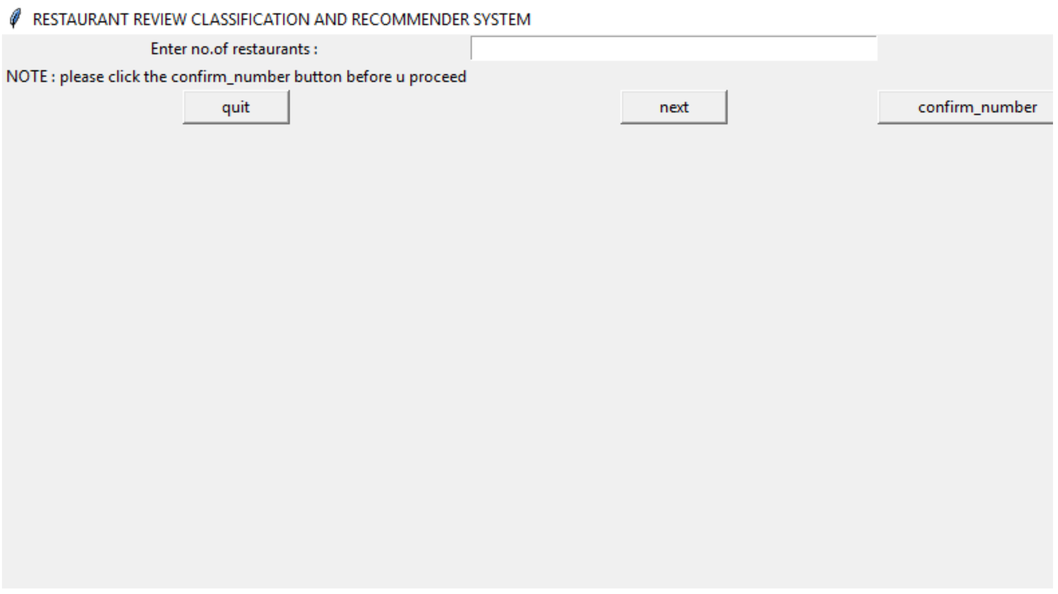
An example input dataset is shown below:

REVIEW	**LIKED (1)/DISLIKED (0)**
Taste is not good.	0
Wow... Loved this place.	1
Not tasty and the texture was just nasty.	0
Stopped by during the late May bank holiday off Rick Steve recommendation and loved it.	1
The selection on the menu was great and so were the prices.	1
Now I am getting angry and I want my damn pho.	0
The potatoes were like rubber and you could tell they had been made up ahead of time being kept under a warmer.	0

Table 4.1: Sample dataset

4.2 RESULT

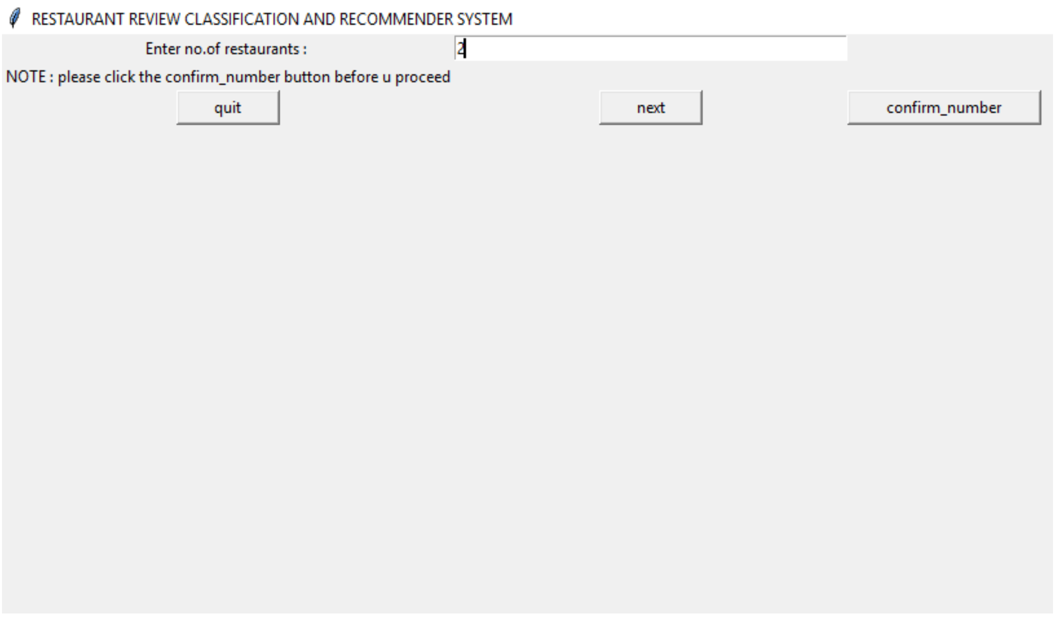
Step 1: Run the code (shift + enter) and the below dialog box 'RESTAURANT REVIEW CLASSIFICATION AND RECOMMENDER SYSTEM' appears:


RESTAURANT REVIEW CLASSIFICATION AND RECOMMENDER SYSTEM

Enter no.of restaurants :

NOTE : please click the confirm_number button before u proceed

Step 2: Enter number of restaurants as below:

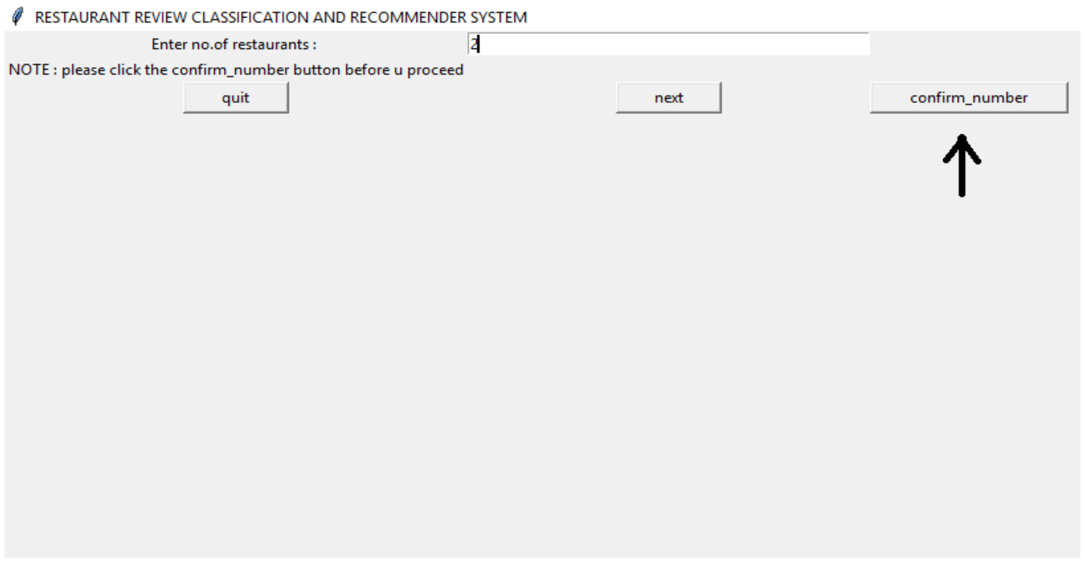

RESTAURANT REVIEW CLASSIFICATION AND RECOMMENDER SYSTEM

Enter no.of restaurants : 2

NOTE : please click the confirm_number button before u proceed

Note 1: At any stage before clicking 'next', one can click on 'quit' and stop the process.

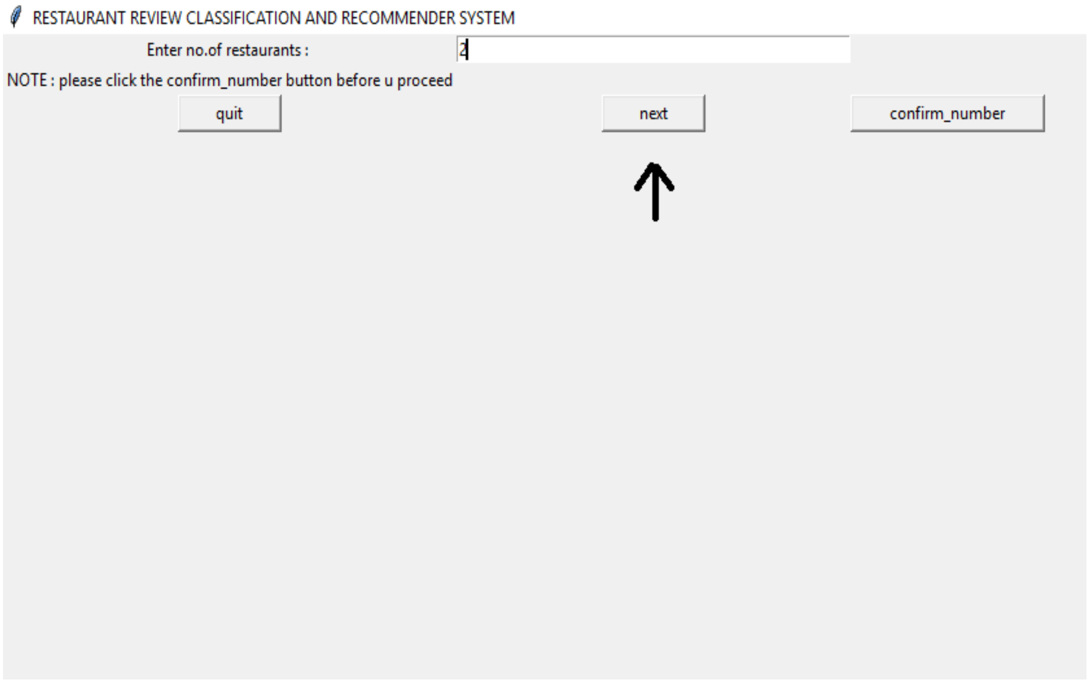
Step 3: Observe the NOTE in the above window and hence click the 'confirm_number' button first


RESTAURANT REVIEW CLASSIFICATION AND RECOMMENDER SYSTEM

Enter no.of restaurants : 2

NOTE : please click the confirm_number button before u proceed

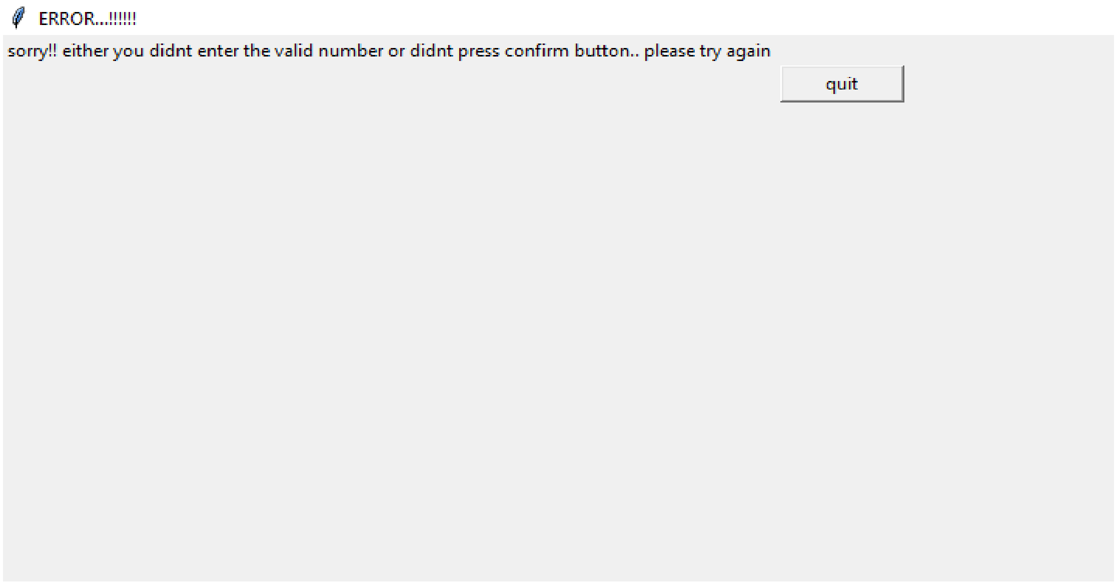
Step 4: Click on 'next' as below:


RESTAURANT REVIEW CLASSIFICATION AND RECOMMENDER SYSTEM

Enter no.of restaurants : 2

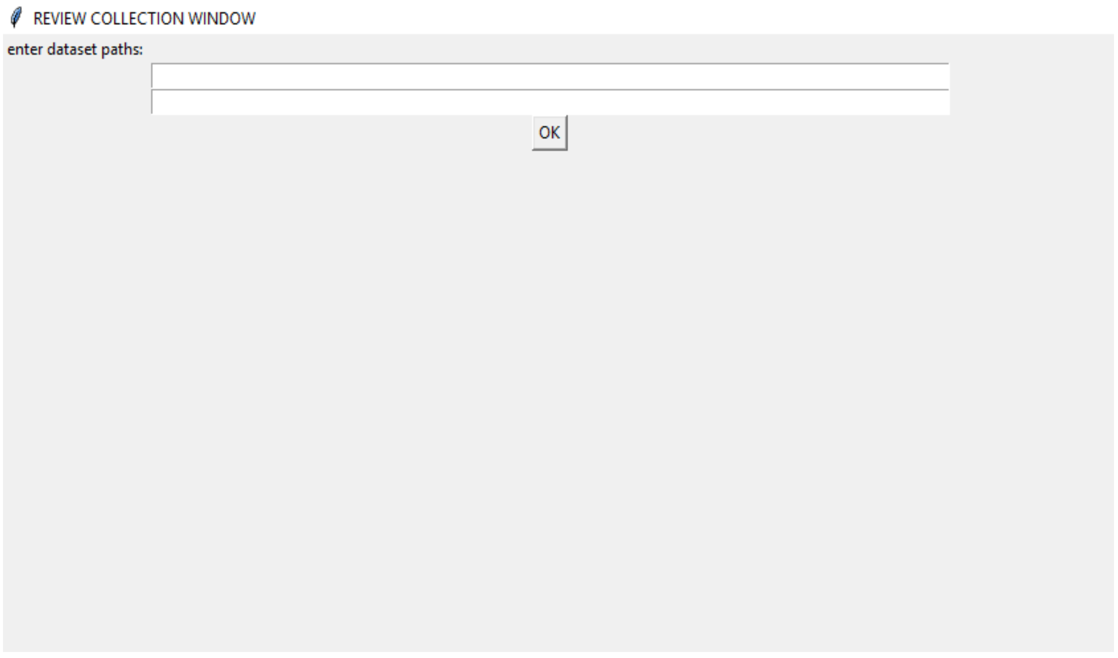
NOTE : please click the confirm_number button before u proceed

Note 2: If you haven’t entered either a valid number (i.e., 0 or any negative integer) or haven’t clicked the ‘confirm_number’ before clicking ‘next’, the following ‘ERROR…!!!!!!’ window appears. Click on ‘quit’ and go to Step 1.

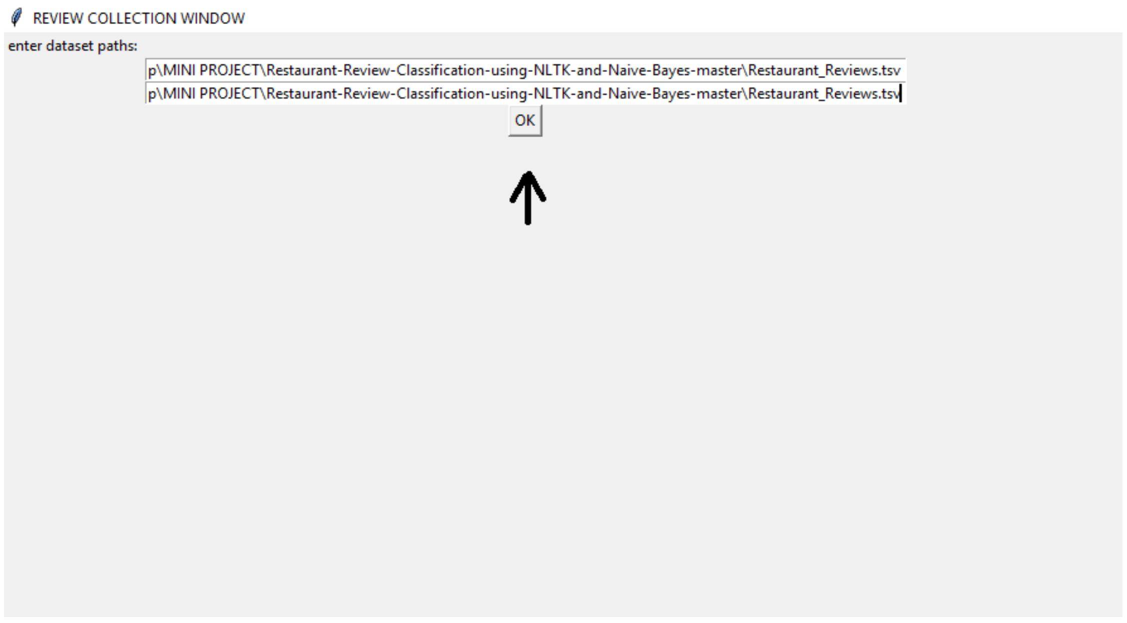


sorry!! either you didnt enter the valid number or didnt press confirm button.. please try again

Step 5: In case of no problem (as mentioned above in Note 2), the following ‘REVIEW COLLECTION WINDOW’ appears.



Step 6: Enter the locations of the restaurant reviews dataset in the specified text boxes and click on 'ok' as below:

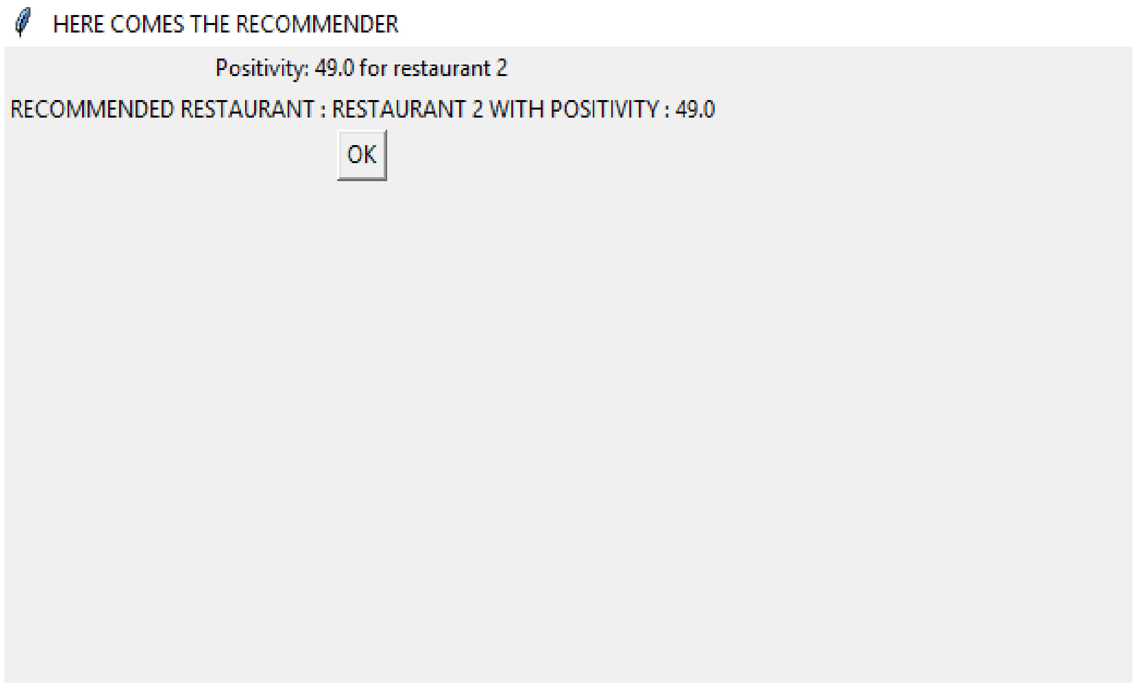


REVIEW COLLECTION WINDOW

enter dataset paths:

```
p\MINI PROJECT\Restaurant-Review-Classification-using-NLTK-and-Naive-Bayes-master\Restaurant_Reviews.tsv
p\MINI PROJECT\Restaurant-Review-Classification-using-NLTK-and-Naive-Bayes-master\Restaurant_Reviews.tsv
```

Step 7: After clicking on 'ok', the window disappears. Within a very few seconds, the following 'HERE COMES THE RECOMMENDER' window appears with the desired output.



HERE COMES THE RECOMMENDER

```
Positivity: 49.0 for restaurant 2
RECOMMENDED RESTAURANT : RESTAURANT 2 WITH POSITIVITY : 49.0
```

4.3 PERFORMANCE OF THE MODEL

The performance of any model can be evaluated by the following measures:
a) Accuracy
b) Error Rate
c) Precision
d) Recall
e) F1 Score

4.3.1 ACCURACY

Accuracy is one metric for evaluating classification models. Informally, accuracy is the fraction of predictions our model got right. Formally, accuracy has the following definition:
Accuracy = Number of correct predictions /Total number of predictions

4.3.2 ERROR RATE

If target values are categorical, the error is expressed as an error rate. This is the proportion of cases where the prediction is wrong.
Error rate= Number of incorrect predictions /Total number of predictions

4.3.3 PRECISION

Precision is defined as the fraction of the examples which are actually positive among all the examples which we predicted positive.
Precision=Number of true positives/Total number of predicted positives
Total number of predicted positives = Number of true positives + Number of false positives

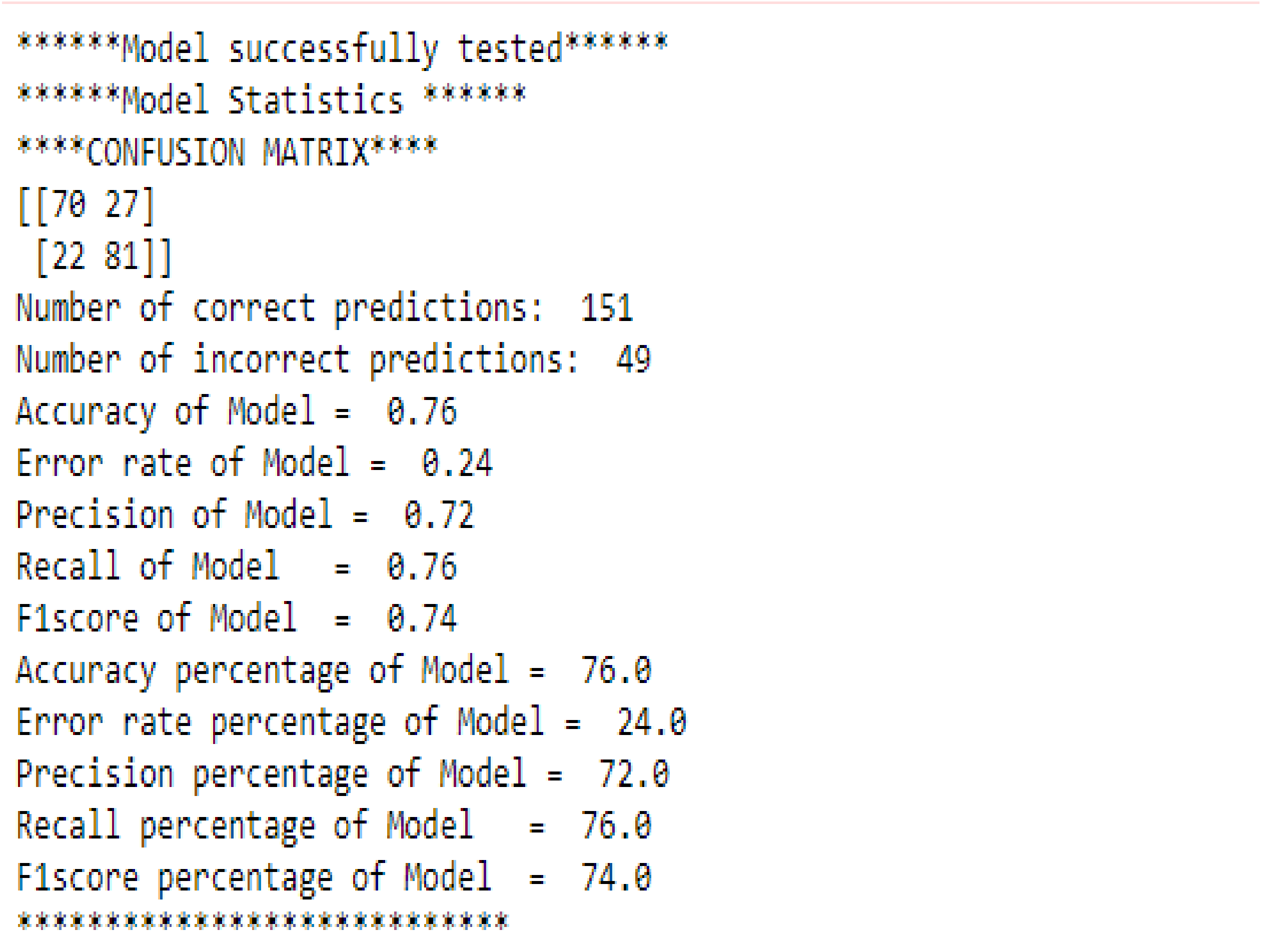
4.3.4 RECALL

Recall (also known as sensitivity) is the fraction of relevant instances that have been retrieved over the total amount of relevant instances.
Recall=Number of true positives/Actual number of positives
Actual number of positives = Number of true positives + Number of false negatives

4.3.5 F1 SCORE

The F1 score is the harmonic average of the precision and recall, where an F1 score reaches its best value at 1 (perfect precision and recall) and worst at 0.

4.3.6 THE STATISTICS OF THE PERFORMANCE OF OUR MODEL IS SHOWN BELOW:



```
******Model successfully tested******
******Model Statistics ******
****CONFUSION MATRIX****
[[70 27]
 [22 81]]
Number of correct predictions:  151
Number of incorrect predictions:  49
Accuracy of Model =  0.76
Error rate of Model =  0.24
Precision of Model =  0.72
Recall of Model    =  0.76
F1score of Model   =  0.74
Accuracy percentage of Model =  76.0
Error rate percentage of Model =  24.0
Precision percentage of Model =  72.0
Recall percentage of Model    =  76.0
F1score percentage of Model   =  74.0
****************************
```

CHAPTER 5

CONCLUSIONS AND FUTURE SCOPE

5.1 CONCLUSIONS

The project entitled **‘Restaurant Review Classification and Recommender System’** is a model for classifying the restaurant reviews and recommending the best restaurant.

Recommender systems have great value in recommending relevant resources to users. It can be quite useful in finding novel and serendipitous recommendations. The effectiveness of recommender system relies on the algorithm it uses to find interesting resources. This thesis presents software-based restaurant recommendation system which is written in PYTHON programming language by making use of ANACONDA-JUPYTER NOTEBOOK environment.

We proposed to build a software that can efficiently predict the positivity of a restaurant and thereby predict the best restaurant for user convenience.

Today’s generation encourages high-tech services especially over the Internet. Hence, the project is developed proficiently not only to help customers to know what is the best restaurant based on reviews but also saves a lot of time and effort. The customer need not ask anyone for suggestions regarding which restaurant he should go for dinner and he even need not search through ‘GOOGLE’ for reviews.

5.2 FUTURE SCOPE

In the near future, recommender systems will be installed in Apache Server and so they will be published in internet. Datasets will be updated continuously and it will make online actual rating predictions to the users whose habits are changing day by day. As a result, it can be sensitively satisfying current user tastes. Web services in particular suffer from producing recommendations of millions of items to millions of users. The time and computational power can even limit the performance of the best hybrid systems. For larger dataset, we can work on scalability problems of recommendation systems. The Prediction approach can also be tried in different datasets to test harmony performance of system scalability problems of recommendation systems.

BIBLIOGRAPHY

1. Adomavicius, G., Tuzhilin, E.: Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. IEEE Trans. Knowl. Data Eng. 17, 734–749 (2005)
2. Arakawa, K., Odagawa, S., Matsushita, F., Kodama, Y., Shioda, T.: Analysis of listeners' favoritemusicbymusicfeatures.In:ProceedingsoftheInternationalConferenceonConsumer Electronics (ICCE), pp. 427–428, IEEE (2006)
3. Baeza-Yates,R.,Ribeiro-Neto,B.(eds.):ModernInformationRetrieval.Addison-Wesley,New York (1999)
4. Balabanovi´c, M., Shoham, Y.: Fab: content-based, collaborative recommendation. Commun. ACM 40(3), 66–72 (1997). doi:10.1145/245108.245124
5. Basu, C., Hirsh, H., Cohen, W.: Recommendation as classification: using social and contentbased information in recommendation. In: Proceedings of the Fifteenth National/Tenth Conference on Artificial intelligence/Innovative Applications of Artificial Intelligence AAAI'98/IAAI'98,pp.714–720.AmericanAssociationforArtificialIntelligence,MenloPark (1998)
6. Billsus, D., Pazzani, M.J.: User modeling for adaptive news access. User Model. User-Adapt. Interact. 10(2–3), 147–180 (2000). doi:10.1023/A:1026501525781
7. Breese, J.S., Heckerman, D., Kadie, C.: Empirical analysis of predictive algorithms for collaborative filtering. In: Proceedings of Fourteenth Conference on Uncertainty in Artificial Intelligence, pp. 43–52. Morgan Kaufmann (1998)
8. Burke, R.: Knowledge-based recommender systems (2000)
9. Burke, R.: Hybrid recommender systems: survey and experiments. User Model. User-Adapt. Interact. 12(4), 331–370 (2002). doi:10.1023/A:1021240730564
10. Celma,O.,Ramrez,M.,Herrera,P.:Foafingthemusic:amusicrecommendationsystembased on RSS feeds and user preferences. In: Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR). London (2005)
11. Claypool, M., Gokhale, A., Miranda, T., Murnikov, P., Netes, D., Sartin, M.: Combining content-basedandcollaborativefiltersinanonlinenewspaper.In:ProceedingsofACMSIGIR Workshop on Recommender Systems (1999)
12. Cox,I.J.,Miller,M.L.,Omohundro,S.M.,Yianilos,P.N.:PicHunter:Bayesianrelevancefeedback for image retrieval. Int. Conf. Pattern Recognit., 3, 361 (1996). doi:10.1109/ICPR.1996. 546971
13. Doulamis, N.D., Doulamis, A.D., Varvarigou, T.A.: Adaptive algorithms for interactive multimedia. IEEE MultiMed. 10(4), 38–47 (2003). doi:10.1109/MMUL.2003.1237549
14. Foote, J.: An overview of audio information retrieval. Multimed. Syst. 7(1), 2–10 (1999)
15. Grimaldi, M., Cunningham, P.: Experimenting with music taste prediction by user profiling

16. Herlocker,J.L.,Konstan,J.A.:Content-independenttask-focusedrecommendation.IEEEInternet Comput. 5(6), 40–47 (2001). doi:10.1109/4236.968830
17. Herlocker, J.L., Konstan, J.A., Riedl, J.: Explaining collaborative filtering recommendations. In: Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work CSCW'00, pp. 241–250. ACM, New York (2000). doi:10.1145/358916.358995
18. Herlocker, J.L., Konstan, J.A., Riedl, J.: An empirical analysis of design choices in neighborhood-basedcollaborativefilteringalgorithms.Inf.Retr.5(4),287–310(2002).doi:10. 1023/A:1020443909834
19. Hoashi, K., Matsumo, K., Inoue, N.: Personalization of user profiles for content-based music retrieval based on relevance feedback. In: Proceedings of ACM International Conference on Multimedia 2003, pp. 110–119, ACM Press (2003)
20. Karypis, G.: Evaluation of item-based top-n recommendation algorithms. In: Proceedings of the Tenth International Conference on Information and Knowledge Management CIKM'01, pp. 247–254. ACM, New York (2001). doi:10.1145/502585.502627
21. Krulwich, B.: Lifestyle finder: intelligent user profiling using large-scale demographic data. AI Mag. 18(2), 37–45 (1997)
22. Linden, G., Smith, B., York, J.: http://Amazon.com recommendations: item-to-item collaborative filtering. IEEE Internet Comput. 7(1), 76–80 (2003). doi:10.1109/MIC.2003.1167344
23. Liu, D.R., Shih, Y.Y.: Integrating AHP and data mining for product recommendation based on customer lifetime value. Inf. Manag. 42(3), 387–400 (2005). doi:10.1016/j.im.2004.01.008
24. Logan, B.: Music recommendation from song sets. In: Proceedings of 5th International Conference on Music Information Retrieval, pp. 425–428 (2004)
25. Mandel, M., Poliner, G., Ellis, D.: Support vector machine active learning for music retrieval ACM multimedia systems journal. ACM Multimed. Syst. J. 12(1), 3–13 (2006)
26. Melville,P.,Mooney,R.J.,Nagarajan,R.:Content boostedcollaborativefilteringforimproved recommendations. In: Proceedings of Eighteenth National Conference on Artificial intelligence, pp. 187–192. American Association for Artificial Intelligence, Menlo Park (2002)
27. Mooney, R.J., Roy, L.: Content-based book recommending using learning for text categorization. In: Proceedings of the Fifth ACM Conference on Digital Libraries, pp. 195–204. ACM DL'00, New York (2000). doi:10.1145/336597.336662
28. Pazzani, M., Billsus, D.: Learning and revising user profiles: the identification of interesting web sites. Mach. Learn. 27(3), 313–331 (1997). doi:10.1023/A:1007369909943
29. Pazzani, M.J.: A framework for collaborative, content-based and demographic filtering. Artif. Intell. Rev. 13(5–6), 393–408 (1999). doi:10.1023/A:1006544522159
30. Pennock, D.M., Horvitz, E., Lawrence, S., Giles, C.L.: Collaborative filtering by personalitydiagnosis:ahybridmemoryandmodelbasedapproach.In:Proceedingsofthe16thConference on Uncertainty in Artificial Intelligence UAI'00, pp. 473–480. Morgan Kaufmann Publishers Inc., San Francisco (2000)

APPENDIX-A

SOURCE CODE

```
#Restaurant Review Classification and Recommender system
import re
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from pandas import DataFrame
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from sklearn.naive_bayes import MultinomialNB
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
import tkinter as tk
from functools import partial
from tkinter import *
import time
class RRCRS(object):
    def __init__(self):
        self.classifier=MultinomialNB()
        self.number=0
        self.list1=[]
        self.entries=[]
    def Recommender(self):
        # Natural Language Processing
        # Importing the libraries
        # Importing the dataset
        dataset=pd.read_csv(r"C:\Users\Siri\Desktop\MINI PROJECT\Restaurant-Review-
Classification-using-NLTK-and-Naive-Bayes-master\Restaurant_Reviews.tsv",delimiter ='\t',
quoting = 3)
```

```
 # Cleaning the texts
        corpus = []
        for i in range(0, 1000):
            review = re.sub('[^a-zA-Z]', ' ', dataset['Review'][i])
            review = review.lower()
            review = review.split()
            ps = PorterStemmer()
            review = [ps.stem(word) for word in review if not word in set(stopwords.words('english'))]
            review = ' '.join(review)
            corpus.append(review)

        # Creating the Bag of Words model
        cv = CountVectorizer(max_features = 1500)
        X = cv.fit_transform(corpus).toarray()
        y = dataset.iloc[:, 1].values

        # Splitting the dataset into the Training set and Test set
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = 0)

        # Fitting Naive Bayes to the Training set
        self.classifier.fit(X_train, y_train)
    def classify(self,path):
        # Importing the testdata
        dataset = pd.read_csv(path,delimiter ='\t', quoting = 3)
        corpus = []
        for i in range(0, 1000):
            review = re.sub('[^a-zA-Z]', ' ', dataset['Review'][i])
            review = review.lower()
            review = review.split()
            ps = PorterStemmer()
            review = [ps.stem(word) for word in review if not word in set(stopwords.words('english'))]
            review = ' '.join(review)
            corpus.append(review)
        # Creating the Bag of Words model
        cv = CountVectorizer(max_features = 1500)
        X = cv.fit_transform(corpus).toarray()
        y = dataset.iloc[:, 1].values
        # Predicting the Test set results
        y_pred = self.classifier.predict(X)
        pos=0
        for i in y_pred:
            if(i==1):
                pos+=1
        return (pos/len(y_pred))*100
    def print_rs(self,path):
        return self.classify(path)
    def getn(*args):
        args[0].number =int(args[1].get())
    def sm(*args):
        args[1].destroy()
        root1=tk.Tk()
        if(args[0].number>0):
            root1.title("REVIEW COLLECTION WINDOW")
            label=Label(root1,text="enter dataset paths: ")
            label.grid(column=0,row=0)
            args[0].entries.extend(Entry(root1,width=100) for _ in range(args[0].number))
            j=2
            for entry in args[0].entries:
                entry.grid(column=2,row=j)
                j+=2
            print1=partial(args[0].print_input,root1)
            btn=Button(root1,text="OK",command=print1).grid(column=2,row=j)
        else:
            root1.title("ERROR...!!!!!!")
            l1=Label(root1,text="sorry!! either you didnt enter the valid  number or didnt press confirm button.. please try again ")
            l1.grid(row=0,column=0)
            q=partial(args[0].quit,root1)
            b2=Button(root1,text='quit',width=10,command=q)
            b2.grid(row=2,column=2)
    def quit(*args):
        args[1].destroy()
    def print_input(*args):
        for entry in args[0].entries:
            args[0].list1.append(entry.get())
        args[1].destroy()
        args[0].process()
    def process(self):
        root_p=tk.Tk()
        l1=Label(root_p,text="Processing....")
        l1.grid(row=3,column=0)
        maxpos=dict()
        rows=5
        for i in range(len(self.list1)):
            maxpos[self.print_rs(self.list1[i])]="restaurant "+str(i+1)
            l2=Label(root_p,text="Processed "+str(i+1)+" dataset(s)")
            l2.grid(row=rows,column=2)
            rows+=2
            time.sleep(3)
        root_p.destroy()
        rows=2
        res_ls=list(maxpos.items())
        result=tk.Tk()
        for i in range(len(maxpos.keys())):
            result.title("HERE COMES THE RESULTS !!!")
            l2=Label(result,text="Positivity: "+str(res_ls[i][0])+" for "+str(res_ls[i][1]))
            l2.grid(row=rows,column=0)
            rows+=2
            time.sleep(3)
        result.title(" HERE COMES THE RECOMMENDER ")
        max_pos=max(maxpos.keys())
        l3=Label(result,text="RECOMMENDED RESTAURANT : "+str(maxpos[max_pos]).upper()+ " WITH POSITIVITY : "+str(max_pos))
        l3.grid(row=rows,column=0)
        q=partial(self.quit,result)
        b1=Button(result,text="OK",command=q)
        b1.grid(row=rows+2,column=0)
def main():
    CR=RRCRS()
    CR.Recommender()
    root=tk.Tk()
    root.title("RESTAURANT REVIEW CLASSIFICATION AND RECOMMENDER SYSTEM")
    pr=partial(CR.sm,root)
    label=Label(root,text="Enter no.of restaurants : ")
    label.grid(column=0,row=0)
    label2=Label(root,text="NOTE : please click the confirm_number button before u proceed")
    label2.grid(column=0,row=1)
    e1=Entry(root,width=50)
    e1.grid(column=1,row=0)
    get=partial(CR.getn,e1)
    b3=Button(root,text='confirm_number',width=20,command=get)
    b1=Button(root,text='next',width=10,command=pr)
    b1.grid(column=1,row=2)
    q=partial(CR.quit,root)
    b2=Button(root,text='quit',width=10,command=q)
    b2.grid(column=0,row=2)
    b3.grid(column=3,row=2)
    root.mainloop()
if __name__=="__main__":
    main()
```