# COUNTERFEIT REVIEW DETECTION USING MACHINE LEARNING

**A PROJECT REPORT**

Submitted by

| | |
|---|---|
| **ABDULLAH.B** | **812420104004** |
| **AZARUDEEN.S** | **812420104016** |
| **IJAS AHAMED.T** | **812420104033** |
| **DAYANITHI.T** | **812420104020** |

Partial fulfillment for the award of the degree

Of

**BACHELOR OF ENGINEERING**

**In**

**COMPUTER SCIENCE AND ENGINEERING**

**M.I.E.T ENGINEERING COLLEGE, TRICHY – 620007**

**ANNA UNIVERSITY :: CHENNAI – 600025**

**MAY 2024**

# COUNTERFEIT REVIEW DETECTION USING MACHINE LEARNING

**A PROJECT REPORT**

Submitted by

| | |
|---|---|
| **ABDULLAH.B** | **812420104004** |
| **AZARUDEEN.S** | **812420104016** |
| **IJAS AHAMED.T** | **812420104033** |
| **DAYANITHI.T** | **812420104020** |

Partial fulfillment for the award of the degree

Of

**BACHELOR OF ENGINEERING**

**In**

**COMPUTER SCIENCE AND ENGINEERING**

**M.I.E.T ENGINEERING COLLEGE, TRICHY – 620007**

**ANNA UNIVERSITY :: CHENNAI – 600025**

**MAY 2024**

**ANNA UNIVERSITY::CHENNAI 600025**

**BONAFIDE CERTIFICATE**

Certified that this final project report " **COUNTERFEIT REVIEW DETECTION USING MACHINE LEARNING** " is the Bonafide of "**A. ABDULLAH" (812420104004), "S. AZARUDEEN" (812420104016) "T. IJAS AHAMED" (812420104033), "T. DAYANITHI" (812420104020)** who carried out the Final project under the supervision.

| | |
|---|---|
| **SIGNATURE** | **SIGNATURE** |
| MR.P.MANIKANDAN M.E., | Ms.SELVASHANTHI M.E., |
| **HEAD OF THE DEPARTMENT** | **SUPERVISOR** |
| | **ASSISTANT PROFESSOR** |
| Department of Computer Science and Engineering | Department of Computer Science and Engineering |
| M.I.E.T .Engineering College, | M.I.E.T .Engineering College, |
| Trichy – 620007 | Trichy – 620007 |

Submitted for the viva-voce held on________________

**INTERNAL EXAMINER** **EXTERNAL EXAMINER**

## ACKNOWLEDGEMENT

First of all, we thank God for this shower of blessing and his divine help which enables us to complete the project successfully.

We extend our sincere thanks to **Alhaj. Janab. Er. A. MOHAMED YUNUS, B.E., M.Sc., (Engg.)** Founder & Chairman of M.I.E.T. Engineering College, Trichy for offering the means of attending our most cherished Goal Environment.

We extend our deepest gratitude to Principal **Dr. A. NAVEEN SAIT, M.E., Ph.D.,** M.I.E.T. Engineering College, Trichy, for providing us permission to do the project work successfully.

We are grateful to express our profound thanks to the head of the department, CSE **Mr .P .MANIKANDAN, M.E.,** who has been source of encouragement and moral strength throughout our study period.

It gives immense pleasure to extend my sincere and heartfelt gratitude to our project guide **MS. SELVASHANTHI B.Tech., M.E.,** Assistant Professor for her valuable untiring and timely suggestion indispensable situation during the study period.

We are extremely thankful to our parents for enlightening us by providing Professional education and for their prayerful support that makes us to complete.

Also heartfelt thanks to our friends, Teaching and non-teaching staff members who helped us to finish the project successfully.

## ABSTRACT

Fake reviews detection attracts many researchers' attention due to the negative impacts on the society. Most existing fake reviews detection approaches mainly focus on semantic analysis of review's contents. We propose a novel fake reviews XGboost technique. The increasing popularity of online review systems motivates malevolent intent in competing sellers and service providers to manipulate consumers by fabricating product/service reviews. Immoral actors use Sybil accounts, bot farms, and purchase authentic accounts to promote products and vilify competitors. Facing the continuous advancement of review spamming techniques, the research community should step back, assess the approaches explored to date to combat fake reviews, and regroup to define new ones. This paper reviews the literature on Fake Review Detection (FRD) on online platforms. It covers both basic research and commercial solutions, and discusses the reasons behind the limited level of success that the current approaches and regulations have had in preventing damage due to deceptive reviews

# TABLE OF CONTENTS

# LIST OF FIGURES

| FIGURE NO | FIGURE NAME | PAGE NO |
|---|---|---|

# LIST OF ABBREVIATIONS

| S.NO. | SHORT FORMS | PAGE NO |
|---|---|---|

## ABSTRACT

Fake reviews detection attracts many researchers' attention due to the negative impacts on the society. Most existing fake reviews detection approaches mainly focus on semantic analysis of review's contents. We propose a novel fake reviews XGboost technique. The increasing popularity of online review systems motivates malevolent intent in competing sellers and service providers to manipulate consumers by fabricating product/service reviews. Immoral actors use Sybil accounts, bot farms, and purchase authentic accounts to promote products and vilify competitors. Facing the continuous advancement of review spamming techniques, the research community should step back, assess the approaches explored to date to combat fake reviews, and regroup to define new ones. This paper reviews the literature on Fake Review Detection (FRD) on online platforms. It covers both basic research and commercial solutions, and discusses the reasons behind the limited level of success that the current approaches and regulations have had in preventing damage due to deceptive reviews.

# CHAPTER 1

# INTRODUCTION

## 1.1 MACHINE LEARNING

Predictive analytics tools are powered by several different models and algorithms that can be applied to wide range of use cases. Determining what predictive modeling techniques are best for your company is key to getting the most out of a predictive analytics solution and leveraging data to make insightful decisions in the statistical context, Machine Learning is defined as an application of artificial intelligence where available information is used through algorithms to process or assist the processing of statistical data. While Machine Learning involves concepts of automation, it requires human guidance. Machine Learning involves a high level of generalization in order to get a system that performs well on yet unseen data instances

Machine learning is a relatively new discipline within Computer Science that provides a collection of data analysis techniques. Some of these techniques are based on well-established statistical methods (e.g. logistic regression and principal component analysis) while many others are not.

Most statistical techniques follow the paradigm of determining a particular probabilistic model that best describes observed data among a class of related models. Similarly, most machine learning techniques are designed to find models that best fit data (i.e. they solve certain optimization problems), except that these machine learning models are no longer restricted to probabilistic ones.

Therefore, an advantage of machine learning techniques over statistical ones is that the latter require underlying probabilistic models while the former do not. Even though some machine learning techniques use probabilistic models, the classical statistical techniques are most often too stringent for the oncoming Big Data era, because data sources are increasingly complex and multi-faceted. Prescribing probabilistic models relating variables from disparate data sources that are plausible and amenable to statistical analysis might be extremely difficult if not impossible.

Machine learning might be able to provide a broader class of more flexible alternative analysis methods better suited to modern sources of data. It is imperative for statistical agencies to explore the possible use of machine learning techniques to determine whether their future needs might be better met with such techniques than with traditional ones.

## 1.2 CLASSES OF MACHINE LEARNING

### Examples of supervised learning

Logistic regression, when used for prediction purposes, is an example of supervised machine learning. In logistic regression, the values of a binary response variable (with values 0 or 1, say) as well as a number of predictor variables (covariates) are observed for a number of observation units. These are called training data in machine learning terminology. The main hypotheses are that the response variable follows a Bernoulli distribution (a class of probabilistic models), and the link between the response and predictor variables is the relation that the logarithm of the posterior odds of the response is a linear function of the predictors. The response variables of the units are assumed to be independent of each other, and the method of maximum likelihood is applied to their joint probability distribution to find the optimal values for the coefficients (these parameterize the aforementioned joint distribution) in this linear function. The particular model with these optimal coefficient values is called the "fitted model," and can be used to "predict" the value of the response variable for a new unit (or, "classify" the new unit as 0 or 1) for which only the predictor values are known. Support Vector Machines (SVM) are an example of a non-statistical supervised machine learning technique; it has the same goal as the logistic regression classifier just described: Given training data, find the best-fitting SVM model, and then use the fitted SVM model to classify

**The most widely used predictive models are:**

**Decision trees:**

Decision trees are a simple, but powerful form of multiple variable analysis. They are produced by algorithms that identify various ways of splitting data into branch-like segments. Decision trees partition data into subsets based on categories of input variables, helping you to understand someone's path of decisions.

**Regression (linearandlogistic)**

Regression is one of the most popular methods in statistics. Regression analysis estimates relationships among variables, finding key patterns in large and diverse data sets and how they relate to each other.

**Neural networks**

Patterned after the operation of neurons in the human brain, neural networks (also called artificial neural networks) are a variety of deep learning technologies. They're typically used to solve complex pattern recognition problems – and are incredibly useful for analyzing large data sets. They are great at handling nonlinear relationships in data – and work well when certain variables are unknown.

**Developing the right environment**

While machine learning and predictive analytics can be a boon for any organization, implementing these solutions haphazardly, without considering how they will fit into everyday operations, will drastically hinder their ability to deliver the insights the organization needs

**Understanding predictive models**

Typically, an organization's data scientists and IT experts are tasked with the development of choosing the right predictive models – or building their own to meet the organization's needs. Today, however, predictive analytics and machine learning is no longer just the domain of mathematicians, statisticians and data scientists, but also that of business analysts and consultants. More and more of a business' employees are using it to develop insights and improve business operations – but problems arise when employees do not know what model to use, how to deploy it, or need information right away.

At SAS, we develop sophisticated software to support organizations with their data governance and analytics. Our data governance solutions help organizations to maintain high-quality data, as well as align operations across the business and pinpoint data problems within the same environment. Our predictive analytics solutions help organizations to turn their data into timely insights for better, faster decision making. These predictive analytics solutions are designed to meet the needs of all types of users and enables them to deploy predictive models rapidly.

## FAKE REVIEW DETECTION

Online reviews are less trustworthy than we think. The credibility of all reviews — even real ones — is questionable. A 2016 study published in The Journal of Consumer Research looked at whether online reviews reflected objective quality as rated by Consumer Reports. The researchers found very little correlation.

## USER REVIEW

A user review is a review conducted by any person who has access to the internet and publishes their experience to a review site or social media platform following product testing or the evaluation of a service. User reviews are commonly provided by consumers who volunteer to write the review, rather than professionals who are paid to evaluate the product or service. User reviews might be compared to professional nonprofit reviews from a consumer organization, or to promotional reviews from an advertiser or company marketing a product. Growth of social media platforms has enabled the facilitation of interaction between consumers after a review has been placed on online communities such as blogs, internet forums or other popular platforms.

## CUSTOMER REVIEW

A customer review is a review of a product or service made by a customer who has purchased and used, or had experience with, the product or service. Customer reviews are a form of customer feedback on electronic commerce and online shopping sites. There are also dedicated review sites, some of which use customer reviews as well as or instead of professional reviews. The reviews may themselves be graded for usefulness or accuracy by other users.

## GOOGLE REVIEWS

With its dominance in search, reviews became a logical progression in Google's product roadmap. Google's interest with reviews peaked in 2002 when it purchased the remaining business assets of review pioneer Deja, integrating much of their content into its own ecosphere. Over time Google changed to include reviews with searches which related to products and businesses, even allowing users to leave reviews.

In later years, Google's integration with Android devices also allowed them to capitalize on their own products including reviews, facilitating the prominence and reliability on reviews from a large portion of smartphone users.

Business owners who sign up to access the Google My Business service are able to prompt their customers to leave reviews directly from the Google My Business control panel.[9] Businesses collecting reviews from Google also receive a boost to their search engine optimization from positive customer feedback on Google as well as other numerous benefits both online and off.

## WEB SCRAPING

Web scraping is an automatic method to obtain large amounts of data from websites. Most of this data is unstructured data in an HTML format which is then converted into structured data in a spreadsheet or a database so that it can be used in various applications. There are many different ways to perform web scraping to obtain data from websites. These include using online services, particular API's or even creating your code for web scraping from scratch. Many large websites, like Google, Twitter, Facebook, Stack Overflow, etc. have API's that allow you to access their data in a structured format. This is the best option, but there are other sites that don't allow users to access large amounts of data in a structured form or they are simply not that technologically advanced. In that situation, it's best to use Web Scraping to scrape the website for data.

So, when a web scraper needs to scrape a site, first the URLs are provided. Then it loads all the HTML code for those sites and a more advanced scraper might even extract all the CSS and JavaScript elements as well. Then the scraper obtains the required data from this HTML code and outputs this data in the format specified by the user. Mostly, this is in the form of an Excel spreadsheet or a CSV file, but the data can also be saved in other formats, such as a JSON file.

**What is Web Scraping used for?**

Web Scraping has multiple applications across various industries. Let's check out some of these now!

***1. Price Monitoring***

Web Scraping can be used by companies to scrap the product data for their products and competing products as well to see how it impacts their pricing strategies. Companies can use this data to fix the optimal pricing for their products so that they can obtain maximum revenue.

***2. Market Research***

Web scraping can be used for market research by companies. High-quality web scraped data obtained in large volumes can be very helpful for companies in analyzing consumer trends and understanding which direction the company should move in the future.

***3. News Monitoring***

Web scraping news sites can provide detailed reports on the current news to a company. This is even more essential for companies that are frequently in the news or that depend on daily news for their day-to-day functioning. After all, news reports can make or break a company in a single day!

***4. Sentiment Analysis***

If companies want to understand the general sentiment for their products among their consumers, then Sentiment Analysis is a must. Companies can use web scraping to collect data from social media websites such as Facebook and Twitter as to what the general sentiment about their products is. This will help them in creating products that people desire and moving ahead of their competition.

### *5. Email Marketing*

Companies can also use Web scraping for email marketing. They can collect Email ID's from various sites using web scraping and then send bulk promotional and marketing Emails to all the people owning these Email ID's.

# CHAPTER 2

# LIETRATURE SURVEY

**2.1 TITLE:** A Tangled Web: Should Online Review Portals Display Fraudulent Reviews?

**YEAR:** Uttara M. Ananthakrishnan , Beibei Li , Michael D. Smith

**AUTHOR:** 2020

**DESCRIPTION:**

The growing interest in online product reviews for legitimate promotion has been accompanied by an increase in fraudulent reviews. However, beyond algorithms for initial fraud detection, little is known about what review portals should do with fraudulent reviews after detecting them. In this paper, we address this question by studying how consumers respond to potentially fraudulent reviews and how review portals can leverage this knowledge to design better fraud management policies. To do this, we combine theoretical development from the trust literature with randomized experiments and statistical analysis using large-scale data from Yelp. We find that consumers tend to increase their trust in the information provided by review portals when the portal displays fraudulent reviews along with non-fraudulent reviews, as opposed to the common practice of censoring suspected fraudulent reviews. The impact of fraudulent reviews on consumers' decision-making process increases with the uncertainty in the initial evaluation of product quality. We also find that consumers do not effectively process the content of fraudulent reviews (negative or positive).

**2.2 TITLE:** Opinion fraud detection via neural auto encoder decision forest

**YEAR:** Manqing Dong, Lina Yao,Xianzhi Wang, Boualem Benatallah, Chaoran Huang, Xiaodong Ning

**AUTHOR:** 2020

**DESCRIPTION:**

Online reviews play an important role in influencing buyers' daily purchase decisions. However, fake and meaningless reviews, which cannot reflect users' genuine purchase experience and opinions, widely exist on the Web and pose great challenges for users to make right choices. Therefore, it is desirable to build a fair model that evaluates the quality of products by distinguishing spamming reviews. We present an end-to-end trainable unified model to leverage the appealing properties from Auto encoder and random forest. A stochastic decision tree model is implemented to guide the global parameter learning process. Extensive experiments were conducted on a large Amazon review dataset. The proposed model consistently outperforms a series of compared methods.

**2.3 TITLE:** Illusions of truth—Experimental insights into human and algorithmic detections of fake online reviews

**YEAR:** 2020

**AUTHOR:** Daria Plotkina, Andreas Munzel, Jessie Pallud

**DESCRIPTION:**

The issue of fake online reviews is increasingly relevant due to the growing importance of online reviews to consumers and the growing frequency of deceptive corporate practices. It is, therefore, necessary to be able to detect fake online reviews. An experiment with 1041 respondents allowed us to create two pools of reviews (fake and truthful) and compare them for psycholinguistic deception cues. The resulting automated tool accounted for review valence and incentive and detected deceptive reviews with 81% accuracy. A follow-up experiment with 407 consumers showed that humans have only a 57% accuracy of detection, even when a deception mindset is activated with information on cues of fake online reviews. Therefore, micro-linguistic automated detection can be used to filter the content of reviewing websites to protect online users. Our independent analysis of reviewing websites confirms the presence of dubious content and, therefore, the need to introduce more sophisticated filtering approaches.

**2.4 TITLE:** Deceptive consumer review detection: a survey

**YEAR: 2020**

**AUTHOR:** Dushyanthi U. Vidanagama, Thushari P. Silva & Asoka S. Karunananda

**DESCRIPTION:**

Consumer reviews are considered to be of utmost significance in the field of e-commerce, for they have a stronghold in deciding the revenue of a business. When arriving at a purchasing decision, a majority of online consumers rely on reviews since they offer credible means of mining opinions of other consumers regarding a particular product. The trustworthiness of online reviews directly affects a company’s reputation and profitability. Such generation of deceptive reviews which manipulate the purchasing decision of consumers is a persistent and harmful issue. Hence, developing methods to assist businesses and consumers by distinguishing between credible reviews and deceptive reviews remains to be a crucial, yet challenging task. In view of that, this paper unravels prominent techniques that have been proposed to solve the issue of deceptive review detection. Accordingly, the primary goal of this paper is to provide an in-depth analysis of current research on detecting deceptive reviews and to identify the characteristics, strengths, and bottlenecks of those methodologies which may need further improvements.

**2.5 TITLE:** Fake online reviews: Literature review, synthesis, and directions for future research

**YEAR:** 2020

**AUTHOR:** Yuanyuan Wu,Eric W.T.Ngai, Pengkun Wu, Chong Wu

**DESCRIPTION:**

Fake online reviews in e-commerce significantly affect online consumers, merchants, and, as a result, market efficiency. Despite scholarly efforts to examine fake reviews, there still lacks a survey that can systematically analyze and summarize its antecedents and consequences. This study proposes an antecedent–consequence–intervention conceptual framework to develop an initial research agenda for investigating fake reviews. Based on a review of the extant literature on this issue, we identify 20 future research questions and suggest 18 propositions. Notably, research on fake reviews is often limited by lack of high-quality datasets. To alleviate this problem, we comprehensively compile and summarize the existing fake reviews-related public datasets. We conclude by presenting the theoretical and practical implications of the current research.

**2.6 TITLE:** A Methodological Template to Construct Ground Truth of Authentic and Fake Online Reviews

**YEAR:** 2019

**AUTHOR:** Snehasish Banerjee

**DESCRIPTION:**

The emergence of opinion spam, scholars in recent years have been investigating how to distinguish between authentic and fake online reviews. In this research area however, constructing ground truth has been a tricky problem. When labeled datasets of authentic and fake reviews are unavailable, it becomes impossible to systematically investigate differences between the two. In light of this problem, the goal of this paper is three-fold: (1) To review existing approaches of developing ground truth, (2) To present an improved methodological template to construct ground truth, and (3) To conduct a quality-check of the newly constructed ground truth. The existing approaches are dissected to identify several peculiarities. The new approach invests in mitigating pitfalls in the current approaches. In the newly constructed ground truth, authentic reviews were found to be not easily distinguishable from fake reviews. Finally, new research directions are identified with the hope that scholars would be able to stay ahead in their relentless race against spammers.

**2.7 TITLE:** A framework for fake review detection in online consumer electronics retailers

**YEAR:** 2019

**AUTHOR:** Rodrigo Barbado, Oscar Araque,Carlos A.Iglesias

**DESCRIPTION:**

The impact of online reviews on businesses has grown significantly during last years, being crucial to determine business success in a wide array of sectors, ranging from restaurants, hotels to e-commerce. Unfortunately, some users use unethical means to improve their online reputation by writing fake reviews of their businesses or competitors. Previous research has addressed fake review detection in a number of domains, such as product or business reviews in restaurants and hotels. However, in spite of its economic interest, the domain of consumer electronics businesses has not yet been thoroughly studied. This article proposes a feature framework for detecting fake reviews that has been evaluated in the consumer electronics domain. The contributions are fourfold: (i) Construction of a dataset for classifying fake reviews in the consumer electronics domain in four different cities based on scraping techniques; (ii) definition of a feature framework for fake review detection; (iii) development of a fake review classification method based on the proposed framework and (iv) evaluation and analysis of the results for each of the cities under study.

**2.8 TITLE:** Spam Review Detection Techniques: A Systematic Literature Review

**YEAR:** 2019

**AUTHOR:** Naveed Hussain,Hamid Turab Mirza, ORCID,Ghulam Rasool,Ibrar Hussain, ORCID andMohammad Kaleem

**DESCRIPTION:**

Online reviews about the purchase of products or services provided have become the main source of users' opinions. In order to gain profit or fame, usually spam reviews are written to promote or demote a few target products or services. This practice is known as review spamming. In the past few years, a variety of methods have been suggested in order to solve the issue of spam reviews. In this study, the researchers carry out a comprehensive review of existing studies on spam review detection using the Systematic Literature Review (SLR) approach. Overall, 76 existing studies are reviewed and analyzed. The researchers evaluated the studies based on how features are extracted from review datasets and different methods and techniques that are employed to solve the review spam detection problem. Moreover, this study analyzes different metrics that are used for the evaluation of the review spam detection methods. This literature review identified two major feature extraction techniques and two different approaches to review spam detection.

**2.9 TITLE:** On Human Predictions with Explanations and Predictions of Machine Learning Models: A Case Study on Deception Detection

**YEAR:** 2019

**AUTHOR:** Vivian Lai , Chenhao Tan

**DESCRIPTION:**

Humans are the final decision makers in critical tasks that involve ethical and legal concerns, ranging from recidivism prediction, to medical diagnosis, to fighting against fake news. Although machine learning models can sometimes achieve impressive performance in these tasks, these tasks are not amenable to full automation. To realize the potential of machine learning for improving human decisions, it is important to understand how assistance from machine learning models affects human performance and human agency. In this paper, we use deception detection as a testbed and investigate how we can harness explanations and predictions of machine learning models to improve human performance while retaining human agency. We propose a spectrum between full human agency and full automation, and develop varying levels of machine assistance along the spectrum that gradually increase the influence of machine predictions. We find that without showing predicted labels, explanations alone slightly improve human performance in the end task.

**2.10 TITLE:** Unsupervised User Behavior Representation for Fraud Review Detection with Cold-Start Problem

**YEAR:** 2020

**AUTHOR:** Qian Li, Qiang Wu, Chengzhang Zhu, Jian Zhang & Wentao Zhao

**DESCRIPTION:**

Detecting fraud review is becoming extremely important in order to provide reliable information in cyberspace, in which, however, handling cold-start problem is a critical and urgent challenge since the case of cold-start fraud review rarely provides sufficient information for further assessing its authenticity. Existing work on detecting cold-start cases relies on the limited contents of the review posted by the user and a traditional classifier to make the decision. However, simply modeling review is not reliable since reviews can be easily manipulated. Also, it is hard to obtain high-quality labeled data for training the classifier. In this paper, we tackle cold-start problems by using a user's behavior representation rather than review contents to measure authenticity, which further consider user social relations with other existing users when posting reviews. The method is completely unsupervised. Comprehensive experiments on Yelp data sets demonstrate our method significantly outperforms the state-of-the-art methods.

# CHAPTER 3

# SYSTEM ANALYSIS

## 3.1 EXISTING SYSTEM

In the existing method, fake Reviews detection multi-task learning model has been presented which is based on the following observations:

(1) Some certain topics have higher percentages of fake reviews;

(2) Some certain news authors have higher intentions to publish fake news. FDML model investigates the impact of topic labels for the fake reviews and introduce contextual information of news at the same time to boost the detection performance on the short fake reviews. The existing methods and regulations have not yet been able to eradicate the damaging effects of fake review activity in practice. In doing so, we point at the difficulties associated with combating the different types of malignant influencers.

## 3.2 DISADVANTAGE

- Low accuracy.
- Need to increase the overall performance of the model.
- The existing model unable to detect the fake news for different dataset.

## 3.3 PROPOSED SYSTEM

- In the proposed method, we proposed the Fake reviews detection technique with architecture.
- A proposed fake review XGboost algorithm system would involve several steps to detect fake reviews and prevent them from being published on online marketplaces or review sites.

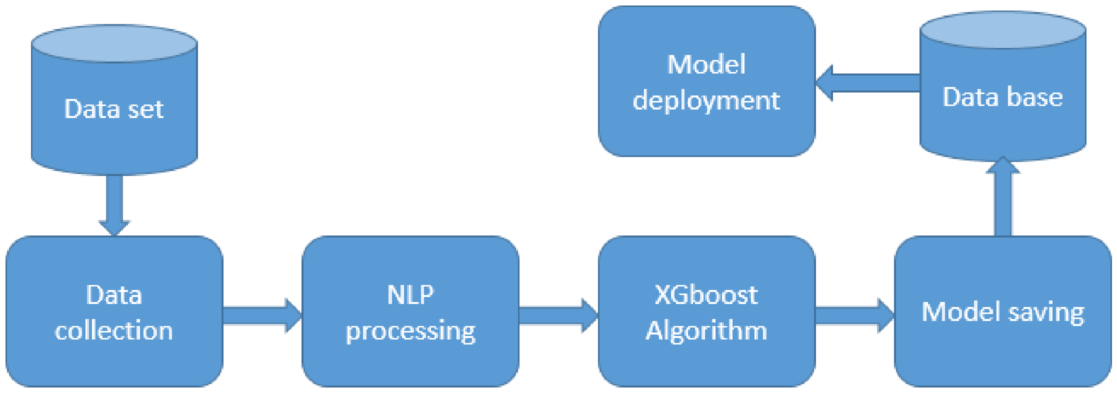
- After that the classification, XGboost Algorithm is take places in order to perform operations.

## 3.4 ADVANTAGE

- Higher accuracy of the model.
- The performance of the model is high.
- The proposed model has ability to work with different kind of dataset.

## 3.5 SYSTEM ARCHITECTURE

A system architecture is the conceptual model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviors of the system.



## XGBOOST

XGBoost (eXtreme Gradient Boosting) is a popular open-source machine learning library that is widely used for supervised learning problems such as regression and classification. It is designed to be fast and scalable, and can handle large datasets with high-dimensional features.

XGBoost is based on the gradient boosting framework, which involves sequentially adding weak learners to an ensemble model in order to improve its performance. XGBoost uses decision trees as its weak learners, and incorporates several advanced techniques to improve their performance and avoid over fitting.

One of the key features of XGBoost is its ability to handle missing values in the input data. It also includes built-in regularization techniques such as L1 and L2 regularization, which can help prevent over fitting and improve the generalization performance of the model. XGBoost is widely used in industry and has won several machine learning competitions on platforms like Kaggle. It is available in several programming languages including Python, R, Java, C++, and Scala, and can be used with a variety of data formats including CSV, TSV, and LibSVM.

# CHAPTER 4

# MODULE IMPLEMENTATION

## 4.1 MODULE LIST

- Data collection
- Preprocessing
- Data splitting
- XGboost model creation
- Model Training
- Model Evaluation
- Model Testing

## 4.2 MODULE IMPLEMENTATION

### 4.2.1 DATA SET COLLECTION

- Collect a dataset of reviews from various sources such as e-commerce websites, social media, and review websites.

### 4.2.2 PREPROCESSING

- Clean and pre-process the data by removing irrelevant information such as HTML tags, punctuation, and stop words. Also, convert the text into a numerical format that can be used as input to the XGboost model.

### 4.2.3 DATA SPLITTING

- Split the dataset into training, validation, and test sets to train and evaluate the XGboost model.

### 4.2.4 XGBOOST MODEL CREATION

- Create an XGboost model with multiple layers that can learn the patterns in the reviews and detect fake reviews.
- The model should have an input layer, multiple XGboost, and an output layer with a sigmoid activation function to predict whether a review is fake or genuine.

### 4.2.5 MODEL TRAINING

- Train the XGboost model using the training dataset and optimize the hyper parameters to achieve the best performance.

### 4.2.6 MODEL TESTING

- Test the final model on the test dataset to ensure that it can generalize well to new reviews.

### 4.2.7 DEPLOYMENT

- Deploy the XGboost model in a web application or mobile app that can automatically detect fake reviews and warn users about potential scams.

# CHAPTER 5

# SYSTEM SPECIFICATION

## H/W SYSTEM CONFIGURATION:-

- processor - Pentium – IV
- RAM - 4 GB (min)
- Hard Disk - 20 GB

## S/W SYSTEM CONFIGURATION:-

|  |  |
|---|---|
| Operating System | : Windows 7 or 8 |
| Software | : python Idle |

# SOFTWARE ENVIRONMENT

## Python Technology:

**Python** is an interpreter, high-level, general-purpose programming language. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. **Python** is often described as a "batteries included" language due to its comprehensive standard library.

## Python Programing Language:

Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming (including by Meta programming and met objects (magic methods)). Many other paradigms are supported via extensions, including design by contract and logic programming.

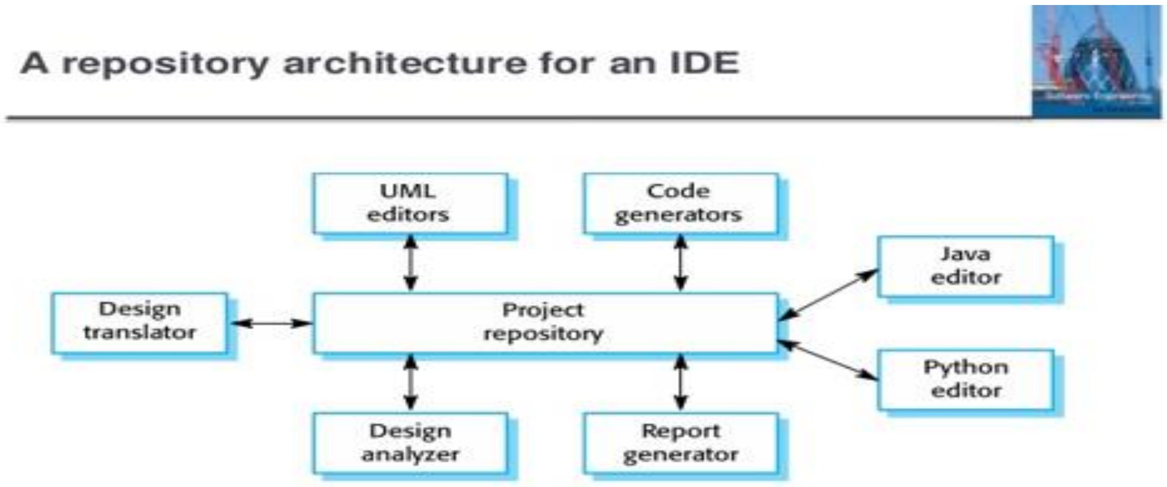
Python packages with a wide range of functionality, including:

- Easy to Learn and Use
- Expressive Language
- Interpreted Language
- Cross-platform Language
- Free and Open Source
- Object-Oriented Language
- Extensible
- Large Standard Library
- GUI Programming Support
- Integrated

Python uses dynamic typing and a combination of reference counting and a cycle-detecting garbage collector for memory management. It also features dynamic name resolution (late binding), which binds method and variable names during program execution.

Rather than having all of its functionality built into its core, Python was designed to be highly extensible. This compact modularity has made it particularly popular as a means of adding programmable interfaces to existing applications. Van Rossum's vision of a small core language with a large standard library and easily extensible interpreter stemmed from his frustrations with ABC, which espoused the opposite approach.

Python is meant to be an easily readable language. Its formatting is visually uncluttered, and it often uses English keywords where other languages use punctuation. Unlike many other languages, it does not use curly brackets to delimit blocks, and semicolons after statements are optional. It has fewer syntactic exceptions and special cases than C or Pascal.

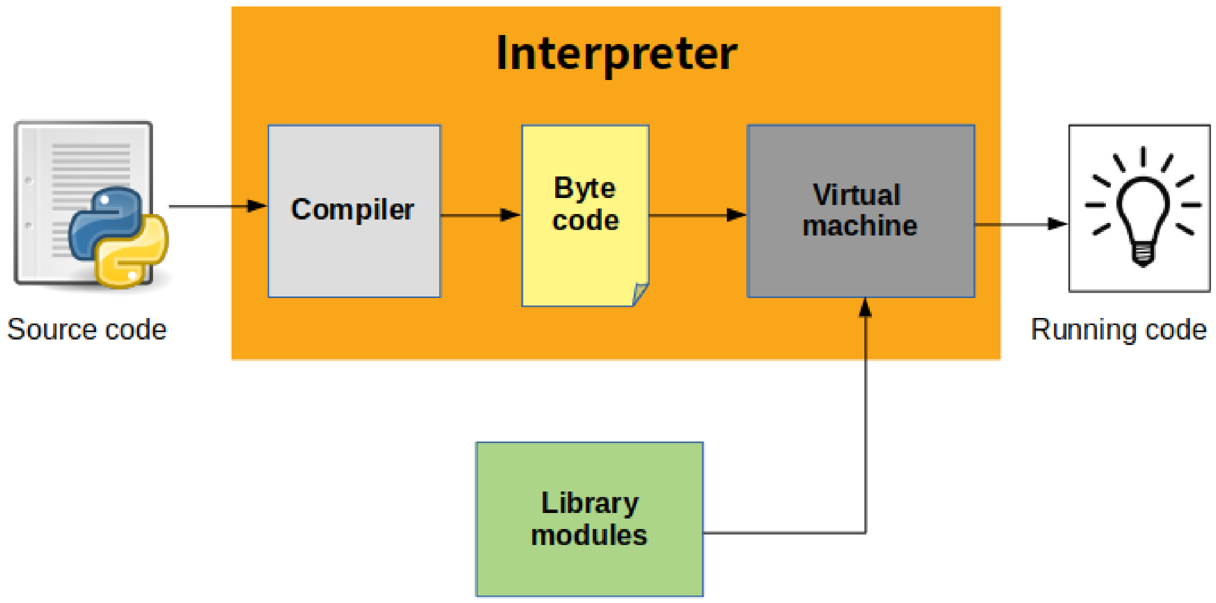


Python strives for a simpler, less-cluttered syntax and grammar while giving developers a choice in their coding methodology. In contrast to Perl's "there is more than one way to do it" motto, Python embraces a "there should be one and preferably only one obvious way to do it" design philosophy. Alex Martelli, a Fellow at the Python Software Foundation and Python book author, writes that "To describe something as 'clever' is not considered a compliment in the Python culture."

Python's developers strive to avoid premature optimization, and reject patches to non-critical parts of the Python reference implementation that would offer marginal increases in speed at the cost of clarity. When speed is important, a Python programmer can move time-critical functions to extension modules written in languages such as C, or use PyPy, a just-in-time compiler. Python is also available, which translates a Python script into C and makes direct C-level API calls into the Python interpreter.

An important goal of Python's developers is keeping it fun to use. This is reflected in the language's name a tribute to the British comedy group Monty Python and in occasionally playful approaches to tutorials and reference materials, such as examples that refer to spam and eggs (from a famous Monty Python sketch) instead of the standard foo and bar.

Python uses duck typing and has typed objects but untyped variable names. Type constraints are not checked at compile time; rather, operations on an object may fail, signifying that the given object is not of a suitable type. Despite being dynamically typed, Python is strongly typed, forbidding operations that are not well-



defined (for example, adding a number to a string) rather than silently attempting to make sense of them.

**The Python Platform:**

The platform module in Python is used to access the underlying platform's data, such as, hardware, operating system, and interpreter version information. The platform module includes tools to see the platform's hardware, operating system, and interpreter version information where the program is running.

There are four functions for getting information about the current Python interpreter. python_version() and python_version_tuple() return different forms of the interpreter version with major, minor, and patch level components. python_compiler() reports on the compiler used to build the interpreter. And python_build() gives a version string for the build of the interpreter.

Platform() returns string containing a general purpose platform identifier. The function accepts two optional Boolean arguments. If aliased is true, the names in the return value are converted from a formal name to their more common form. When terse is true, returns a minimal value with some parts dropped.

**What does python technology do?**

Python is quite popular among programmers, but the practice shows that business owners are also Python development believers and for good reason. Software developers love it for its straightforward syntax and reputation as one of the easiest programming languages to learn. Business owners or CTOs appreciate the fact that there’s a framework for pretty much anything – from web apps to machine learning.

Moreover, it is not just a language but more a technology platform that has come together through a gigantic collaboration from thousands of individual professional developers forming a huge and peculiar community of aficionados.

So what are the tangible benefits the language brings to those who decided to use it as a core technology? Below you will find just some of those reasons.

**PRODUCTIVITY AND SPEED**

It is a widespread theory within development circles that developing Python applications is approximately up to 10 times faster than developing the same application in Java or C/C++. The impressive benefit in terms of time saving can be explained by the clean object-oriented design, enhanced process control capabilities, and strong integration and text processing capacities. Moreover, its own unit testing framework contributes substantially to its speed and productivity.

**PYTHON IS POPULAR FOR WEB APPS**

Web development shows no signs of slowing down, so technologies for rapid and productive web development still prevail within the market. Along with JavaScript and Ruby, Python, with its most popular web framework Django, has great support for building web apps and is rather popular within the web development community.

## OPEN-SOURCE AND FRIENDLY COMMUNITY

As stated on the official website, it is developed under an OSI-approved open source license, making it freely usable and distributable. Additionally, the development is driven by the community, actively participating and organizing conference, meet-ups, hackathons, etc. fostering friendliness and knowledge-sharing.

## PYTHON IS QUICK TO LEARN

It is said that the language is relatively simple so you can get pretty quick results without actually wasting too much time on constant improvements and digging into the complex engineering insights of the technology. Even though Python programmers are really in high demand these days, its friendliness and attractiveness only help to increase number of those eager to master this programming language.

## BROAD APPLICATION

It is used for the broadest spectrum of activities and applications for nearly all possible industries. It ranges from simple automation tasks to gaming, web development, and even complex enterprise systems. These are the areas where this technology is still the king with no or little competence:

- Machine learning as it has a plethora of libraries implementing machine learning algorithms.
- Web development as it provides back end for a website or an app.
- Cloud computing as Python is also known to be among one of the most popular cloud-enabled languages even used by Google in numerous enterprise-level software apps.

- Scripting.
- Desktop GUI applications.

**Python compiler**

The Python compiler package is a tool for analyzing Python source code and generating Python bytecode. The compiler contains libraries to generate an abstract syntax tree from Python source code and to generate Python bytecode from the tree.

The compiler package is a Python source to bytecode translator written in Python. It uses the built-in parser and standard parser module to generate a concrete syntax tree. This tree is used to generate an abstract syntax tree (AST) and then Python bytecode.

The full functionality of the package duplicates the built-in compiler provided with the Python interpreter. It is intended to match its behavior almost exactly. Why implement another compiler that does the same thing? The package is useful for a variety of purposes. It can be modified more easily than the built-in compiler. The AST it generates is useful for analyzing Python source code.

**The basic interface**

The top-level of the package defines four functions. If you import compiler, you will get these functions and a collection of modules contained in the package.

**compiler.parse(buf)**

Returns an abstract syntax tree for the Python source code in buf. The function raises Syntax Error if there is an error in the source code. The return value is a compiler.ast. Module instance that contains the tree.

**compiler.parseFile(path)**

Return an abstract syntax tree for the Python source code in the file specified by path. It is equivalent to parse(open(path).read()).

## LIMITATIONS

There are some problems with the error checking of the compiler package. The interpreter detects syntax errors in two distinct phases. One set of errors is detected by the interpreter's parser, the other set by the compiler. The compiler package relies on the interpreter's parser, so it get the first phases of error checking for free. It implements the second phase itself, and that implementation is incomplete. For example, the compiler package does not raise an error if a name appears more than once in an argument list: def f(x, x): ...

A future version of the compiler should fix these problems.

## PYTHON ABSTRACT SYNTAX

The compiler.ast module defines an abstract syntax for Python. In the abstract syntax tree, each node represents a syntactic construct. The root of the tree is Module object.

The abstract syntax offers a higher level interface to parsed Python source code. The parser module and the compiler written in C for the Python interpreter use a concrete syntax tree. The concrete syntax is tied closely to the grammar description used for the Python parser. Instead of a single node for a construct, there are often several levels of nested nodes that are introduced by Python's precedence rules.

The abstract syntax tree is created by the compiler.transformer module. The transformer relies on the built-in Python parser to generate a concrete syntax tree. It generates an abstract syntax tree from the concrete tree.

The transformer module was created by Greg Stein and Bill Tutt for an experimental Python-to-C compiler. The current version contains a number of modifications and improvements, but the basic form of the abstract syntax and of the transformer are due to Stein and Tutt.

## AST NODES

The compiler.ast module is generated from a text file that describes each node type and its elements. Each node type is represented as a class that inherits from the abstract base class compiler.ast.Node and defines a set of named attributes for child nodes.

class compiler.ast.Node

The Node instances are created automatically by the parser generator. The recommended interface for specific Node instances is to use the public attributes to access child nodes. A public attribute may be bound to a single node or to a sequence of nodes, depending on the Node type. For example, the bases attribute of the Class node, is bound to a list of base class nodes, and the doc attribute is bound to a single node.

Each Node instance has a lineno attribute which may be None. XXX Not sure what the rules are for which nodes will have a useful lineno.

**All Node objects offer the following methods:**

**getChildren()**

Returns a flattened list of the child nodes and objects in the order they occur. Specifically, the order of the nodes is the order in which they appear in the Python grammar. Not all of the children are Node instances. The names of functions and classes, for example, are plain strings.

**getChildNodes()**

Returns a flattened list of the child nodes in the order they occur. This method is like getChildren(), except that it only returns those children that are Node instances.

The While node has three attributes: test, body, and else_. (If the natural name for an attribute is also a Python reserved word, it can’t be used as an attribute name. An underscore is appended to the word to make it a legal identifier, hence else_ instead of else.)

The if statement is more complicated because it can include several tests.

The If node only defines two attributes: tests and else_. The tests attribute is a sequence of test expression, consequent body pairs. There is one pair for each if/elif clause. The first element of the pair is the test expression. The second elements is a Stmt node that contains the code to execute if the test is true.

The getChildren() method of If returns a flat list of child nodes. If there are three if/elif clauses and no else clause, then getChildren() will return a list of six elements: the first test expression, the first Stmt, the second text expression, etc.

The following table lists each of the Node subclasses defined in compiler.ast and each of the public attributes available on their instances. The values of most of the attributes are themselves Node instances or sequences of instances. When the value is something other than an instance, the type is noted in the comment. The attributes are listed in the order in which they are returned by getChildren() and getChildNodes().

## DEVELOPMENT ENVIRONMENTS:

Most Python implementations (including CPython) include a read–eval–print loop (REPL), permitting them to function as a command line interpreter for which the user enters statements sequentially and receives results immediately.

Other shells, including IDLE and IPython, add further abilities such as auto-completion, session state retention and syntax highlighting.

## IMPLEMENTATIONS

### Reference implementation

CPython is the reference implementation of Python. It is written in C, meeting the C89 standard with several select C99 features. It compiles Python programs into an intermediate bytecode which is then executed by its virtual machine. CPython is distributed with a large standard library written in a mixture of C and native Python. It is available for many platforms, including Windows and most modern Unix-like systems. Platform portability was one of its earliest priorities.

### Other implementations

PyPy is a fast, compliant interpreter of Python 2.7 and 3.5. Its just-in-time compiler brings a significant speed improvement over CPython but several libraries written in C cannot be used with it.

Stackless Python is a significant fork of CPython that implements microthreads; it does not use the C memory stack, thus allowing massively concurrent programs. PyPy also has a stackless version.

MicroPython and CircuitPython are Python 3 variants optimized for microcontrollers. This includes Lego Mindstorms EV3.

RustPython is a Python 3 interpreter written in Rust.

**Unsupported implementations**

Other just-in-time Python compilers have been developed, but are now unsupported:

Google began a project named Unladen Swallow in 2009, with the aim of speeding up the Python interpreter five-fold by using the LLVM, and of improving its multithreading ability to scale to thousands of cores, while ordinary implementations suffer from the global interpreter lock.

Psyco is a just-in-time specialising compiler that integrates with CPython and transforms bytecode to machine code at runtime. The emitted code is specialized for certain data types and is faster than standard Python code.

In 2005, Nokia released a Python interpreter for the Series 60 mobile phones named PyS60. It includes many of the modules from the CPython implementations and some additional modules to integrate with the Symbian operating system. The project has been kept up-to-date to run on all variants of the S60 platform, and several third-party modules are available. The Nokia N900 also supports Python with GTK widget libraries, enabling programs to be written and run on the target device.

**Cross-compilers to other languages**

There are several compilers to high-level object languages, with either unrestricted Python, a restricted subset of Python, or a language similar to Python as the source language:

- Jython enables the use of the Java class library from a Python program.
- IronPython follows a similar approach in order to run Python programs on the .NET Common Language Runtime.

- The RPython language can be compiled to C, and is used to build the PyPy interpreter of Python.
- Pyjs compiles Python to JavaScript.
- Cython compiles Python to C and C++.
- Numba uses LLVM to compile Python to machine code.
- Pythran compiles Python to C++.
- Somewhat dated Pyrex (latest release in 2010) and Shed Skin (latest release in 2013) compile to C and C++ respectively.
- Google's Grumpy compiles Python to Go.
- MyHDL compiles Python to VHDL.
- Nuitka compiles Python into C++.

## PERFORMANCE

A performance comparison of various Python implementations on a non-numerical (combinatorial) workload was presented at EuroSciPy '13.

## API DOCUMENTATION GENERATORS

Python API documentation generators include:

- Sphinx
- Epydoc
- HeaderDoc
- Pydoc

## USES

Python has been successfully embedded in many software products as a scripting language, including in finite element method software such as Abaqus, 3D parametric modeler like FreeCAD, 3D animation packages such as 3ds Max, Blender, Cinema 4D, Lightwave, Houdini, Maya, modo, MotionBuilder, Softimage, the visual effects compositor Nuke, 2D imaging programs like GIMP, Inkscape, Scribus and Paint Shop Pro, and musical notation programs like scorewriter and capella. GNU Debugger uses Python as a pretty printer to show complex structures such as C++ containers. Esri promotes Python as the best choice for writing scripts in ArcGIS. It has also been used in several video games, and has been adopted as first of the three available programming languages in Google App Engine, the other two being Java and Go.

Python is commonly used in artificial intelligence projects with the help of libraries like TensorFlow, Keras and Scikit-learn. As a scripting language with modular architecture, simple syntax and rich text processing tools, Python is often used for natural language processing.

Many operating systems include Python as a standard component. It ships with most Linux distributions, AmigaOS 4, FreeBSD (as a package), NetBSD, OpenBSD (as a package) and macOS and can be used from the command line (terminal). Many Linux distributions use installers written in Python: Ubuntu uses the Ubiquity installer, while Red Hat Linux and Fedora use the Anaconda installer. Gentoo Linux uses Python in its package management system, Portage.

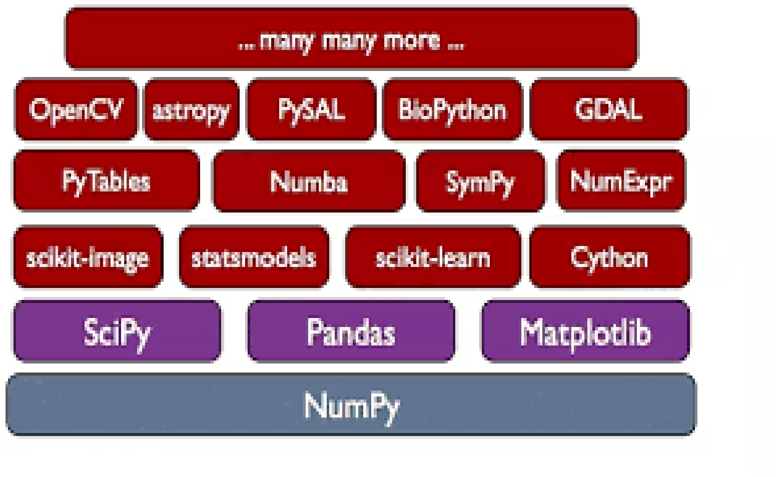
Python is used extensively in the information security industry, including in exploit development.

Most of the Sugar software for the One Laptop per Child XO, now developed at Sugar Labs, is written in Python. The Raspberry Pi single-board computer project has adopted Python as its main user-programming language.

LibreOffice includes Python, and intends to replace Java with Python. Its Python Scripting Provider is a core feature since Version 4.0 from 7 February 2013.

## PANDAS

In computer programming, pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license. The name is derived from the term "panel data", an econometrics term for data sets that include observations over multiple time periods for the same individuals.



### Library features

- Data Frame object for data manipulation with integrated indexing.
- Tools for reading and writing data between in-memory data structures and different file formats.
- Data alignment and integrated handling of missing data.
- Reshaping and pivoting of data sets.
- Label-based slicing, fancy indexing, and sub setting of large data sets.
- Data structure column insertion and deletion.
- Group by engine allowing split-apply-combine operations on data sets.
- Data set merging and joining.

- Hierarchical axis indexing to work with high-dimensional data in a lower-dimensional data structure.
- Time series-functionality: Date range generation and frequency conversion, moving window statistics, moving window linear regressions, date shifting and lagging.
- Provides data filtration.

# CHAPTER 6

# SYSTEM DESIGN

## 6.1 UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

### GOALS:

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

## 6.2 USE CASE DIAGRAM:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

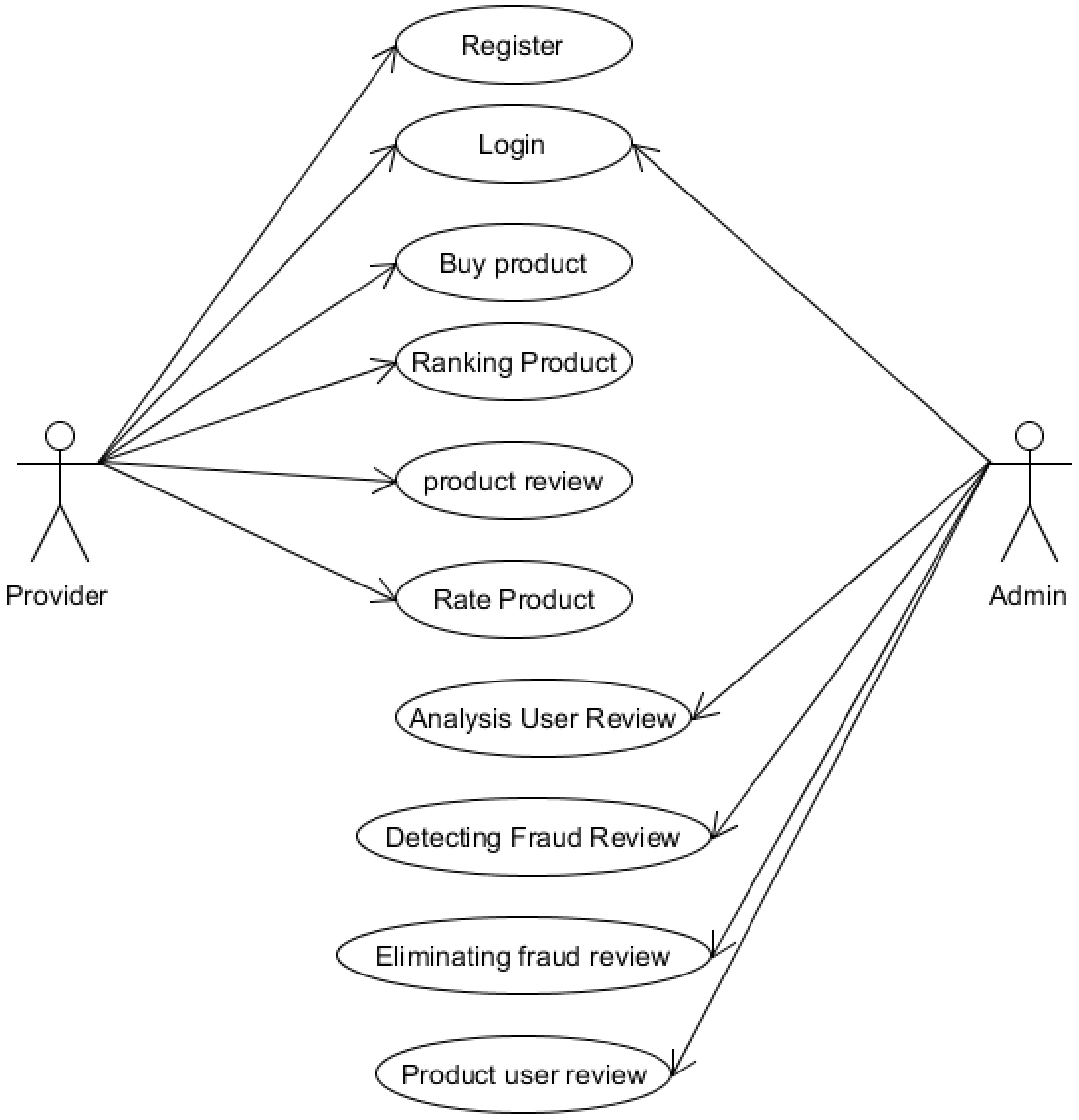


**Fig 6.2 Use case diagram**

## 6.3 CLASS DIAGRAM:

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

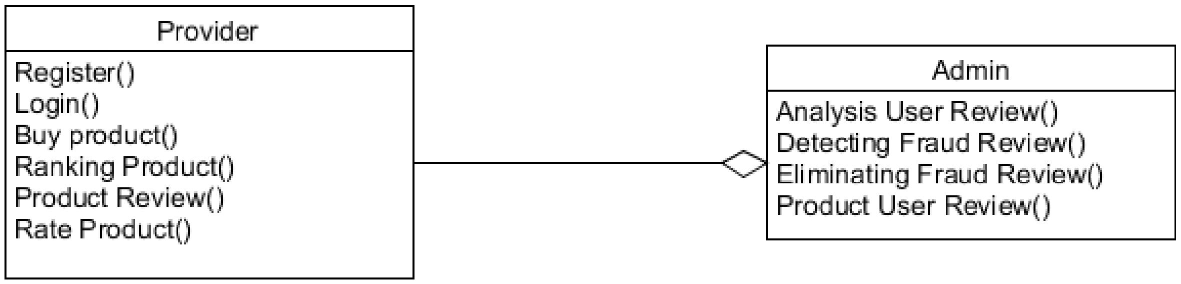


**Fig 6.3: Class diagram**

## 6.4 SEQUENCE DIAGRAM:

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

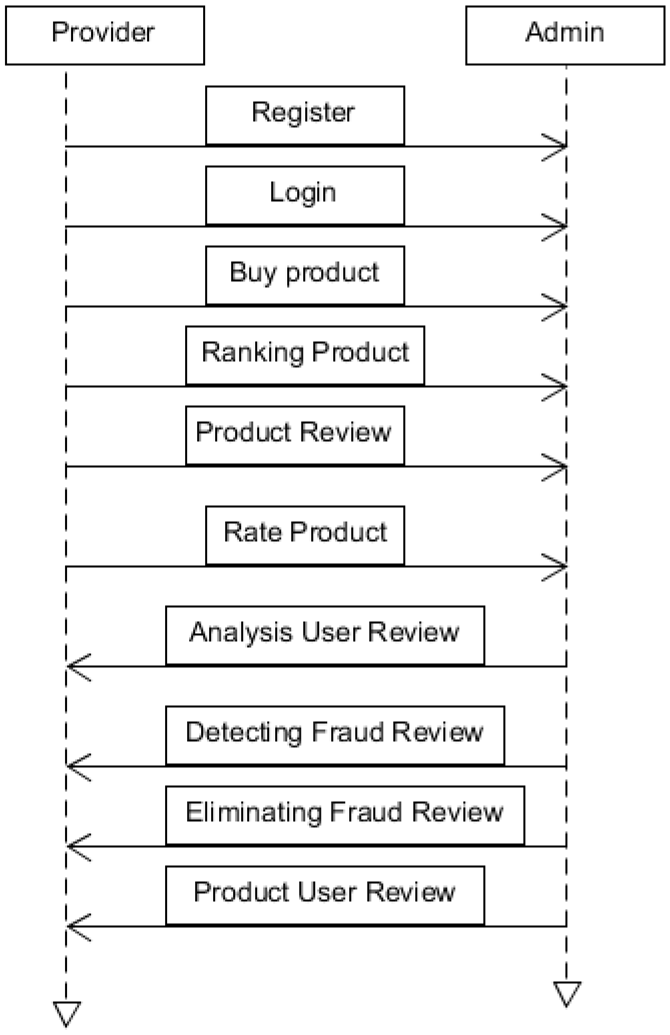
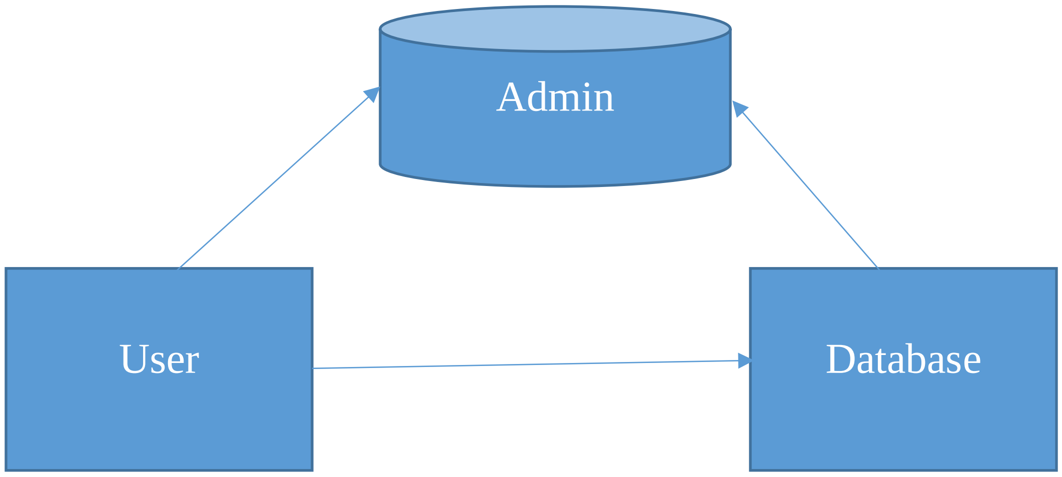


**Fig 6.4: sequence diagram**

## 6.5 DEPLOYMENT:

Component diagrams are used to describe the components and deployment diagrams shows how they are deployed in hardware. UML is mainly designed to focus on the software artifacts of a system. However, these two diagrams are special diagrams used to focus on software and hardware components.



## 6.6 DATA FLOW DIAGRAM:

1. The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
2. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.

3. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
4. DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.

# CHAPTER 7
# SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product it is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

## TYPES OF TESTS

### Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**Integration testing**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

**Functional test**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

| | |
|---|---|
| Valid Input | : identified classes of valid input must be accepted. |
| Invalid Input | : identified classes of invalid input must be rejected. |
| Functions | : identified functions must be exercised. |

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

***System Test***

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

***White Box Testing***

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

## Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

## 7.1 Unit Testing:

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

### Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

### Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

**Features to be tested**

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

## 7.2 Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

## 7.3 Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

# CHAPTER 8

# CONCLUSION

We focused on the task of identifying spam reviews. After analyzing the reviews in the datasets, we propose a hypothesis that fine-grained aspect information can be used as a new scheme for fake review detection and reconstructed the representation of reviews from four perspectives: users, products, reviews text, and fine-grained aspects. We proposed a multilevel interactive attention neural network model with aspect plan; to optimize the model's objective function, we transformed the implicit relationship between users, reviews and products into a regularization term. To verify the effectiveness of the MIANA, we conducted extensive experiments on three public datasets. Our experiments showed that the classification effect has been significantly improved, that the MIANA outperforms the state-of-the-art methods for fake review detection tasks, and proved the effectiveness and feasibility of our proposed scheme.

## REFERENCE

[1] R. Filieri and F. McLeay, ‘‘E-WOM and accommodation: An analysis of the factors that influence travelers' adoption of information from online reviews,’’ J. Travel Res., vol. 53, no. 1, pp. 44–57, Jan. 2014.

[2] E. Kauffmann, J. Peral, D. Gil, A. Ferrández, R. Sellers, and H. Mora, ‘‘A framework for big data analytics in commercial social networks: A case study on sentiment analysis and fake review detection for marketing decision-making,’’ Ind. Marketing Manage., vol. 90, pp. 523–537, Oct. 2020.

[3] N. Jindal and B. Liu, ‘‘Review spam detection,’’ in Proc. 16th Int. Conf. World Wide Web, 2007, pp. 1189–1190

[4] A. Mukherjee, V. Venkataraman, B. Liu, and N. S. Glance, ‘‘what yelp fake review filter might be doing,’’ in Proc. ICWSM, 2013, pp. 409–418.

[5] S. Rayana and L. Akoglu, ‘‘Collective opinion spam detection: Bridging review networks and metadata,’’ in Proc. 21th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, Aug. 2015, pp. 985–994.

[6] F. Li, M. Huang, Y. Yang, and X. Zhu, ‘‘Learning to identify review spam,’’ in Proc. IJCAI 22nd Int. Joint Conf. Artif. Intell., vol. 3, 2011, pp. 2488–2493.

[7] X. Hu, J. Tang, H. Gao, and H. Liu, ‘‘Social spammer detection with sentiment information,’’ in Proc. IEEE Int. Conf. Data Mining, Dec. 2014, pp. 180–189.

[8] S. Kc and A. Mukherjee, ‘‘on the temporal dynamics of opinion spamming: Case studies on yelp,’’ in Proc. 25th Int. Conf. World Wide Web, Apr. 2016, pp. 369–379.

[9] Y. Ren and Y. Zhang, ‘‘Deceptive opinion spam detection using neuralnetwork,’’ in Proc. 26th Int. Conf. Comput. Linguistics, Tech. Papers COLING, Dec. 2016, pp. 140–150.

[10] X. Wang, K. Liu, and J. Zhao, ‘‘Handling cold-start problem in review spam detection by jointly embedding texts and behaviors,’’ in Proc. 55th Annu. Meeting Assoc. Comput. Linguistics (Long Papers), vol. 1, 2017, pp. 366–376. [Online]. Available: https://www.aclweb.org/anthology/P17- 1034.pdf

[11] C. Yuan, W. Zhou, Q. Ma, S. Lv, J. Han, and S. Hu, ‘‘Learning review representations from user and product level information for spam detection,’’ in Proc. IEEE Int. Conf. Data Mining (ICDM), Nov. 2019, pp. 1444–1449.

[12] Y. Lu, M. Castellanos, U. Dayal, and C. Zhai, ‘‘Automatic construction of a context-aware sentiment lexicon: An optimization approach,’’ in Proc. 20th Int. Conf. World Wide Web - WWW, 2011, pp. 347–356.

[13] G. Ji, S. He, L. Xu, K. Liu, and J. Zhao, ‘‘Knowledge graph embedding via dynamic mapping matrix,’’ in Proc. 53rd Annu. Meeting Assoc. Comput. Linguistics 7th Int. Joint Conf. Natural Lang. Process. (Long Papers), vol. 1, 2015, pp. 687–696. [Online]. Available: https://www. aclweb.org/anthology/P15-1067.pdf

[14] N. Jindal and B. Liu, ‘‘Opinion spam and analysis,’’ in Proc. Int. Conf. Web Search Web Data Mining WSDM, 2008, pp. 219–230.

[15] J. Li, M. Ott, C. Cardie, and E. Hovy, ‘‘Towards a general rule for identifying deceptive opinion spam,’’ in Proc. 52nd Annu. Meeting Assoc. Comput. Linguistics (Long Papers), vol. 1, 2014, pp. 1566–1576. [Online]. Available: https://www.aclweb.org/anthology/P14-1147.pdf.

[16] X. Wang, K. Liu, S. He, and J. Zhao, ‘‘Learning to represent review with tensor decomposition for spam detection,’’ in Proc. Conf. Empirical Methods Natural Lang. Process., 2016, pp. 866–875.

[17] A. Melleng, A. Jurek-Loughrey, and P. Deepak, ‘‘Sentiment and emotion based text representation for fake reviews detection,’’ in Proc. Int. Conf. Recent Adv. Natural Lang. Process. (RANLP), Oct. 2019, pp. 750–757.

[18] M. Z. Asghar, A. Ullah, S. Ahmad, and A. Khan, ‘‘Opinion spam detection framework using hybrid classification scheme,’’ Soft Comput., vol. 24, no. 5, pp. 3475–3498, Mar. 2020.

[19] P. Hajek, A. Barushka, and M. Munk, ‘‘Fake consumer review detection using deep neural networks integrating word embeddings and emotion mining,’’ Neural Comput. Appl., vol. 32, no. 23, pp. 17259–17274, Dec. 2020.

[20] Y. Zhang, G. Lai, M. Zhang, Y. Zhang, Y. Liu, and S. Ma, ‘‘Explicit factor models for explainable recommendation based on phrase-level sentiment analysis,’’ in Proc. 37th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr., Jul. 2014, pp. 83–92.

[21] Y. Jo and A. H. Oh, ‘‘Aspect and sentiment unification model for online review analysis,’’ in Proc. 4th ACM Int. Conf. Web Search Data Mining - WSDM, 2011, pp. 815–824.

[22] Y. Zhang, H. Zhang, M. Zhang, Y. Liu, and S. Ma, ‘‘Do users rate or review?: Boost phrase-level sentiment labeling with review-level sentiment classification,’’ in Proc. 37th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr., Jul. 2014, pp. 1027–1030.

[23] S. Poria, E. Cambria, and A. Gelbukh, ‘‘Aspect extraction for opinion mining with a deep convolutional neural network,’’ Knowl.-Based Syst., vol. 108, pp. 42–49, Sep. 2016.

[24] C. M. Yilmaz and A. O. Durahim, ‘‘SPR2EP: A semi-supervised spam review detection framework,’’ in Proc. IEEE/ACM Int. Conf. Adv. Social Netw. Anal. Mining (ASONAM), Aug. 2018, pp. 306–313.

[25] X. Wang, K. Liu, and J. Zhao, ‘‘Detecting deceptive review spam via attention-based neural networks,’’ in Proc. Nat. CCF Conf. Natural Lang. Process. Chin. Comput., 2017, pp. 866–876.

[26] D. P. Kingma and J. Ba, ‘‘Adam: A method for stochastic optimization,’’ in Proc. Int. Conf. Learn. Represent. (ICLR), 2015, pp. 1–15. [Online]. Available: https://arxiv.org/pdf/1412.6980.pdf

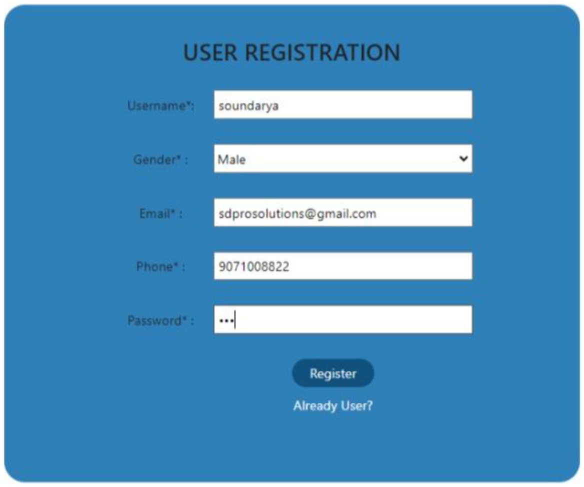
**SOURCE CODE**

```
from flask import Flask, render_template, request, redirect, url_for,flash
import mysql.connector
import os
from datetime import datetime, date
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import sklearn as sk
import pickle
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB, GaussianNB
from sklearn.model_selection import train_test_split

UPLOAD_FOLDER = 'static/file/'
app = Flask(__name__)
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
app.secret_key = b'_5#y2L"F4Q8z\n\xec]/'

mydb =
mysql.connector.connect(host="localhost",user="root",password="",database="ecommerce")
mycursor = mydb.cursor(buffered=True)

model=pickle.load(open('model.pkl','rb'))

@app.route('/')
def index():
    return render_template('index.html')
```

```
@app.route('/user')
def login():
    return render_template('login.html')
@app.route('/admin')
def login1():
    return render_template('login1.html')

@app.route('/reg')
def reg():
    return render_template('user_reg.html')

@app.route('/validate',methods=['POST','GET'])
def validate():
    global uname
    global upass
    if request.method == 'POST':
        uname = request.form.get('username')
        upass = request.form.get('password')
        sql = 'SELECT * FROM `uses` WHERE `name` = %s AND `password` = %s'
        val = (uname, upass)
        mycursor.execute(sql, val)
        result = mycursor.fetchall()
        if result:
            return redirect(url_for('user_dash'))
        # elif uname == 'admin1' and upass == '1234':
        #     return redirect(url_for('admin_dash'))
        else:
            return render_template('login.html', msg = 'Invlid Data')
```

```
@app.route('/ad_validate', methods=['POST', 'GET'])
def ad_validate():
    if request.method == 'POST':
        uname = request.form.get('username')
        upass = request.form.get('password')
        if uname == 'admin1' and upass == '1234':
           return redirect(url_for('admin_dash'))
        else:
            return render_template('login1.html', msg='Invalid')
    elif request.method == 'GET':
        return render_template('login1.html', msg='Please login')

@app.route('/admin_dash')
def admin_dash():
    return render_template('admin_dash.html')

@app.route('/user_dash')
def user_dash():
    sql = 'SELECT * FROM `products`'
    mycursor.execute(sql)
    result = mycursor.fetchall()
    if result:
        return render_template('dashboard.html', data = result)
    else:
        return render_template('dashboard.html', msg = 'No Products')

@app.route('/view')
def view():
    sql = 'SELECT * FROM `products`'
    mycursor.execute(sql)
    result = mycursor.fetchall()
    if result:
        return render_template('view.html', data = result)
    else:
        return render_template('view.html', msg = 'No Data')


@app.route('/add_prod',methods = ['POST','GET'])
def add_prod():
    if request.method == 'POST':
        name = request.form.get('name')
        prod_type = request.form.get('prod_type')
        img = request.files['img']
        price = request.form.get('price')
        desc = request.form.get('desc')
        prod_img = os.path.join(app.config['UPLOAD_FOLDER'], img.filename)
        img.save(prod_img)
        sql = 'INSERT INTO `products` (`name`, `type`, `img`, `price`, `desc`) VALUES (%s, %s,
%s, %s, %s)'
        val = (name, prod_type, prod_img, price, desc)
        mycursor.execute(sql, val)
        mydb.commit()
        return render_template('admin_dash.html', msg = 'Product Added')


@app.route('/userreg',methods = ['POST','GET'])
def userreg():
    if request.method == 'POST':
        name = request.form.get('username')
        gender = request.form.get('gender')
        mail = request.form.get('mail')
        phone = request.form.get('phone')
        password = request.form.get('password')
        sql = "INSERT INTO uses (`name`, `gender`, `mail`, `phone`, `password`) VALUES (%s,
%s, %s, %s, %s)"
        val = (name, gender, mail, phone, password)
        mycursor.execute(sql, val)
        mydb.commit()
        return render_template('login.html')


@app.route('/review_page',methods = ['POST','GET'])
def review_page():
    if request.method == 'POST':
        name = request.form.get('product')
        sql = 'SELECT * FROM `products` WHERE `name` = %s'
        val = (name,)
        mycursor.execute(sql,val)
        result = mycursor.fetchall()
        return render_template('review.html',data = result)


@app.route('/add_review',methods = ['POST','GET'])
def add_review():
    if request.method == 'POST':
        prod_name = request.form.get('prod_name')
        purchase = request.form.get('purchase')
        if purchase is None:
            flash("You are successfully login into the Flask Application")
            login_successful=True
            return render_template('alertmessage.html', login_successful=login_successful)
        print("purchase",purchase)
        review = request.form.get('review')
        rating = request.form.get('rating')
        sql = 'SELECT * FROM `reviews` WHERE `name` = %s AND `review` = %s AND
`prod_name` = %s'
        val = (uname ,review, prod_name)
        mycursor.execute(sql, val)
        result = mycursor.fetchall()

        '''if result:
            status = 'Fake'
            v = 0
        else:
            status = 'Real'
            v = 1'''
        df = pd.read_csv('deceptive-opinion.csv')
        df1 = df[['deceptive', 'text']]
        df1.loc[df1['deceptive'] == 'deceptive', 'deceptive'] = 0
        df1.loc[df1['deceptive'] == 'truthful', 'deceptive'] = 1
        X = df1['text']
        Y = np.asarray(df1['deceptive'], dtype = int)
        X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.3,random_state=109)
        cv = CountVectorizer()
        #cv =MultinomialNB()
        x = cv.fit_transform(X_train)
        y = cv.transform(X_test)
        #message = request.form.get('enteredinfo')
        #data = [message]
        data = [review]
        # Ensure consistent feature dimensions
        #assert X_train.shape[1] == X_test.shape[1], "Training and testing data have different
feature dimensions"

        # # Train MultinomialNB model
        # model = MultinomialNB()
        # model.fit(X_train, y_train)

        # # Predictions
        # y_pred = model.predict(data)

        # # Evaluate model
        # #accuracy = accuracy_score(y_test, y_pred)
        # print("Accuracy:", y_pred)
        vect = cv.transform(data).toarray()
        pred = model.predict(vect)
        print(pred)
        ans = pred[0]
        if ans == 1:
            status = 'Real'
        else:
            status = 'Fake'



        v = 0

        sql = 'SELECT `type` FROM `products` WHERE `name` = %s'
        val = (prod_name,)
        print("prod_name",prod_name)
        mycursor.execute(sql, val)
        result1 = mycursor.fetchone()
        sql = 'INSERT INTO `reviews` (`name`, `prod_name`, `review`, `rating`, `dept`, `status`,
`value`) VALUES (%s, %s, %s, %s, %s, %s, %s)'
        val = (uname, prod_name, review, rating, result1[0], status, v)
        mycursor.execute(sql, val)
        mydb.commit()
        sql = 'SELECT * FROM `reviews`'
        mycursor.execute(sql)
        result2 = mycursor.fetchall()
        if result2:
            my_name = []
            count = []
            for i in result2:
                my_name.append(i[1])
                count.append(i[6])
            x = np.array(my_name)
            y = np.array(count)
            plt.bar(x,y)
            #plt.savefig('report.png')

        return render_template('review.html', msg = 'Review Added')


@app.route('/review')
def review():
    sql = 'SELECT * FROM `reviews`'
    mycursor.execute(sql)
    result = mycursor.fetchall()
    if result:
        return render_template('admin_review.html', data = result)
    else:
        return render_template('admin_review.html', msg = 'No Reviews')


@app.route('/usereview')
def userreview():
    sql = 'SELECT * FROM `reviews`'
    mycursor.execute(sql)
    result = mycursor.fetchall()
    if result:
        return render_template('userreview.html', data = result)
    else:
        return render_template('review.html', msg = 'No Reviews')


if __name__ == '__main__':
    app.run(debug=True,port=3004)
```

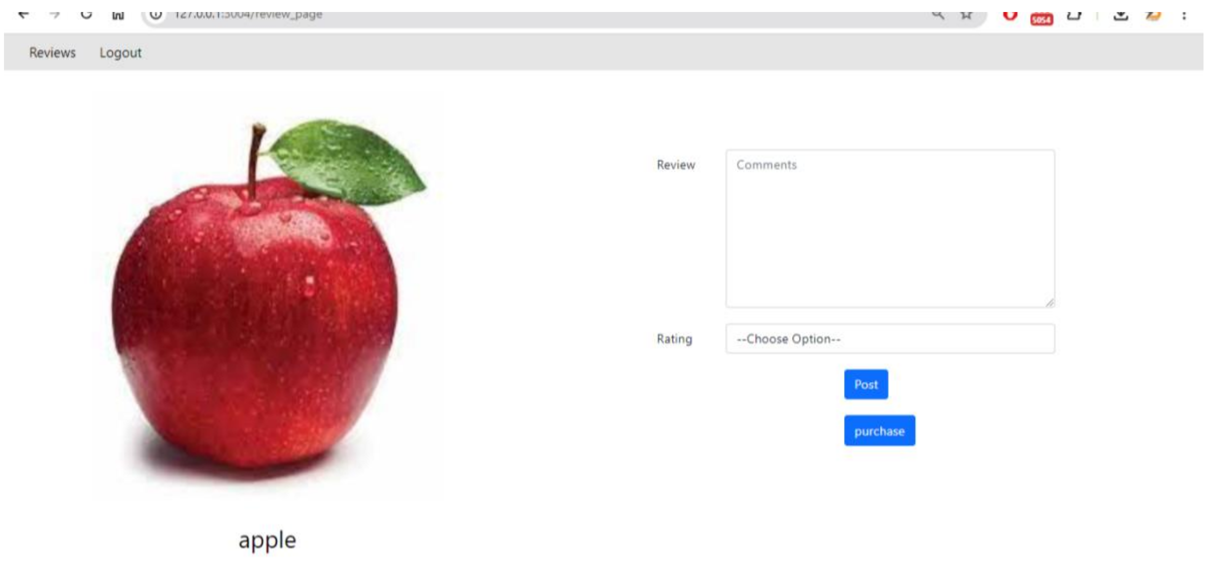
**SCREENSHOT**



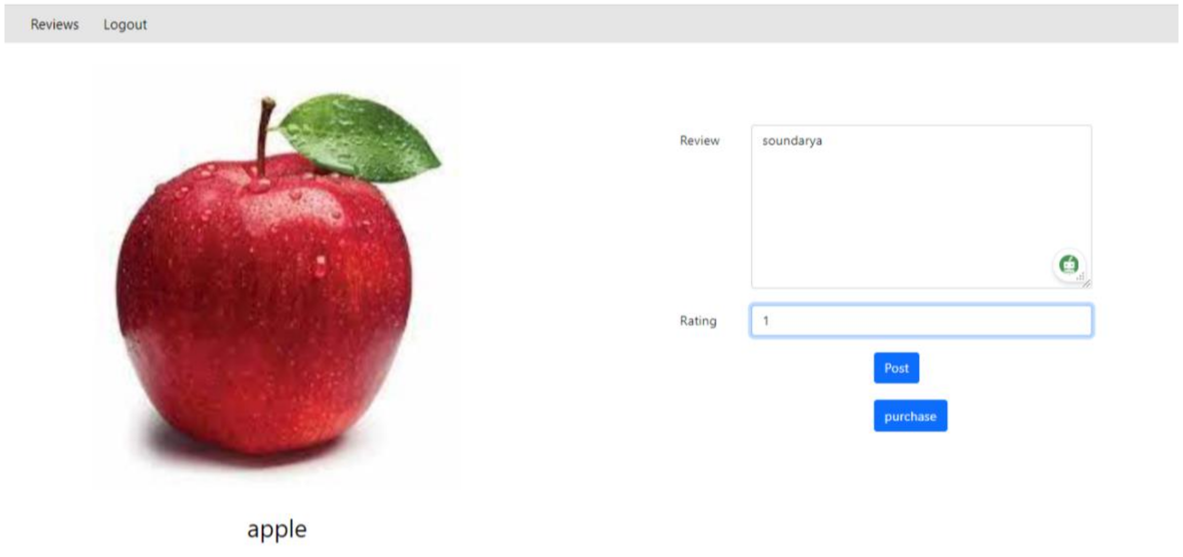
USER REGISTRATION

| | |
|---|---|
| Username*: | soundarya |
| Gender* : | Male |
| Email* : | sdprosolutions@gmail.com |
| Phone* : | 9071008822 |
| Password* : | ••• |

**Figure: user register**



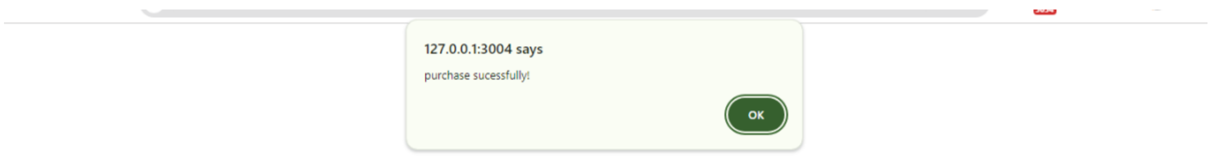
127.0.0.1:5004/review_page



soundarya

1

**Product reviews**


127.0.0.1:3004 says

purchase sucessfully!