

SUBJECTIVE ANSWER EVALUATION USING MACHINE LEARNING

**A Report Submitted
in Partial Fulfillment of the Requirements
for the Degree of**

BACHELOR OF TECHNOLOGY

in

Computer Science and Engineering

by

Madhur Mishra (2101640109005)

Aditi Singh (2001640100018)

Anuskha Tripathi (2001640100063)

Divyanshu Bhargaw (2001640100103)

Shobhit Singh Sengar (2001640100243)

Under the Supervision of

Mr. Rajat Verma

(Assistant Professor)

Pranveer Singh Institute of Technology, Kanpur



**DR. APJ ABDUL KALAM TECHNICAL
UNIVERSITY, LUCKNOW**

May, 2024

DECLARATION

We hereby declare that the work presented in this report entitled “**Subjective Answer Evaluation Using Machine Learning**”, was carried out by us. We have not submitted the matter embodied in this report for the award of any other degree or diploma of any other University or Institute. We have given due credit to the original authors/sources for all the words, ideas, diagrams, graphics, computer programs, experiments, results, that are not my original contribution. We have used quotation marks to identify verbatim sentences and given credit to the original authors/sources. We affirm that no portion of my work is plagiarized, and the experiments and results reported in the report are not manipulated. In the event of a complaint of plagiarism and the manipulation of the experiments and results, we shall be fully responsible and answerable.

Name : Madhur Mishra

Roll. No. : 2101640109005

Signature :

Name : Aditi Singh

Roll. No. : 2001640100018

Signature :

Name : Anushka Tripathi

Roll. No. : 2001640100063

Signature :

Name : Divyanshu Bhargaw

Roll. No. : 2001640100103

Signature :

Name : Shobhit Singh Sengar

Roll. No. : 2001640100243

Signature :

CERTIFICATE

This is to certify that project report entitled “**Subjective Answer Evaluation Using Machine Learning** ” which is submitted by Madhur Mishra , Aditi Singh , Anushka Tripathi , Divyanshu Bhargaw , Shobhit Singh Sengar in partial fulfillment of the requirement for the award of degree B.Tech. in the Department of **Computer Science and Engineering of Pranveer Singh Institute of Technology**, affiliated to **Dr. A.P.J. Abdul Kalam Technical University, Lucknow** is a record of the candidates own work carried out by them under my supervision. The project embodies the result of original work and studies carried out by the students themselves and the contents of the project do not form the basis for the award of any other degree to the candidate or to anybody else.

Signature:

Dr. Vishal Nagar
Dean-CSE
PSIT, Kanpur

Signature:

Mr. Rajat Verma
(Assistant Professor)
CSE Department ,
PSIT , Kanpur

ABSTRACT

Human graders have typically been used to evaluate subjective answers in educational tests, but this is a time-consuming, uneven, and biased procedure. This research investigates the use of machine learning approaches to automate and improve the evaluation of subjective responses, capitalizing on advances in natural language processing (NLP) to achieve high accuracy and consistency. We created and trained models on several datasets of graded replies, mostly using BERT (Bidirectional Encoder Representations from Transformers), to analyze and assess the semantic and contextual accuracy of students' responses. Our findings show that these models can reliably evaluate the relevance, completeness, and accuracy of subjective replies, decreasing biases and enhancing scalability.

Furthermore, the models offer specific feedback, resulting in a more individualized learning experience. The possibility of real-time evaluation, linguistic assistance, and interaction with educational platforms indicates a transformational influence on educational assessment. Future study will focus on improving model training, broadening contextual knowledge, guaranteeing ethical use, and investigating the long-term effects on educational outcomes. This initiative represents a significant step toward more efficient, equitable, and scalable subjective response evaluation in education.

Our technique entails training machine learning models on large datasets of manually graded responses. These models are intended to analyze and evaluate the semantic content and context of students' replies, with an emphasis on criteria like relevance, completeness, and accuracy. The findings show that our models can nearly match human grading performance, with considerable gains in terms of eliminating grading biases and effectively managing massive amounts of data.

ACKNOWLEDGEMENT

It gives us a great sense of pleasure to present the report of the B.Tech. Project undertaken during B.Tech. Final Year. We owe a special debt of gratitude to our project supervisor **Mr. Rajat Verma**, Department of Computer Science and Engineering, Pranveer Singh Institute of Technology, Kanpur for his constant support and guidance throughout the course of our work. His sincere, thoroughness and perseverance have been a constant source of inspiration for us. It is only his cognizant efforts that our endeavors have seen light of the day.

We also take the opportunity to acknowledge the contribution of **Prof. (Dr.) Vishal Nagar**, Dean, Department of Computer Science & Engineering, Pranveer Singh Institute of Technology, Kanpur for his full support and assistance during the development of the project. We also do not like to miss the opportunity to acknowledge the contribution of all faculty members of the department for their kind assistance and cooperation during the development of our project. Last but not the least, we acknowledge our friends for their contribution in the completion of the project.

Signature :

Name : Madhur Mishra

Roll No. : 2101640109005

Signature :

Name : Aditi Singh

Roll No. : 2001640100018

Signature :

Name : Anushka Tripathi

Roll No. : 2001640100063

Signature :

Name : Divyanshu Bhargaw

Roll No. : 2001640100103

Signature :

Name : Shobhit Singh

Sengar

Roll No. : 2001640100243

TABLE OF CONTENTS

Declaration	ii
Certificate	iii
Abstract	iv
Acknowledgement	v
List Of Figures	x
1. CHAPTER 1: INTRODUCTION	1
1.1. Machine Learning	1
1.2. Machine Learning Working	1
1.3. Inferring	2
1.4. Machine Learning Algorithm & Uses	4
1.4.1 Supervised Learning	4
1.4.2 Classification	5
1.4.3 Regression	5
1.5 Machine Learning Algorithm	5
1.6 Challenges and Limitation of Machine Learning	6
1.7 Applications of Machine Learning	7
1.7.1 Augmentation	7
1.7.2 Automation	7
1.7.3 Finance Industry	7
1.7.4 Government Organization	7
1.7.5 Healthcare Industry	7
1.7.6 Marketing	
1.7.6.1 Example of Application of Machine Learning in supply chain	7

1.7.6.2 Example of Machine Learning Google Car	8
1.8 Importance of Machine Learning	8
1.9 Overview	9
1.10 Machine Learning Approaches	9
1.10.1 Supervised Learning	9
1.10.2 Unsupervised Learning	9
1.10.3 Reinforcement Learning	10
1.11 History and Relationship to Other Fields	10
1.12 Data Mining	11
1.13 Optimization	11
1.14 Generalization	11
1.15 Statistics	12
1.16 Theory	12
1.17 Type's of Machine Learning Algorithms	13
1.17.1 Supervised Learning	13
1.17.2 Unsupervised Learning	13
1.17.3 Semi-supervised Learning	14
1.17.4 Reinforcement Learning	14
1.17.5 Self Learning	15
1.17.6 Feature Learning	15
1.17.7 Sparse Dictionary Learning	16
1.17.8 Anomaly Detection	16
1.17.9 Robot Learning	17
1.17.10 Association Rules	17
1.17.11 Artificial Neutral Networks	18

1.17.12 Decision Trees	19
1.17.13 Support Vector Machine	19
1.17.14 Regression Analysis	19
1.17.15 Bayesian Network	20
1.17.16 Genetic Algorithms	20
1.17.17 Training Models	20
1.17.18 Federated Learning	21
2. CHAPTER 2: LITERATURE REVIEW	22
3. CHAPTER 3: SYSTEM ANALYSIS & STUDY	26
3.1 Existing System	26
3.2 Disadvantages of Existing System	26
3.3 Proposed System	26
3.4 Advantages of Proposed System	26
3.5 Feasibility Study	27
3.6 Operational Feasibility	28
4. CHAPTER 4: SYSTEM DESIGN	29
4.1 System Architecture	29
4.2 Data Flow Diagram	29
4.3 Class Diagram	31
4.4 Activity Diagram	32
5. CHAPTER 5: INPUT & OUTPUT DESIGN	33
5.1 Input Design	33
5.2 Objectives	33
5.3 Output Design	34
5.4 System Testing	34

5.5 Types of Testing	35
5.5.1 Unit Testing	35
5.5.2 Integration Testing	36
5.5.3 Functional Test	37
5.5.4 System Test	38
5.5.5 White Box Testing	38
5.5.6 Black Box Testing	38
6. CHAPTER 6: FRONT END & BACK END	39
6.1 Front End	39
6.2 History of Python	39
6.3 Python Feature's	40
6.4 Getting Python	41
6.5 Interactive Mode of Programming	41
6.6 Script Mode of Programming	42
6.7 Flask Framework	43
6.8 Python	47
6.9 Python Highlights	47
6.10 Uses of Python	47
6.11 Back End	48
7. CHAPTER 7: IMPLEMENTATION	51
7.1 Implementation of Code	58
8. CHAPTER 8: RESULT & DISCUSSION	65
9. CHAPTER 9: CONCLUSION & FUTURE SCOPE	67
9.1 Future Scope	68
REFERENCES	70

LIST OF FIGURES

Figure No.	Figure Description	Page No.
Figure 1.1	Learning Phase Model	2
Figure 1.2	Inference Model	2
Figure 1.3	Working of Machine Learning	3
Figure 1.4	Machine Learning Algorithms	4
Figure 1.5	Regarding Algorithms	6
Figure 1.6	Challenges of Machine Learning	6
Figure 1.7	Fields of Machine Learning	21
Figure 4.1	System Architecture	29
Figure 4.2	Data Flow Diagram	30
Figure 4.3	Class Diagram	31
Figure 4.4	Activity Diagram	32
Figure 5.1	System Testing	34
Figure 5.2	Types of Testing	35
Figure 5.3	Unit Testing	36
Figure 5.4	Integration Testing	37
Figure 6.1	Interactive Mode of Programming	42
Figure 6.2	Python Prompt	42
Figure 6.3	Hello Python Prompt	42
Figure 6.4	Successful Attempt	43
Figure 6.5	Path Variable	43
Figure 6.6	Result	43

Figure 6.7	Login Page Window Code	44
Figure 6.8	Post and Get Code	45
Figure 6.9	Login Page Window	45
Figure 6.10	Local Host Page	46
Figure 6.11	Front End Vs Back End	50
Figure 7.1	Style Code Part 1	51
Figure 7.2	Style Code Part 2	51
Figure 7.3	CSS Code Part 1	52
Figure 7.4	CSS Code Part 2	52
Figure 7.5	Python Code Part 1	53
Figure 7.6	Python Code Part 2	53
Figure 7.7	Python Code Part 3	54
Figure 7.8	Python Code Part 4	55
Figure 7.9	Python Code Part 5	56
Figure 7.10	Python Code Part 6	57
Figure 7.11	Python Code Part 7	57
Figure 8.1	Desktop View	65
Figure 8.2	Uploading and Prediction Window	65
Figure 8.3	Login Window	66
Figure 8.4	Score Appearing Window	66

CHAPTER 1

INTRODUCTION

1.1 Machine Learning

Machine Learning is a system of computer algorithms that can learn from example through self-improvement without being explicitly coded by a programmer. Machine learning is a part of artificial Intelligence which combines data with statistical tools to predict an output which can be used to make actionable insights. The breakthrough comes with the idea that a machine can singularly learn from the data (i.e., example) to produce accurate results. Machine learning is closely related to data mining and Bayesian predictive modeling. The machine receives data as input and uses an algorithm to formulate answers.

A typical machine learning tasks are to provide a recommendation. For those who have a Netflix account, all recommendations of movies or series are based on the user's historical data. Tech companies are using unsupervised learning to improve the user experience with personalizing recommendation. Machine learning is also used for a variety of tasks like fraud detection, predictive maintenance, portfolio optimization, automatize task and so on [1].

1.2 Machine Learning Working

Machine learning is the brain where all the learning takes place. The way the machine learns is similar to the human being. Humans learn from experience. The more we know, the more easily we can predict. By analogy, when we face an unknown situation, the likelihood of success is lower than the known situation. Machines are trained the same. To make an accurate prediction, the machine sees an example. When we give the machine a similar example, it can figure out the outcome. However, like a human, if it is feed a previously unseen example, the machine has difficulties to predict. The core objective of machine learning is the **learning** and **inference**. First of all, the machine learns through the discovery of patterns. This discovery is made thanks to the **data**. One crucial part of the data scientist is to choose carefully which data to provide to the machine. The list of attributes used to solve a problem is called a **feature vector**. You can think of a feature vector as a subset of data that is used to tackle a problem. The machine uses some fancy algorithms to simplify the reality and transform this discovery into a **model**. Therefore, the learning stage is used to describe the data and summarize it into a model.[2]

The given below figure 1.1 highlights the working cycle of machine learning.

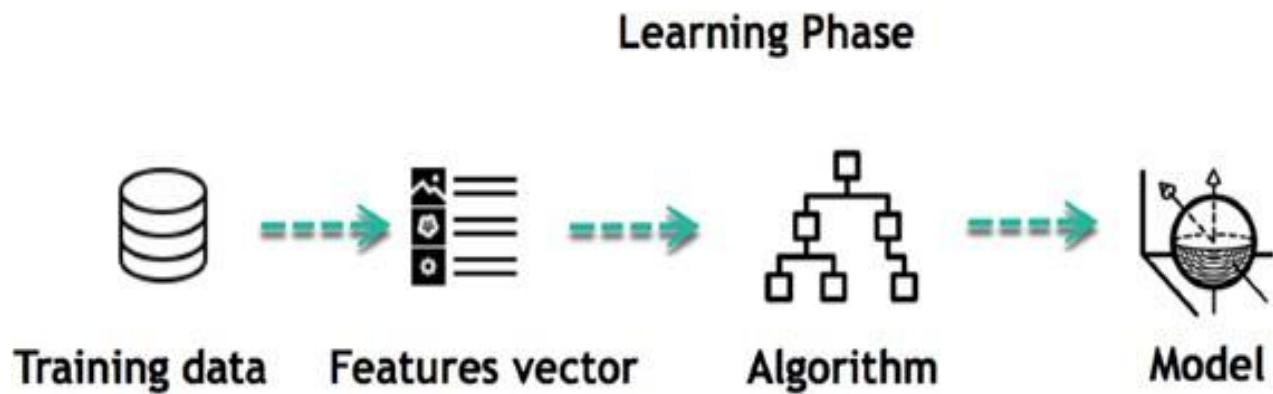


Figure 1.1 Learning Phase Model

1.3 Inferring

When the model is built, it is possible to test how powerful it is on never-seen- before data. The new data are transformed into a features vector, go through the model and give a prediction. This is all the beautiful part of machine learning. There is no need to update the rules or train again the model. You can use the model previously trained to make inference on new data.

The given below figure 1.2 highlights inference model.

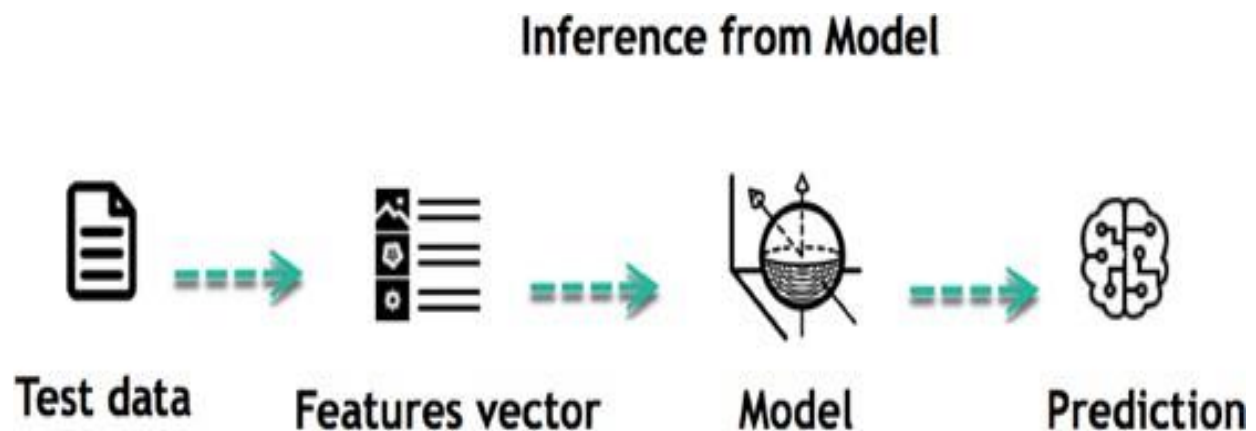


Figure 1.2 Inference Model

The life of Machine Learning programs is straightforward and can be summarized in the following points:

- 1) Define a question
- 2) Collect data
- 3) Visualize data
- 4) Train algorithm
- 5) Test the Algorithm
- 6) Collect feedback
- 7) Refine the algorithm
- 8) Loop 4-7 until the results are satisfying
- 9) Use the model to make a prediction

The given below figure 1.3 highlights all relevant working of Machine Learning.

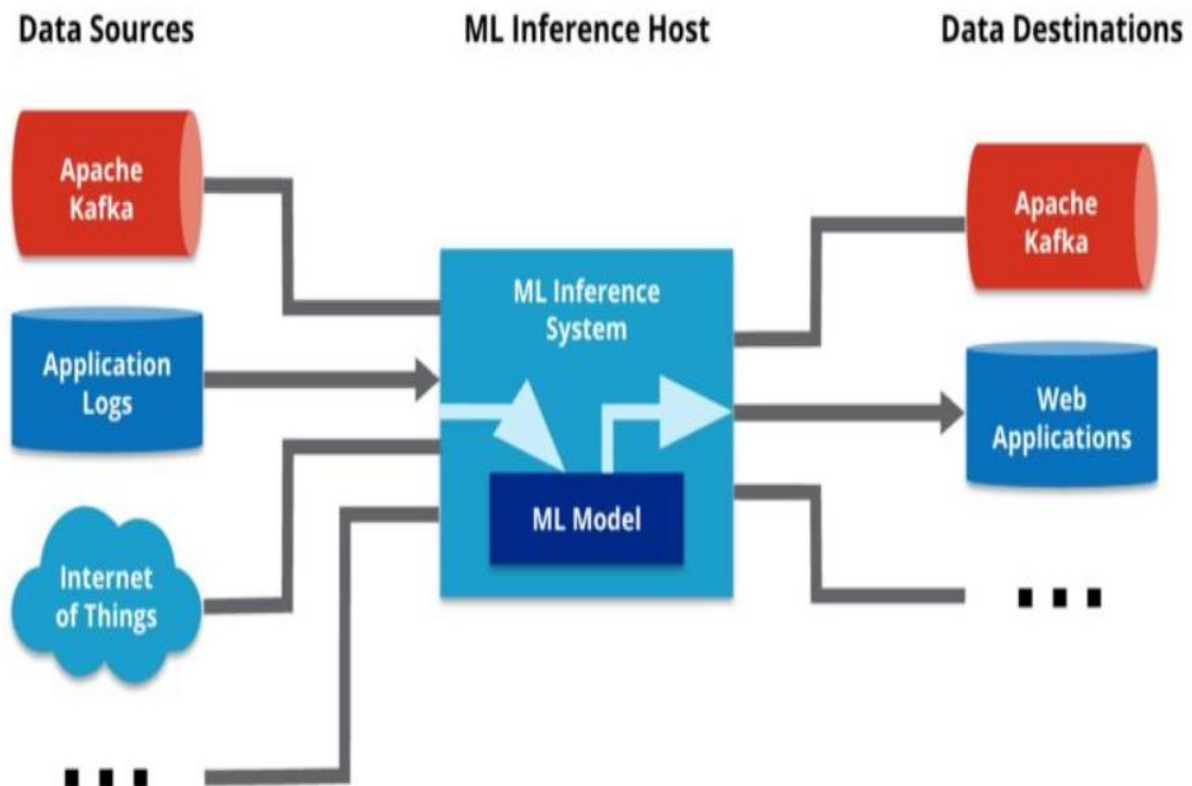


Figure 1.3 Relevant Working of Machine Learning

1.4 Machine Learning Algorithms and their uses

Machine learning can be grouped into two broad learning tasks: Supervised and Unsupervised. There are many other algorithms which are given below in this figure.

The given below figure 1.4 highlights Machine Learning Algorithms.

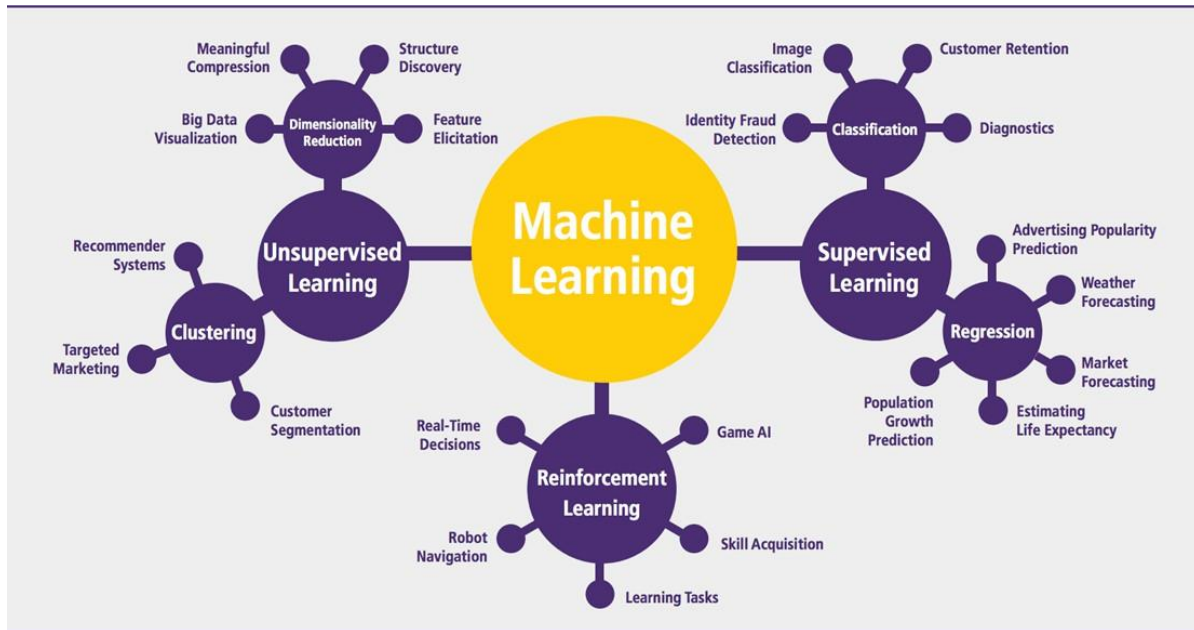


Figure 1.4 Machine Learning Algorithms

1.4.1 Supervised learning

An algorithm uses training data and feedback from humans to learn the relationship of given inputs to a given output. For instance, a practitioner can use marketing expense and weather forecast as input data to predict the sales of cans.

You can use supervised learning when the output data is known. The algorithm will predict new data.

There are two categories of supervised learning

- 1) Classification task
- 2) Regression task

1.4.2 Classification

Imagine you want to predict the gender of a customer for a commercial. You will start gathering data on the height, weight, job, salary, purchasing basket, etc. from your customer database. You know the gender of each of your customer, it can only be male or female. The objective of the classifier will be to assign a probability of being a male or a female (i.e., the label) based on the information (i.e., features you have collected). When the model learned how to recognize male or female, you can use new data to make a prediction. For instance, you just got new information from an unknown customer, and you want to know if it is a male or female. If the classifier predicts male = 70%, it means the algorithm is sure at 70% that this customer is a male, and 30% it is a female.

The label can be of two or more classes. The above Machine learning example has only two classes, but if a classifier needs to predict object, it has dozens of classes (e.g., glass, table, shoes, etc. each object represents a class).[3]

1.4.3 Regression

When the output is a continuous value, the task is a regression. For instance, a financial analyst may need to forecast the value of a stock based on a range of feature like equity, previous stock performances, macroeconomics index. The system will be trained to estimate the price of the stocks with the lowest possible error.

1.5 Machine Learning (ML) algorithm

There are plenty of machine learning algorithms. The choice of the algorithm is based on the objective. In the Machine learning example below, the task is to predict the type of flower among the three varieties. The predictions are based on the length and the width of the petal. The picture depicts the results of ten different algorithms. The picture on the top left is the dataset. The data is classified into three categories: red, light blue and dark blue. There are some groupings. For instance, from the second image, everything in the upper left belongs to the red category, in the middle part, there is a mixture of uncertainty and light blue while the bottom corresponds to the dark category. The other images show different algorithms and how they try to classified the data.

The given below figure1.5 highlights Algorithms Regarding Machine Learning and the generation.

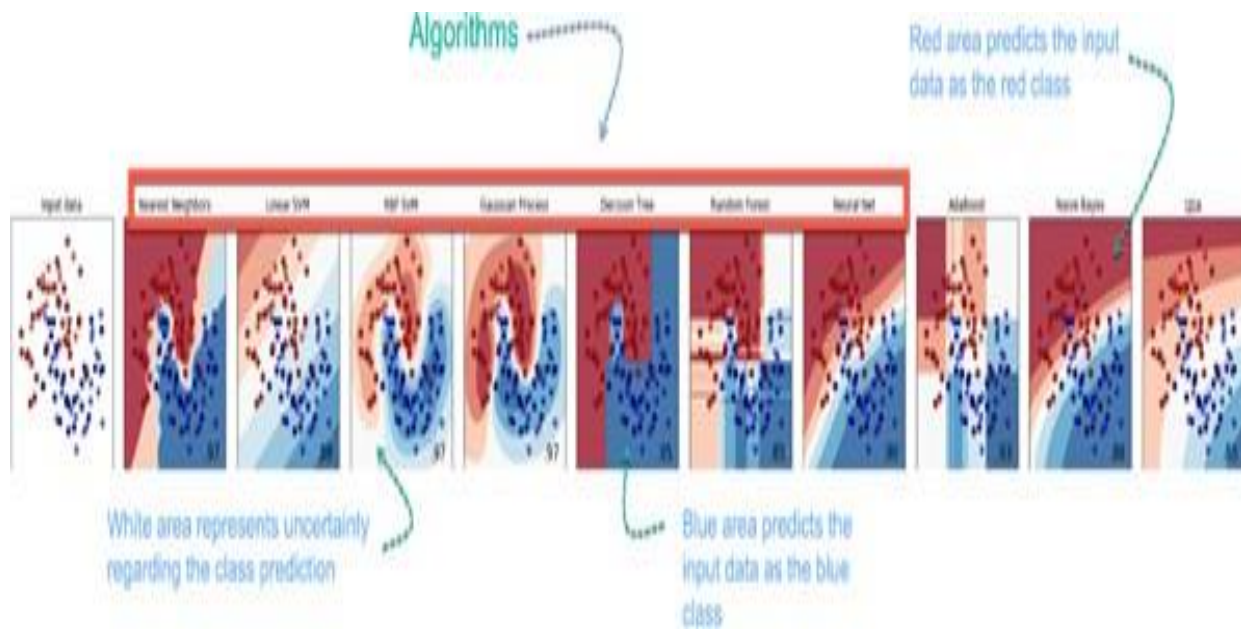


Figure 1.5 Regarding Algorithms

1.6 Challenges and Limitations of Machine Learning

The primary challenge of machine learning is the lack of data or the diversity in the dataset. A machine cannot learn if there is no data available. Besides, a dataset with a lack of diversity gives the machine a hard time. A machine needs to have heterogeneity to learn meaningful insight. It is rare that an algorithm can extract information when there are no or few variations. It is recommended to have at least 20 observations per group to help the machine learn. This constraint leads to poor evaluation and prediction. The given below figure 1.6 highlights challenges of ML.



Figure 1.6 Challenges of Machine Learning

1.7 Applications of machine learning

1.7.1 Augmentation

- 1) Machine learning, which assists humans with their day-to-day tasks, personally or commercially without having complete control of the output. Such machine learning is used in different ways such as Virtual Assistant, Data analysis, software solutions. The primary user is to reduce errors due to human bias.

1.7.2 Automation

- 1) Machine learning, which works entirely autonomously in any field without the need for any human intervention. For example, robots performing the essential process steps in manufacturing plants.

1.7.3 Finance Industry

- 1) Machine learning is growing in popularity in the finance industry. Banks are mainly using ML to find patterns inside the data but also to prevent fraud.

1.7.4 Government Organization

- 1) The government makes use of ML to manage public safety and utilities. Take the example of China with the massive face recognition. The government uses Artificial intelligence to prevent jaywalker.

1.7.5 Healthcare Industry

- 1) Healthcare was one of the first industry to use machine learning with image detection.

1.7.6 Marketing

- 1) Broad use of AI is done in marketing thanks to abundant access to data. Before the age of mass data, researchers develop advanced mathematical tools like Bayesian analysis to estimate the value of a customer. With the boom of data, marketing department relies on AI to optimize the customer relationship and marketing campaign.

1.7.6.1 Example of application of Machine Learning in Supply Chain

Machine learning gives terrific results for visual pattern recognition, opening up many potential applications in physical inspection and maintenance across the entire supply chain

network. Unsupervised learning can quickly search for comparable patterns in the diverse dataset. In turn, the machine can perform quality inspection throughout the logistics hub, shipment with damage and wear. For instance, IBM's Watson platform can determine shipping container damage. Watson combines visual and systems-based data to track, report and make recommendations in real-time.

1.7.6.2 Example of Machine Learning Google Car

Everybody knows the Google car. The car is full of lasers on the roof which are telling it where it is regarding the surrounding area. It has radar in the front, which is informing the car of the speed and motion of all the cars around it. It uses all of that data to figure out not only how to drive the car but also to figure out and predict what potential drivers around the car are going to do. What's impressive is that the car is processing almost a gigabyte a second of data.[4]

1.8 Importance of Machine Learning

Machine learning is the best tool so far to analyze, understand and identify a pattern in the data. One of the main ideas behind machine learning is that the computer can be trained to automate tasks that would be exhaustive or impossible for a human being. The clear breach from the traditional analysis is that machine learning can take decisions with minimal human intervention. Take the following example for this ML tutorial; a retail agent can estimate the price of a house based on his own experience and his knowledge of the market.

A machine can be trained to translate the knowledge of an expert into features. The features are all the characteristics of a house, neighborhood, economic environment, etc. that make the price difference. For the expert, it took him probably some years to master the art of estimate the price of a house. His expertise is getting better and better after each sale.

For the machine, it takes millions of data, (i.e., example) to master this art. At the very beginning of its learning, the machine makes a mistake, somehow like the junior salesman. Once the machine sees all the example, it got enough knowledge to make its estimation. At the same time, with incredible accuracy. The machine is also able to adjust its mistake accordingly. Most of the big company have understood the value of machine learning and holding data. McKinsey have estimated that the value of analytics ranges from \$9.5 trillion to \$15.4 trillion while \$5 to 7 trillion can be attributed to the most advanced AI techniques. Machine learning (ML) is the

study of computer algorithms that improve automatically through experience. It is seen as a part of artificial intelligence. Machine learning algorithms build a model based on sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to do so. Machine learning algorithms are used in a wide variety of applications, such as email filtering and computer vision, where it is difficult or unfeasible to develop conventional algorithms to perform the needed tasks.[5]

1.9 Overview

Machine learning involves computers discovering how they can perform tasks without being explicitly programmed to do so. It involves computers learning from data provided so that they carry out certain tasks. For simple tasks assigned to computers, it is possible to program algorithms telling the machine how to execute all steps required to solve the problem at hand; on the computer's part, no learning is needed. For more advanced tasks, it can be challenging for a human to manually create the needed algorithms. In practice, it can turn out to be more effective to help the machine develop its own algorithm, rather than having human programmers specify every needed step. The discipline of machine learning employs various approaches to teach computers to accomplish tasks where no fully satisfactory algorithm is available. In cases where vast numbers of potential answers exist, one approach is to label some of the correct answers as valid. This can then be used as training data for the computer to improve the algorithm(s) it uses to determine correct answers. For example, to train a system for the task of digital character recognition, the MNIST dataset of handwritten digits has often been used.[6]

1.10 Machine Learning Approaches

Machine learning approaches are traditionally divided into three broad categories, depending on the nature of the "signal" or "feedback" available to the learning system.

1.10.1 Supervised Learning

The computer is presented with example inputs and their desired outputs, given by a "teacher", and the goal is to learn a general rule that maps inputs to outputs.

1.10.2 Unsupervised Learning

No labels are given to the learning algorithm, leaving it on its own to find structure in its

input. Unsupervised learning can be a goal in itself (discovering hidden patterns in data) or a means towards an end (feature learning).

1.10.3 Reinforcement Learning

A computer program interacts with a dynamic environment in which it must perform a certain goal (such as driving a vehicle or playing a game against an opponent). As it navigates its problem space, the program is provided feedback that's analogous to rewards, which it tries to maximize.

1.11 History and Relationships to Other Fields

The term machine learning was coined in 1959 by Arthur Samuel, an American IBMer and pioneer in the field of computer gaming and artificial intelligence. A representative book of the machine learning research during the 1960s was the Nilsson's book on Learning Machines, dealing mostly with machine learning for pattern classification. Interest related to pattern recognition continued into the 1970s, as described by Duda and Hart in 1973. In 1981 a report was given on using teaching strategies so that a neural network learns to recognize 40 characters (26 letters, 10 digits, and 4 special symbols) from a computer terminal. Tom M. Mitchell provided a widely quoted, more formal definition of the algorithms studied in the machine learning field: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T , as measured by P , improves with experience," This definition of the tasks in which machine learning is concerned offers a fundamentally operational definition rather than defining the field in cognitive terms. This follows Alan Turing's proposal in his paper "Computing Machinery and Intelligence", in which the question "Can machines think?" is replaced with the question "Can machines do what we (as thinking entities) can do?". Modern day machine learning has two objectives, one is to classify data based on models which have been developed, the other purpose is to make predictions for future outcomes based on these models. A hypothetical algorithm specific to classifying data may use computer vision of moles coupled with supervised learning in order to train it to classify the cancerous moles. Whereas, a machine learning algorithm for stock trading may inform the trader of future potential predictions.[7]

1.12 Data mining

Machine learning and data mining often employ the same methods and overlap significantly, but while machine learning focuses on prediction, based on known properties learned from the training data, data mining focuses on the discovery of (previously) unknown properties in the data (this is the analysis step of knowledge discovery in databases). Data mining uses many machine learning methods, but with different goals; on the other hand, machine learning also employs data mining methods as "unsupervised learning" or as a preprocessing step to improve learner accuracy. Much of the confusion between these two research communities (which do often have separate conferences and separate journals, ECML PKDD being a major exception) comes from the basic assumptions they work with: in machine learning, performance is usually evaluated with respect to the ability to reproduce known knowledge, while in knowledge discovery and data mining (KDD) the key task is the discovery of previously unknown knowledge. Evaluated with respect to known knowledge, an uninformed (unsupervised) method will easily be outperformed by other supervised methods, while in a typical KDD task, supervised methods cannot be used due to the unavailability of training data.[8]

1.13 Optimization

Machine learning also has intimate ties to optimization: many learning problems are formulated as minimization of some loss function on a training set of examples. Loss functions express the discrepancy between the predictions of the model being trained and the actual problem instances (for example, in classification, one wants to assign a label to instances, and models are trained to correctly predict the pre- assigned labels of a set of examples).[9]

1.14 Generalization

The difference between optimization and machine learning arises from the goal of generalization: while optimization algorithms can minimize the loss on a training set, machine learning is concerned with minimizing the loss on unseen samples. Characterizing the generalization of various learning algorithms is an active topic of current research, especially for deep learning algorithms.[10]

1.15 Statistics

Machine learning and statistics are closely related fields in terms of methods, but distinct in their principal goal: statistics draws population inferences from a sample, while machine learning finds generalizable predictive patterns. According to Michael I. Jordan, the idea's of machine learning, from methodological principles to theoretical tools, have had a long pre-history in statistics. He also suggested the term data science as a placeholder to call the overall field. Leo Breman distinguished two statistical modeling paradigms: data model and algorithmic model, wherein "algorithmic model" means more or less the machine learning algorithms like Random Forest. Some statisticians have adopted methods from machine learning, leading to a combined field that they call statistical learning.[11]

1.16 Theory

A core objective of a learner is to generalize from its experience. Generalization in this context is the ability of a learning machine to perform accurately on new, unseen examples/tasks after having experienced a learning data set. The training examples come from some generally unknown probability distribution (considered representative of the space of occurrences) and the learner has to build a general model about this space that enables it to produce sufficiently accurate predictions in new cases. The computational analysis of machine learning algorithms and their performance is a branch of theoretical computer science known as computational learning theory. Because training sets are finite and the future is uncertain, learning theory usually does not yield guarantees of the performance of algorithms. Instead, probabilistic bounds on the performance are quite common. The bias-variance decomposition is one way to quantify generalization error. For the best performance in the context of generalization, the complexity of the hypothesis should match the complexity of the function underlying the data. If the hypothesis is less complex than the function, then the model has under fitted the data. If the complexity of the model is increased in response, then the training error decreases. But if the hypothesis is too complex, then the model is subject to overfitting and generalization will be poorer. In addition to performance bounds, learning theorists study the time complexity and feasibility of learning. In computational learning theory, a computation is considered feasible if it can be done in polynomial time. There are two kinds of time complexity results. Positive results show that a certain class of functions can be learned in polynomial time. Negative results show that certain classes cannot be learned in

polynomial time.[12]

1.17 Type's of Machine Learning Algorithms

The types of machine learning algorithms differ in their approach, the type of data they input and output, and the type of task or problem that they are intended to solve.

1.17.1 Supervised Learning

A support vector machine is a supervised learning model that divides the data into regions separated by a linear boundary. Here, the linear boundary divides the black circles from the white. Supervised learning algorithms build a mathematical model of a set of data that contains both the inputs and the desired outputs. The data is known as training data, and consists of a set of training examples. Each training example has one or more inputs and the desired output, also known as a supervisory signal. In the mathematical model, each training example is represented by an array or vector, sometimes called a feature vector, and the training data is represented by a matrix. Through iterative optimization of an objective function, supervised learning algorithms learn a function that can be used to predict the output associated with new inputs. An optimal function will allow the algorithm to correctly determine the output for inputs that were not a part of the training data. An algorithm that improves the accuracy of its outputs or predictions over time is said to have learned to perform that task. Types of supervised learning algorithms include active learning, classification and regression. Classification algorithms are used when the outputs are restricted to a limited set of values, and regression algorithms are used when the outputs may have any numerical value within a range. As an example, for a classification algorithm that filters emails, the input would be an incoming email, and the output would be the name of the folder in which to file the email.[13]

1.17.2 Unsupervised Learning

Unsupervised learning algorithms take a set of data that contains only inputs, and find structure in the data, like grouping or clustering of data points. The algorithms, therefore, learn from test data that has not been labeled, classified or categorized. Instead of responding to feedback, unsupervised learning algorithms identify commonalities in the data and react based on the presence or absence of such commonalities in each new piece of data. A central application of unsupervised learning is in the field of density estimation in statistics, such as

finding the probability density function. Though unsupervised learning encompasses other domains involving summarizing and explaining data features. Cluster analysis is the assignment of a set of observations into subsets (called clusters) so that observations within the same cluster are similar according to one or more predesignated criteria, while observations drawn from different clusters are dissimilar. Different clustering techniques make different assumptions on the structure of the data, often defined by some similarity metric and evaluated, for example, by internal compactness, or the similarity between members of the same cluster, and separation, the difference between clusters. Other methods are based on estimated density and graph connectivity.[14]

1.17.3 Semi-supervised Learning

Semi-supervised learning falls between unsupervised learning (without any labeled training data) and supervised learning (with completely labeled training data). Some of the training examples are missing training labels, yet many machine- learning researchers have found that unlabeled data, when used in conjunction with a small amount of labeled data, can produce a considerable improvement in learning accuracy. In weakly supervised learning, the training labels are noisy, limited, or imprecise; however, these labels are often cheaper to obtain, resulting in larger effective training sets.[15]

1.17.4 Reinforcement Learning

Reinforcement learning is an area of machine learning concerned with how software agents ought to take actions in an environment so as to maximize some notion of cumulative reward. Due to its generality, the field is studied in many other disciplines, such as game theory, control theory, operations research, information theory, simulation-based optimization, multi-agent systems, swarm intelligence, statistics and genetic algorithms. In machine learning, the environment is typically represented as a Markov decision process (MDP). Many reinforcement learning algorithms use dynamic programming techniques. Reinforcement learning algorithms do not assume knowledge of an exact mathematical model of the MDP, and are used when exact models are infeasible. Reinforcement learning algorithms are used in autonomous vehicles or in learning to play a game against a human opponent.

1.17.5 Self Learning

Self-learning as a machine learning paradigm was introduced in 1982 along with a neural network capable of self-learning named crossbar adaptive array (CAA). It is a learning with no external rewards and no external teacher advice. The CAA self-learning algorithm computes, in a crossbar fashion, both decisions about actions and emotions (feelings) about consequence situations. The system is driven by the interaction between cognition and emotion. The self-learning algorithm updates a memory matrix $W = \|w(a,s)\|$ such that in each iteration executes the following machine learning routine.[16]

1.17.6 Feature Learning

Several learning algorithms aim at discovering better representations of the inputs provided during training. Classic examples include principal components analysis and cluster analysis. Feature learning algorithms, also called representation learning algorithms, often attempt to preserve the information in their input but also transform it in a way that makes it useful, often as a pre-processing step before performing classification or predictions. This technique allows reconstruction of the inputs coming from the unknown data-generating distribution, while not being necessarily faithful to configurations that are implausible under that distribution. This replaces manual feature engineering, and allows a machine to both learn the features and use them to perform a specific task. Feature learning can be either supervised or unsupervised. In supervised feature learning, features are learned using labeled input data. Examples include artificial neural networks, multilayer perceptrons, and supervised dictionary learning. In unsupervised feature learning, features are learned with unlabeled input data. Examples include dictionary learning, independent component analysis, autoencoders, matrix factorization and various forms of clustering. Manifold learning algorithms attempt to do so under the constraint that the learned representation is low-dimensional. Sparse coding algorithms attempt to do so under the constraint that the learned representation is sparse, meaning that the mathematical model has many zeros. Multilinear subspace learning algorithms aim to learn low-dimensional representations directly from tensor representations for multidimensional data, without reshaping them into higher-dimensional vectors. Deep learning algorithms discover multiple levels of representation, or a hierarchy of features, with higher-level, more abstract features defined in terms of (or generating) lower-level features. It has been argued that an intelligent machine is one that learns a

representation that disentangles the underlying factors of variation that explain the observed.[17]

1.17.7 Sparse Dictionary Learning

Sparse dictionary learning is a feature learning method where a training example is represented as a linear combination of basis functions, and is assumed to be a sparse matrix. The method is strongly NP-hard and difficult to solve approximately. A popular heuristic method for sparse dictionary learning is the K-SVD algorithm. Sparse dictionary learning has been applied in several contexts. In classification, the problem is to determine the class to which a previously unseen training example belongs. For a dictionary where each class has already been built, a new training example is associated with the class that is best sparsely represented by the corresponding dictionary. Sparse dictionary learning has also been applied in image de-noising. The key idea is that a clean image patch can be sparsely represented by an image dictionary, but the noise cannot.

1.17.8 Anomaly Detection

In data mining, anomaly detection, also known as outlier detection, is the identification of rare items, events or observations which raise suspicions by differing significantly from the majority of the data. Typically, the anomalous items represent an issue such as bank fraud, a structural defect, medical problems or errors in a text. Anomalies are referred to as outliers, novelties, noise, deviations and exceptions. In particular, in the context of abuse and network intrusion detection, the interesting objects are often not rare objects, but unexpected bursts of inactivity. This pattern does not adhere to the common statistical definition of an outlier as a rare object, and many outlier detection methods (in particular, unsupervised algorithms) will fail on such data unless it has been aggregated appropriately. Instead, a cluster analysis algorithm may be able to detect the micro-clusters formed by these patterns. Three broad categories of anomaly detection techniques exist. Unsupervised anomaly detection techniques detect anomalies in an unlabeled test data set under the assumption that the majority of the instances in the data set are normal, by looking for instances that seem to fit least to the remainder of the data set. Supervised anomaly detection techniques require a data set that has been labeled as "normal" and "abnormal" and involves training a classifier (the key difference to many other statistical classification problems is the inherently unbalanced nature of outlier detection). Semi-supervised anomaly detection techniques construct a model representing normal behavior from a given

normal training data set and then test the likelihood of a test instance to be generated by the model.

1.17.9 Robot Learning

In developmental robotics, robot learning algorithms generate their own sequences of learning experiences, also known as a curriculum, to cumulatively acquire new skills through self-guided exploration and social interaction with humans. These robots use guidance mechanisms such as active learning, maturation, motor synergies and imitation.

1.17.10 Association Rules

Association rule learning is a rule-based machine learning method for discovering relationships between variables in large databases. It is intended to identify strong rules discovered in databases using some measure of "interestingness". Rule-based machine learning is a general term for any machine learning method that identifies, learns, or evolves "rules" to store, manipulate or apply knowledge. The defining characteristic of a rule-based machine learning algorithm is the identification and utilization of a set of relational rules that collectively represent the knowledge captured by the system. This is in contrast to other machine learning algorithms that commonly identify a singular model that can be universally applied to any instance in order to make a prediction. Rule-based machine learning approaches include learning classifier systems, association rule learning, and artificial immune systems. Based on the concept of strong rules, Rakesh Agrawal, Tomasz Imieliński and Arun Swami introduced association rules for discovering regularities between products in large-scale transaction data recorded by point-of-sale (POS) systems in supermarket. with a learning component, performing either supervised learning, reinforcement learning, or unsupervised learning. They seek to identify a set of context-dependent rules that collectively store and apply knowledge in a piecewise manner in order to make predictions. Inductive logic programming (ILP) is an approach to rule-learning using logic programming as a uniform representation for input examples, background knowledge, and hypotheses. Given an encoding of the known background knowledge and a set of examples represented as a logical database of facts, an ILP system will derive a hypothesized logic program that entails all positive and no negative examples. Inductive programming is a related field that considers any kind of programming language for representing hypotheses (and not only logic programming), such as functional programs. Inductive logic programming is particularly useful in bioinformatics and natural language processing. Gordon Plotkin and Ehud Shapiro laid the

initial theoretical foundation for inductive machine learning in a logical setting. Shapiro built their first implementation (Model Inference System) in 1981: a Prolog program that inductively inferred logic programs from positive and negative examples. The term inductive here refers to philosophical induction, suggesting a theory to explain observed facts, rather than mathematical induction, proving a property for all members of a well-ordered set.[18]

1.17.11 Artificial Neural Networks

An artificial neural network is an interconnected group of nodes, akin to the vast network of neurons in a brain. Here, each circular node represents an artificial neuron and an arrow represents a connection from the output of one artificial neuron to the input of another. Artificial neural networks (ANNs), or connectionist systems, are computing systems vaguely inspired by the biological neural networks that constitute animal brains. Such systems "learn" to perform tasks by considering examples, generally without being programmed with any task-specific rules. An ANN is a model based on a collection of connected units or nodes called "artificial neurons", which loosely model the neurons in a biological brain. Each connection, like the synapses in a biological brain, can transmit information, a "signal", from one artificial neuron to another. An artificial neuron that receives a signal can process it and then signal additional artificial neurons connected to it. In common ANN implementations, the signal at a connection between artificial neurons is a real number, and the output of each artificial neuron is computed by some non-linear function of the sum of its inputs. The connections between artificial neurons are called "edges". Artificial neurons and edges typically have a weight that adjusts as learning proceeds. The weight increases or decreases the strength of the signal at a connection. Artificial neurons may have a threshold such that the signal is only sent if the aggregate signal crosses that threshold. Typically, artificial neurons are aggregated into layers. Different layers may perform different kinds of transformations on their inputs. Signals travel from the first layer (the input layer) to the last layer (the output layer), possibly after traversing the layers multiple times. The original goal of the ANN approach was to solve problems in the same way that a human brain would. However, over time, attention moved to performing specific tasks, leading to deviations from biology. Artificial neural networks have been used on a variety of tasks, including computer vision, speech recognition, machine translation, social network filtering, playing board and video games and medical diagnosis. Deep learning consists of multiple hidden layers in an artificial neural network.

This approach tries to model the way the human brain processes light and sound into vision and hearing. Some successful applications of deep learning are computer vision and speech recognition.[19]

1.17.12 Decision Trees

Decision tree learning uses a decision tree as a predictive model to go from observations about an item (represented in the branches) to conclusions about the item's target value (represented in the leaves). It is one of the predictive modeling approaches used in statistics, data mining, and machine learning. Tree models where the target variable can take a discrete set of values are called classification trees; in these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels. Decision trees where the target variable can take continuous values (typically real numbers) are called regression trees. In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making. In data mining, a decision tree describes data, but the resulting classification tree can be an input for decision making.[20]

1.17.13 Support Vector Machines

Support vector machines (SVMs), also known as support vector networks, are a set of related supervised learning methods used for classification and regression. Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that predicts whether a new example falls into one category or the other. An SVM training algorithm is a non- probabilistic, binary, linear classifier, although methods such as Platt scaling exist to use SVM in a probabilistic classification setting. In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.[21]

1.17.14 Regression Analysis

Regression analysis encompasses a large variety of statistical methods to estimate the relationship between input variables and their associated features. Its most common form is linear regression, where a single line is drawn to best fit the given data according to a mathematical criterion such as ordinary least squares. The latter is often extended by regularization

(mathematics) methods to mitigate overfitting and bias, as in ridge regression. When dealing with non-linear problems, go-to models include polynomial regression (for example, used for trendline fitting in Microsoft Excel, logistic regression (often used in statistical classification) or even kernel regression, which introduces non-linearity by taking advantage of the kernel trick to implicitly map input variables to higher-dimensional space.[22]

1.17.15 Bayesian Network

A simple Bayesian Network. Rain influences whether the sprinkler is activated, and both rain and the sprinkler influence whether the grass is wet. A Bayesian network, belief network, or directed acyclic graphical model is a probabilistic graphical model that represents a set of random variables and their conditional independence with a directed acyclic graph (DAG). For example, a Bayesian network could represent the probabilistic relationships between diseases and symptoms. Given symptoms, the network can be used to compute the probabilities of the presence of various diseases. Efficient algorithms exist that perform inference and learning. Bayesian networks that model sequences of variables, like speech signals or protein sequences, are called dynamic Bayesian networks. Generalizations of Bayesian networks that can represent and solve decision problems under uncertainty are called influence diagrams.[23]

1.17.16 Genetic Algorithms

A genetic algorithm (GA) is a search algorithm and heuristic technique that mimics the process of natural selection, using methods such as mutation and crossover to generate new genotypes in the hope of finding good solutions to a given problem. In machine learning, genetic algorithms were used in the 1980s and 1990s. Conversely, machine learning techniques have been used to improve the performance of genetic and evolutionary algorithms.

1.17.17 Training Models

Usually, machine learning models require a lot of data in order for them to perform well. Usually, when training a machine learning model, one needs to collect a large, representative sample of data from a training set. Data from the training set can be as varied as a corpus of text, a collection of images, and data collected from individual users of a service. Overfitting is something to watch out for when training a machine learning model. Trained models derived from

biased data can result in skewed or undesired predictions. Algorithmic bias is a potential result from data not fully prepared for training.[24]

1.17.18 Federated Learning

Federated learning is an adapted form of distributed artificial intelligence to training machine learning models that decentralizes the training process, allowing for users' privacy to be maintained by not needing to send their data to a centralized server. This also increases efficiency by decentralizing the training process to many devices. For example, Google board uses federated machine learning to train search query prediction models on users' mobile phones without having to send individual searches back to Google.[25]

The given below figure 1.7 highlights the various fields of Machine Learning.

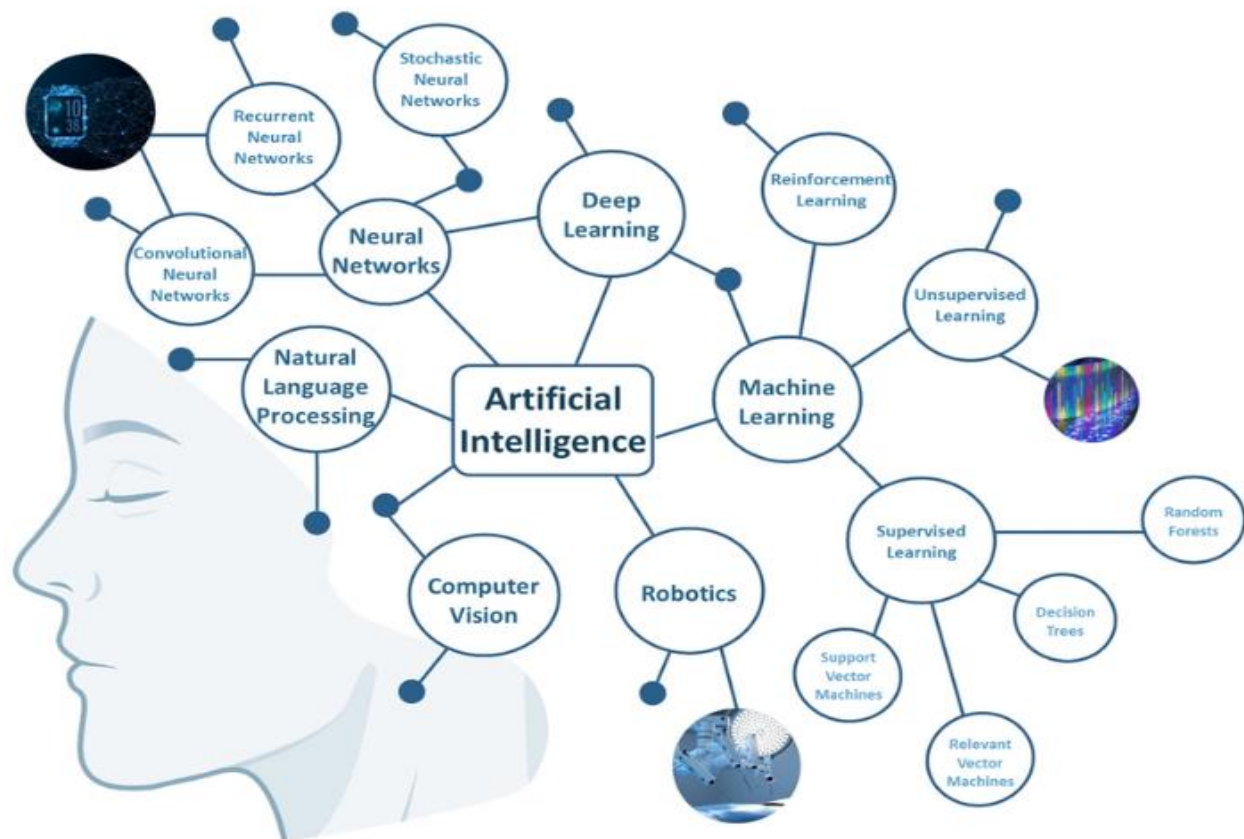


Figure 1.7 Fields of Machine Learning

CHAPTER 2

LITERATURE REVIEW

In recent years the data mining is data analyzing techniques that used to analyze crime data previously stored from various sources to find patterns and trends in crimes. In additional, it can be applied to increase efficiency in solving the crimes faster and also can be applied to automatically notify the crimes. However, there are many data mining techniques. In order to increase efficiency of crime detection, it is necessary to select the data mining techniques suitably. This paper reviews the literatures on various data mining applications, especially applications that applied to solve the crimes. Survey also throws light on research gaps and challenges of crime data mining. In additional to that, this paper provides insight about the data mining for finding the patterns and trends in crime to be used appropriately and to be a help for beginners in the research of crime data mining.

1) , “Subjective answer evaluation using machine learning

AUTHORS: P. Patil, S. Patil, V. Miniyar, and A. Bandal

The research presented here has two key objectives. The first is to apply risk terrain modeling (RTM) to forecast the crime of shootings. The risk terrain maps that were produced from RTM use a range of contextual information relevant to the opportunity structure of shootings to estimate risks of future shootings as they are distributed throughout a geography. The second objective was to test the predictive power of the risk terrain maps over two month time periods, and to compare them against the predictive ability of retrospective hot spot maps. Results suggest that risk terrains provide a statistically significant forecast of future shootings across a range of cut points and are substantially more accurate than retrospective hot spot mapping. In addition, risk terrain maps produce information that can be operationalized by police administrators easily and efficiently, such as for directing police patrols to coalesced high risk areas.

2) Italian text categorization with lemmatization and support vector machines

AUTHORS: M. Faúndez-Zanuy, F. C. Morabito

The present research examines a structural model of violent crime in Portland, Oregon, exploring spatial patterns of both crime and its covariates. Using standard structural measures drawn from an opportunity framework, the study provides results from a global ordinary least squares model, assumed to fit for all locations within the study area. Geographically weighted regression (GWR) is then introduced as an alternative to such traditional approaches to modeling crime. The GWR procedure estimates a local model, producing a set of mappable parameter estimates and t-values of significance that vary over space. Several structural measures are found to have relationships with crime that vary significantly with location. Results indicate that a mixed model— with both spatially varying and fixed parameters—may provide the most accurate model of crime. The present study demonstrates the utility of GWR for exploring local processes that drive crime levels and examining misspecification of a global model of urban violence.

3) “A new simple and effective measure for bag-of-word inter-document similarity measurement

AUTHORS: Ting, T. Washio, and G. Haffari,

Social networks 1 produce enormous quantity of data. Twitter, a microblogging network, consists of over 230 million active users posting over 500 million tweets every day. We propose to analyze public data from Twitter to predict crime rates. Crime rates have increased in the past recent years. Although crime stoppers are utilizing various technics to reduce crime rates, none of the previous approaches targeted utilizing the language usage (offensive vs. non-offensive) in Tweets as a source of information to predict crime rates. In this paper, we hypothesize that analyzing the language usage in tweets is a valid measure to predict crime rates in cities. Tweets were collected for a period of 3 months in the Houston and New York City by locking the collection by geographic longitude and latitude. Further, tweets regarding crime events in the two cities were collected for verification of the validity of the prediction algorithm. We utilized Support Vector Machine (SVM) classifier to create a model of prediction of crime rates based on tweets. Finally, we report the validity of prediction algorithm in predicting crime rates in cities.

4) AUTHORS: M. Oghbaie and M. M. Zanjireh

In this paper¹ we introduce a family of models to describe the spatio-temporal dynamics of criminal activity. It is argued here that with a minimal set of mechanisms corresponding to elements that are basic in the study of crime, one can observe the formation of hot spots. By analyzing the simplest versions of our model, we exhibit a self-organized critical state of illegal activities that we proposed.

5) ‘Pairwise document similarity measure based on present term set,

AUTHORS: M. Oghbaie and M. M. Zanjireh

In this paper¹ we introduce a family of models to describe the spatio-temporal dynamics of criminal activity. It is argued here that with a minimal set of mechanisms corresponding to elements that are basic in the study of crime, one can observe the formation of hot spots. By analysing the simplest versions of our model, we exhibit a self-organised critical state of illegal activities that we propose to call a warm spot or a tepid milieu² depending on the context. It is characterised by a positive level of illegal or uncivil activity that maintains itself without exploding, in contrast with genuine hot spots where localised high level or peaks are being formed. Within our framework, we further investigate optimal policy issues under the constraint of limited resources in law enforcement and deterrence. We also introduce extensions of our model that take into account repeated victimisation effects, local and long range interactions, and briefly discuss some of the resulting effects such as hysteresis phenomena.

6) Early work by Page (1966) introduced Project Essay Grade (PEG), one of the first automated essay scoring systems using statistical techniques. More recent AES systems leverage ML and NLP to evaluate essays on multiple dimensions such as coherence, organization, and argument strength.

7) Studies such as Piech et al. (2013) investigated the use of ML for grading assignments in Massive Open Online Courses (MOOCs). These systems need to scale to handle thousands of submissions and often incorporate peer grading to enhance the training data.

8) The work by Burrows, Gurevych, and Stein (2015) emphasizes the importance of semantic analysis in evaluating subjective answers. Techniques such as Latent Semantic Analysis (LSA) and Word2Vec have been used to capture the meaning of text beyond simple keyword matching.

9) Recent studies have shown that transformer-based models like BERT significantly outperform previous models in tasks such as question answering, text classification, and essay scoring. Devlin et al. (2018) demonstrated BERT's capability to understand context, making it highly effective for subjective answer evaluation.

CHAPTER-3

SYSTEM ANALYSIS & STUDY

3.1 Existing System

- 1) Much work has been done on the topic of subjective answers evaluation in one form or another, such as measuring Similarity between different texts, words, and even documents.
- 2) Finding the context behind the text and mapping it with the solution's context, counting the noun-phrase in the documents, matching keywords in the answers, and so on.

3.2 Disadvantages of Existing System

- 1) Existing studies tend to have synonyms.
- 2) Existing studies tend to have an extensive range of possible lengths.
- 3) Existing studies tend to be randomly ordered among their sentences.

3.3 Proposed System

- 1) This project proposes a new and improved way of evaluating descriptive question answers automatically using machine learning and natural language processing.
- 2) It uses 2 step approach to solving this problem.
- 3) First, the answers are evaluated using the solution and provided keywords using various similarity-based techniques such as word mover's distance.
- 4) This form of evaluation by machines is a big step forward in aiding the educational sector to perform their other duties efficiently and reduce the manual labor in trivial tasks such as comparing the answers with a correct solution.

3.4 Advantages of Proposed System

- 1) The purpose of this work is to improve our previously proposed prediction framework through alternative crime mapping and feature engineering approaches, and provide an open-source implementation that police analysts can use to deploy more effective predictive policing.

- 2) This work helps the law enforcement agencies to predict and detect crimes in India with improved accuracy and thus reduces the crime rate.

3.5 Feasibility Study

The feasibility of the project is analyzed in this phase and business for the proposal is put forth with a very general plan for the project and some estimates. During the system analysis the feasibility study of the proposed system is to be easily carried out. This is to ensure that the proposed system is not a burden to company. For feasibility analysis, some understanding of the major requirements for the system is essential.[26]

Three key considerations involved in the feasibility analysis are as follows:

- 1) Economical Feasibility
- 2) Technical Feasibility
- 3) Social Feasibility

1) Economic Feasibility

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely only available. Only the customized products had to be purchased.

2) Technical Feasibility

This study is carried out to check the technical feasibility, that is, the technical way requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available of technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for various implementing this system.

3) Social Feasibility

The aspect of study is to check the level of acceptance of the system by the user. This

includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

3.6 Operational Feasibility

Operational feasibility is the measure of how well a proposed system solves the problems, and takes advantage of the opportunities identified during scope definition and how it satisfies the requirements identified in the requirements analysis phase of system development. The operational feasibility assessment focuses on the degree to which the proposed development project fits in with the existing business environment and objectives with regard to development schedule, delivery date and existing business processes. To ensure success, desired operational outcomes must be imparted during design and development. These include such design-dependent parameters as reliability, maintainability, supportability, usability, producibility, disposability, sustainability, affordability and others. These parameters are required to be considered at the early stages of design if desired operational data are to be realized. A system design and development require appropriate and timely application of engineering and management efforts to meet the previously mentioned parameters. A system may serve its intended purpose most effectively when its technical and operating characteristics are engineered into the design. [27]

CHAPTER 4

SYSTEM DESIGN

4.1 System Architecture

The given below figure 4.1 highlights the system architecture used in this project.

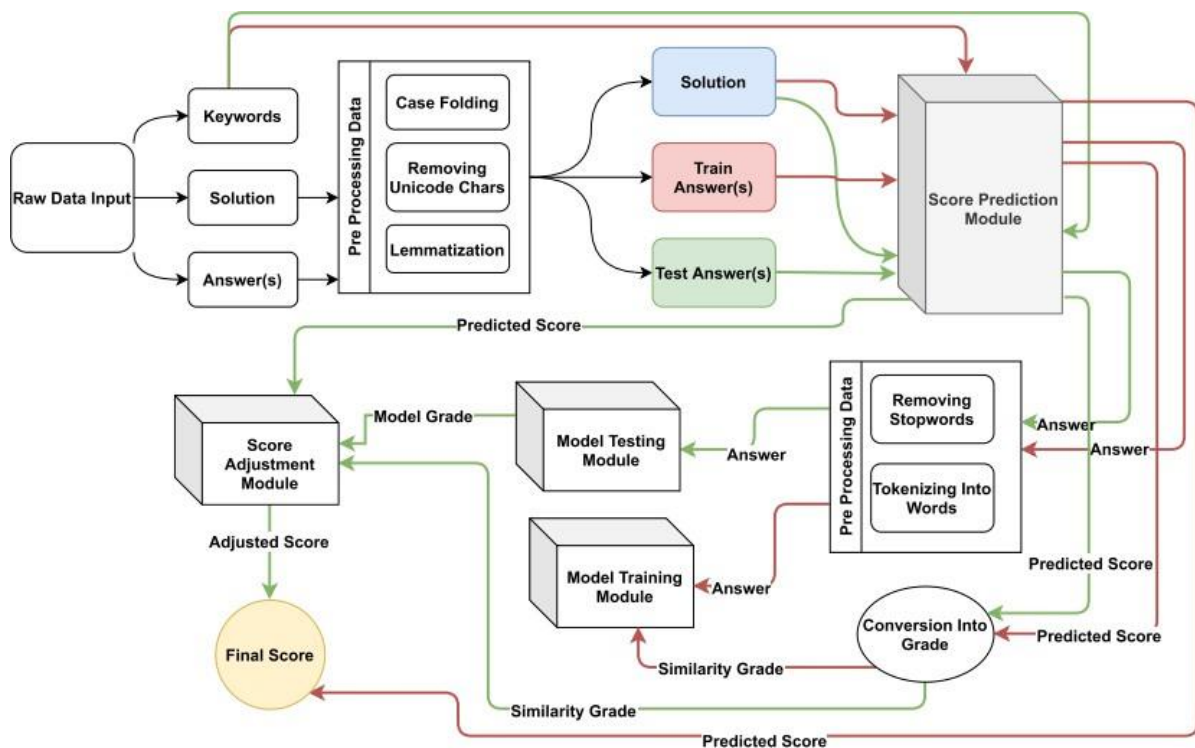


Figure 4.1 System Architecture

4.2 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 this in 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.

The given below figure 4.2 highlights the Data Flow Diagram.

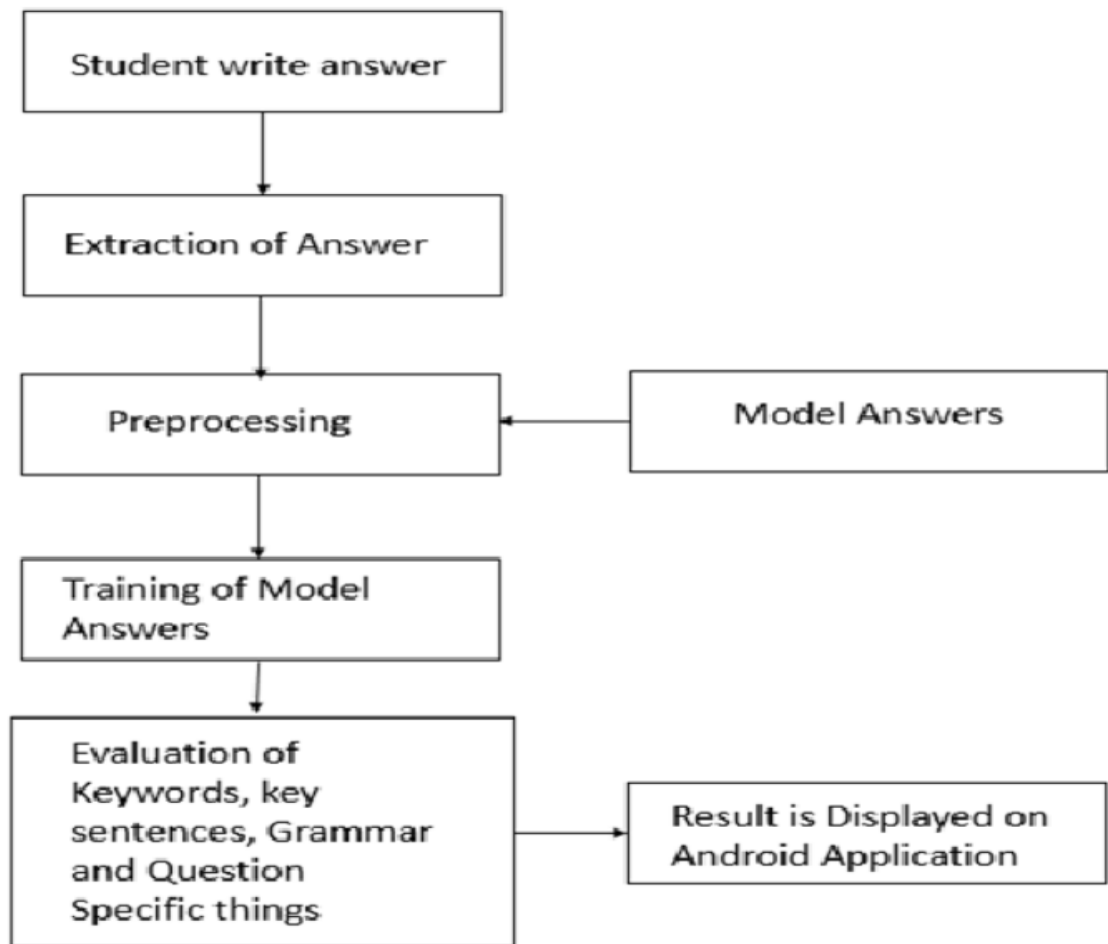


Figure 4.2 Data Flow Diagram

4.3 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 class, their & attributes, operations (or methods), and the relationships among the classes. It also explains which class is contains information.

The given below figure 4.3 highlights the Class Diagram.

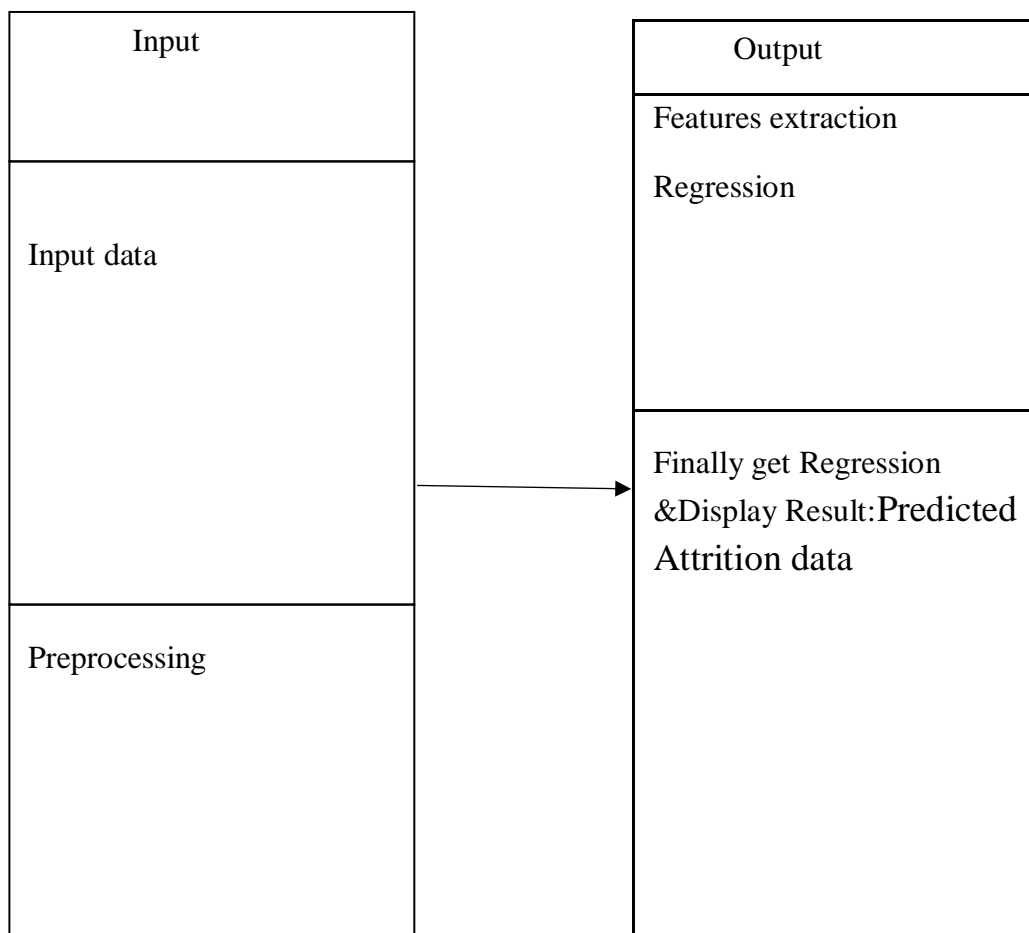


Figure 4.3 Class Diagram

4.4 Activity Diagram

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, the activity diagrams can be used to describe the business and operational step-by-step work flows of various components in vast system. An activity diagram shows the overall flow of control.[28]

The given below figure 4.4 highlights the Activity Diagram.

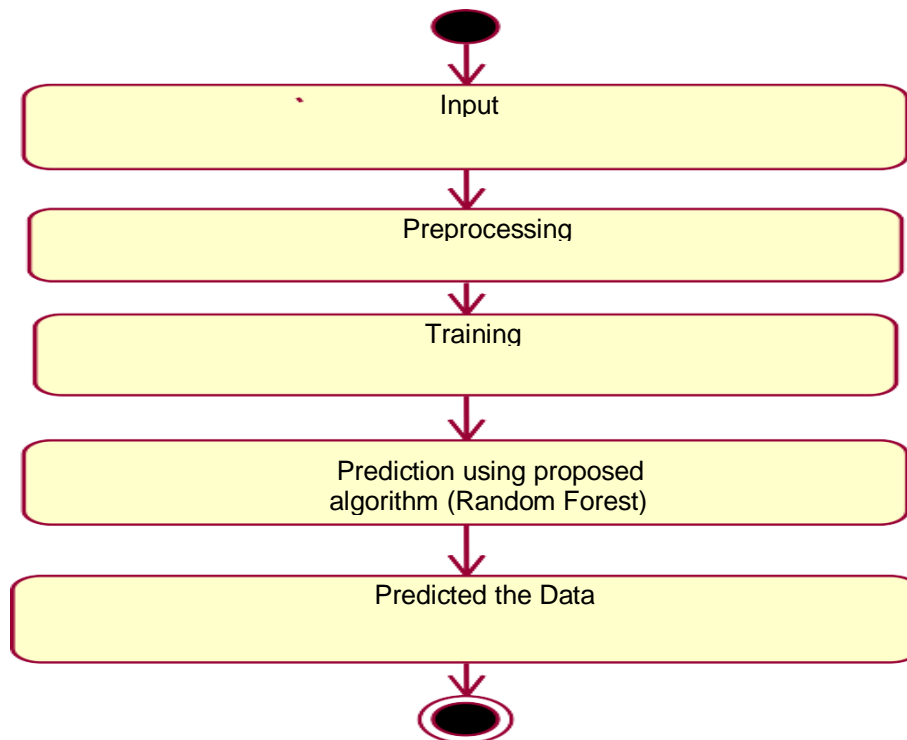


Figure 4.4 Activity Diagram

CHAPTER 5

INPUT & OUTPUT DESIGN

5.1 Input Design

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- 1) What data should be given as input?
- 2) How the data should be arranged or coded?
- 3) The dialog to guide the operating personnel in providing input.
- 4) Methods for preparing input validations and steps to follow when error occur.

5.2 Objectives

1) Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

2) It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

3) When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant.

5.3 Output Design

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.[29]

5.4 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 all functionality of components, sub - assemblies, assemblies and/or a finished product It is a the process of exercising software with the intent of ensuring that a software system that meets its requirements and user expectations and does not fail in an unacceptable way of manner. There are various types of test. Each test type addresses a specific testing of its requirement.

The given below figure 5.1 highlights the System Testing.



Figure 5.1 System Testing

5.5 Types of Testing

The given below figure 5.2 highlights the types of testing.

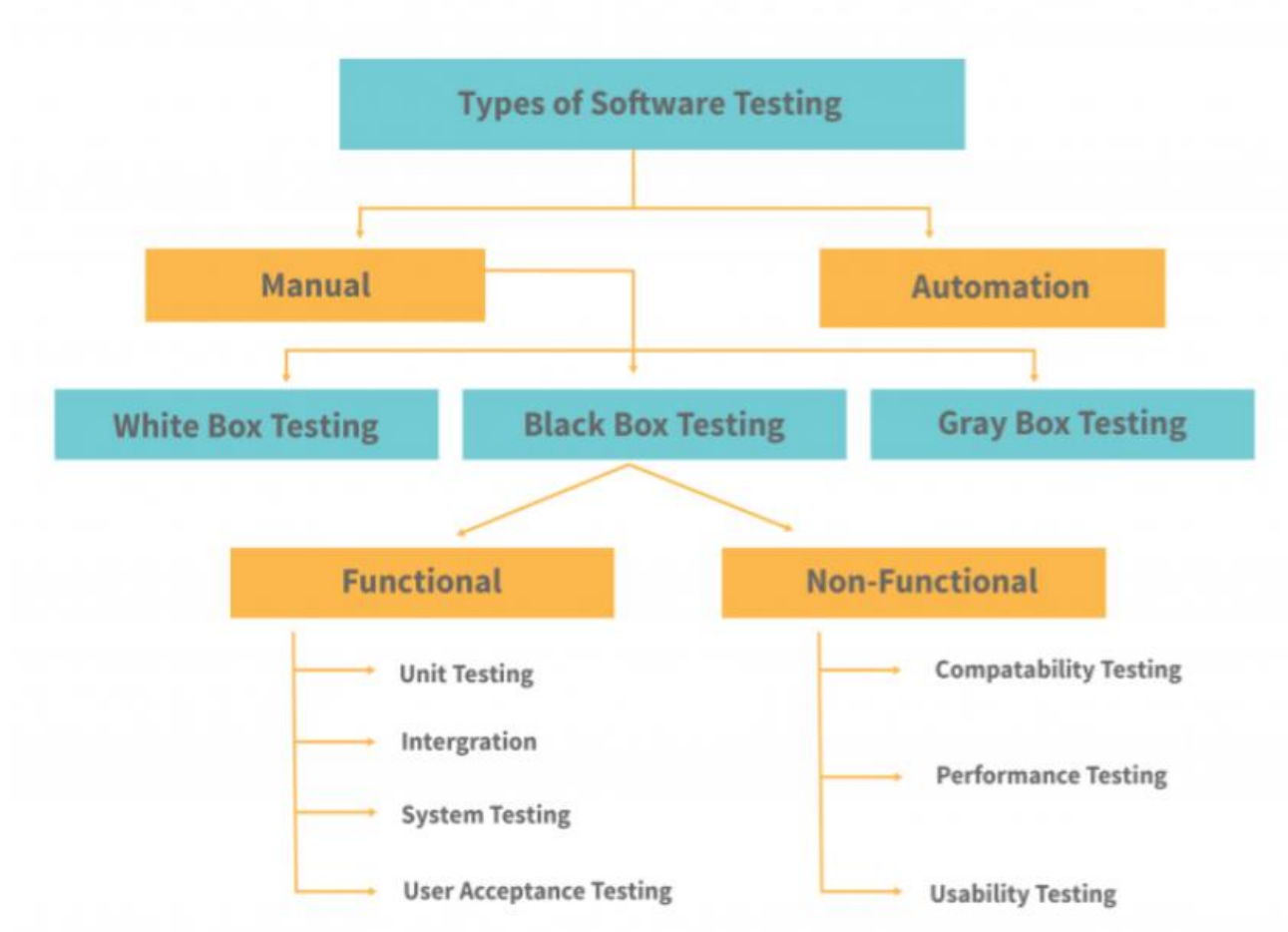


Figure 5.2 Types of Testing

5.5.1 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. The given below figure 5.3 highlights the Unit Testing.



Figure 5.3 Unit Testing

5.5.2 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 results of screens or fields. Integration tests demonstrate that although the components were individually satisfactory, 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.

The given below figure 5.4 highlights the Integration Testing.

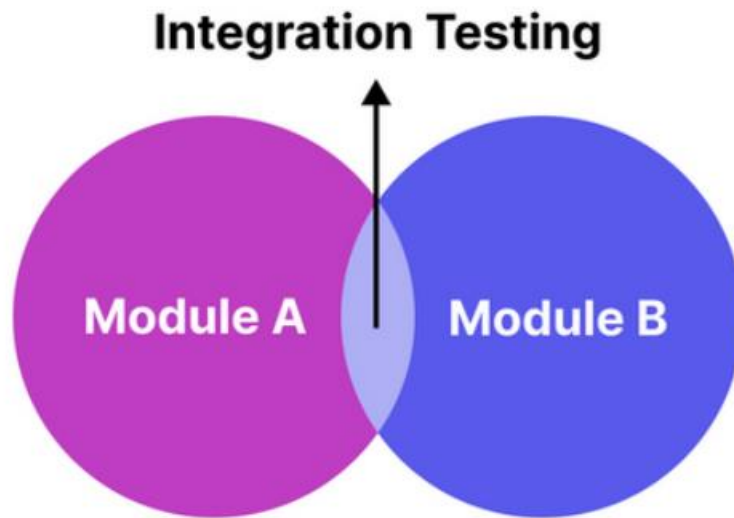


Figure 5.4 Integration Testing

5.5.3 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, keyfunctions, or special test cases. In addition, systematic coverage pertaining toidentify 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.

5.5.4 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.

5.5.5 White Box Testing

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

5.5.6 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. 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.[30]

CHAPTER 6

FRONT END AND BACK END

6.1 Front End

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- 1) **Python is Interpreted** – Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- 2) **Python is Interactive** – You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- 3) **Python is Object-Oriented** – Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- 4) **Python is a Beginner's Language** – Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

6.2 History of Python

- 1) Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.
- 2) Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, Small Talk, and Unix shell and other scripting languages.
- 3) Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

6.3 Python Feature's

Python's features include –

- 1) **Easy-to-learn** – Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- 2) **Easy-to-read** – Python code is more clearly defined and visible to the eyes.
- 3) **Easy-to-maintain** – Python's source code is fairly easy-to-maintain.
- 4) **A broad standard library** – Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- 5) **Interactive Mode** – Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- 6) **Portable** – Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- 7) **Extendable** – You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- 8) **Databases** – Python provides interfaces to all major commercial databases.
- 9) **GUI Programming** – Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
- 10) **Scalable** – Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below –

- 1) It supports functional and structured programming methods as well as OOP.
- 2) It can be used as a scripting language or can be compiled to byte-code for building large applications.

- 3) It provides very high-level dynamic data types and supports dynamic type checking.
- 4) It supports automatic garbage collection.
- 5) It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

Python is available on a wide variety of platforms including Linux and Mac OS X. Let's understand how to set up our Python environment.

6.4 Getting Python

The most up-to-date and current source code, binaries, documentation, news, etc., is available on the official website of Python <https://www.python.org>. Windows Installation is given below :

Here are the steps to install Python on Windows machine.

- 1) Open a Web browser and go to <https://www.python.org/downloads/>.
- 2) Follow the link for the Windows installer python-XYZ. MSI file where XYZ is the version you need to install.
- 3) To use this installer python-XYZ.msi, the Windows system must support Microsoft Installer 2.0. Save the installer file to your local machine and then run it to find out if your machine supports MSI.
- 4) Run the downloaded file. This brings up the Python install wizard, which is really easy to use. Just accept the default settings, wait until the install is finished, and you are done.

The Python language has many similarities to Perl, C, and Java. However, there are some definite differences between the languages.

First Python Program

Let us execute programs in different modes of programming.

6.5 Interactive Mode Programming

Invoking the interpreter without passing a script file as a parameter brings up the following prompt –

The given below figure 6.1 highlights the interactive mode of programming.

```
>>>
```

```
$ python  
  
Python2.4.3(#1,Nov112010,13:34:43)  
  
[GCC 4.1.220080704(RedHat4.1.2-48)] on linux2  
  
Type "help", "copyright", "credits" or "license" for more information.
```

Figure 6.1 Interactive Mode of Programming

Type the following text at the Python prompt and press the Enter –

The given below figure 6.2 highlights the python prompt.

```
>>>print "Hello, Python!"
```

Figure 6.2 Python Prompt

If you are running new version of Python, then you would need to use print statement with parenthesis as in **print ("Hello, Python!")**; However in Python version 2.4.3, this produces the following result –

The given below figure 6.3 highlights the Hello Python Prompt.

```
Hello, Python!
```

Figure 6.3 Hello Python Prompt

6.6 Script Mode Programming

Invoking the interpreter with a script parameter begins execution of the script and continues until the script is finished. When the script is finished, the interpreter is no longer active. Let us write a simple Python program in a script. Python files have extension **.py**. Type the following source code in a test.py file –

The given below figure 6.4 highlights the successful attempt of code.

```
print"Hello, Python!"
```

Figure 6.4 Successful Attempt

We assume that you have Python interpreter set in PATH variable. Now, try to run this program as follows –

The given below figure 6.5 highlights the Path Variable.

```
$ python test.py
```

Figure 6.5 Path Variable

This produces the following result –

The given below figure 6.6 highlights the result of above mention code.

```
Hello, Python!
```

Figure 6.6 Result

6.7 Flask Framework

Flask is a web application framework written in Python. Armin Ronacher, who leads an international group of Python enthusiasts named Pocco, develops it.

Flask is based on Werkzeug WSGI toolkit and Jinja2 template engine. Both are Pocco projects.

Http protocol is the foundation of data communication in world wide web. Different methods of data retrieval from specified URL are defined in this protocol.

The following table summarizes different http methods –

By default, the Flask route responds to the **GET** requests. However, this preference can be altered by providing methods argument to **route()** decorator.

In order to demonstrate the use of **POST** method in URL routing, first let us create an HTML form and use the **POST** method to send form data to a URL.

The given below figure 6.7 highlights the login page window code.

```
<html>

<body>

<formaction="http://localhost:5000/login"method="post">

<p>Enter Name:</p>

<p><inputtype="text"name="nm"/></p>

<p><inputtype="submit"value="submit"/></p>

</form>
</body>
</html>

from flask import Flask, redirect, url_for, request

app=Flask(__name__)

@app.route('/success/<name>')
```

Figure 6.7 Login Page Window Code

The given below figure 6.8 highlights the Post and Get Code.

```
def success(name):  
  
    return'welcome %s'% name  
  
    @app.route('/login',methods=['POST','GET'])  
  
    def login():  
  
        if request.method=='POST':  
  
            user=request.form['nm']  
  
            return redirect(url_for('success',name= user))  
        else:  
            user=request.args.get('nm')  
            return redirect(url_for('success',name= user))
```

Figure 6.8 Post and Get Code

After the development server starts running, open **login.html** in the browser, enter name in the text field and click **Submit**.

The given below figure 6.9 highlights the login window page.



The screenshot shows a web browser window. The address bar contains the file path 'file:///C:/login.ht'. The main content area of the browser displays a simple login form. At the top, it says 'Enter Name:'. Below this is a text input field with the value 'mvl' entered. Underneath the input field is a button labeled 'submit'.

Figure 6.9 Login Window Page

Form data is POSTed to the URL in action clause of form tag.

http://localhost/login is mapped to the **login()** function. Since the server has received data by **POST** method, value of 'nm' parameter obtained from the form data is obtained by –

```
user = request.form['nm']
```

It is passed to **‘/success’** URL as variable part. The browser displays a **welcome** message in the window.

The given below figure 6.10 highlights the local host page.

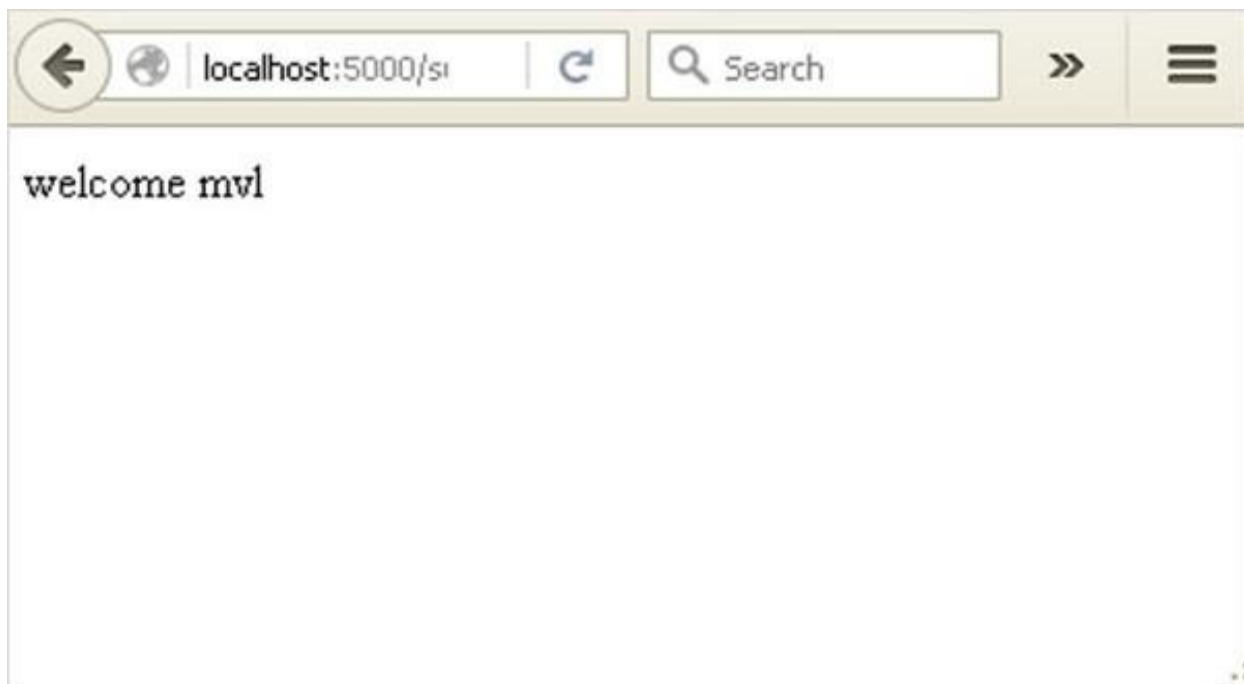


Figure 6.10 Local Host Page

Change the method parameter to **‘GET’** in **login.html** and open it again in the browser. The data received on server is by the **GET** method. The value of 'nm' parameter is now obtained by –

```
User = request.args.get('nm')
```

Here, **args** is dictionary object containing a list of pairs of form parameter and its corresponding value. The value corresponding to 'nm' parameter is passed on to **‘/success’** URL as before.

6.8 Python

Python is a popular programming language. It was created in 1991 by Guido van Rossum.

It is used for:

- 1) web development (server-side),
- 2) software development,
- 3) mathematics,
- 4) system scripting.

6.9 Python Highlights

- 1) Python can be used on a server to create web applications.
- 2) Python can be used alongside software to create workflows.
- 3) Python can connect to database systems. It can also read and modify files.
- 4) Python can be used to handle big data and perform complex mathematics.
- 5) Python can be used for rapid prototyping, or for production-ready software development.

6.10 Uses of Python

- 1) Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- 2) Python has a simple syntax similar to the English language.
- 3) Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
- 4) Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
- 5) Python can be treated in a procedural way, an object-orientated way or a functional way.
- 6) The most recent major version of Python is Python 3, which we shall be using in this tutorial. However, Python 2, although not being updated with anything other than security updates, is still quite popular.
- 7) In this tutorial Python will be written in a text editor. It is possible to write Python in an Integrated Development Environment, such as Thonny, Pycharm, Netbeans or Eclipse which are particularly useful when managing larger collections of Python files. Python Syntax compared to other programming languages
- 8) Python was designed to for readability, and has some similarities to the English language with influence from mathematics

6.11 Back End

Many PCs and Macs will have python already installed.

To check if you have python installed on a Windows PC, search in the start bar for Python or run the following on the Command Line (cmd.exe):

```
C:\Users\Your Name>python --version
```

To check if you have python installed on a Linux or Mac, then on linux open the command line or on Mac open the Terminal and type:

```
python --version
```

If you find that you do not have python installed on your computer, then you can download it for free from the following website: <https://www.python.org/>

Python Quickstart

Python is an interpreted programming language, this means that as a developer you write Python (.py) files in a text editor and then put those files into the python interpreter to be executed.

The way to run a python file is like this on the command line:

```
C:\Users\Your Name>python helloworld.py
```

Where "helloworld.py" is the name of your python file.

Let's write our first Python file, called helloworld.py, which can be done in any text editor.

```
helloworld.py
```

```
print("Hello, World!")
```

Simple as that. Save your file. Open your command line, navigate to the directory where you saved your file, and run:

```
C:\Users\Your Name>python helloworld.py
```

The output should read:

```
Hello, World!
```

Congratulations, you have written and executed your first Python program.

The Python Command Line

To test a short amount of code in python sometimes it is quickest and easiest not to write the code in a file. This is made possible because Python can be run as a command line itself.

Type the following on the Windows, Mac or Linux command line:

```
C:\Users\Your Name>python
```

From there you can write any python, including our hello world example from earlier in the tutorial:

```
C:\Users\Your Name>python
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32 bit
(Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello, World!")
```

Which will write "Hello, World!" in the command line:

```
C:\Users\Your Name>python
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32 bit
(Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello, World!")
Hello, World!
```

Whenever you are done in the python command line, you can simply type the following to quit the python command line interface:

```
exit()
```

Execute Python Syntax

As we learned in the previous page, Python syntax can be executed by writing directly in the Command Line:

```
>>> print("Hello, World!")
Hello,
World!
```

Or by creating a python file on the server, using the .py file extension, and running it in the Command Line:

```
C:\Users\Your Name>python myfile.py
```

Python Indentations

Where in other programming languages the indentation in code is for readability only, in Python the indentation is very important.

Python uses indentation to indicate a block of code.

The given below figure 6.11 highlights the Front End vs Back End.

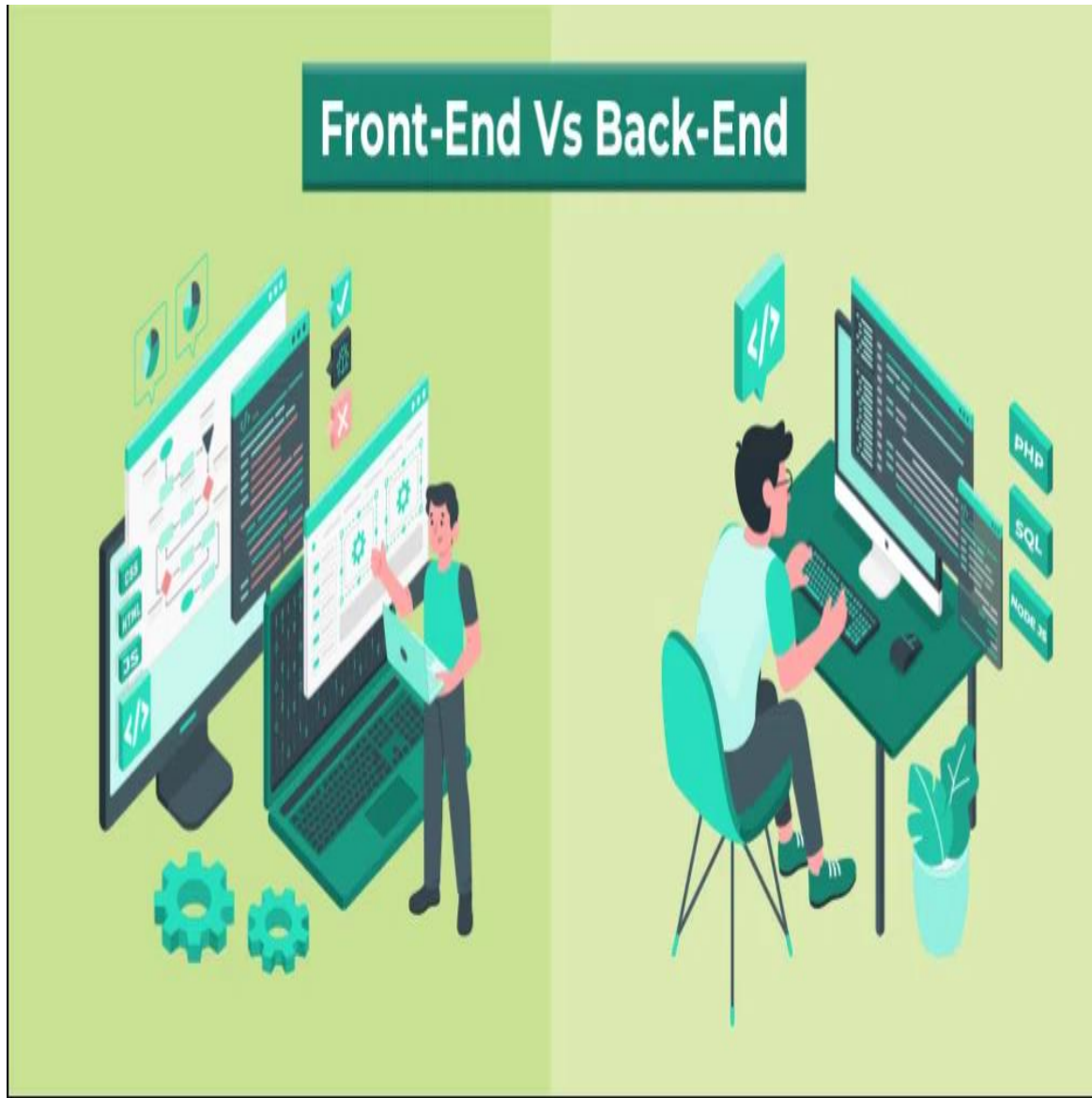


Figure 6.11 Front End Vs Back End

Chapter 7

IMPLEMENTATION

Here are some screenshots of code regarding the project which would be used in the project. It includes Code regarding CSS, Python App.

The given below figure 7.1 shows Style Code – Part 1

```
1  /**
2   * Template Name: Restaurantly - v3.0.1
3   * Template URL: https://bootstrapmade.com/restaurantly-restaurant-template/
4   * Author: BootstrapMade.com
5   * License: https://bootstrapmade.com/license/
6   */
7
8  /*-----
9  # General
10 -----*/
11 body {
12     font-family: "Open Sans", sans-serif;
13     background: #0c0b09;
14     color: #fff;
15 }
16
17 a {
18     color: #cda45e;
19     text-decoration: none;
20 }
21
22 a:hover {
23     color: #d9ba85;
24     text-decoration: none;
25 }
26
27 h1, h2, h3, h4, h5, h6 {
28     font-family: "Playfair Display", serif;
29 }
```

Figure 7.1 Style Code - Part 1

The given below figure 7.2 highlights the Style Code – Part 2

```
27 h1, h2, h3, h4, h5, h6 {
28     font-family: "Playfair Display", serif;
29 }
30
31 /*-----
32 # Preloader
33 -----*/
34 #preloader {
35     position: fixed;
36     top: 0;
37     left: 0;
38     right: 0;
39     bottom: 0;
40     z-index: 9999;
41     overflow: hidden;
42     background: #1a1814;
43 }
44
45 #preloader:before {
46     content: "";
47     position: fixed;
48     top: calc(50% - 30px);
49     left: calc(50% - 30px);
50     border: 6px solid #1a1814;
51     border-top-color: #cda45e;
52     border-bottom-color: #cda45e;
53     border-radius: 50%;
54     width: 60px;
55     height: 60px;
```

Figure 7.2 Style Code - Part 2

The given below figure 7.3 highlights the CSS Code – Part 1

```
58 }
59
60 @-webkit-keyframes animate-preloader {
61   0% {
62     transform: rotate(0deg);
63   }
64   100% {
65     transform: rotate(360deg);
66   }
67 }
68
69 @keyframes animate-preloader {
70   0% {
71     transform: rotate(0deg);
72   }
73   100% {
74     transform: rotate(360deg);
75   }
76 }
77
78 /*-----
79 # Back to top button
80 -----*/
81 .back-to-top {
82   position: fixed;
83   visibility: hidden;
84   opacity: 0;
85   right: 15px;
86   bottom: 15px;
```

Figure 7.3 CSS Code– Part 1

The given below figure 7.4 highlights the CSS Code – Part 2

```
82   position: fixed;
83   visibility: hidden;
84   opacity: 0;
85   right: 15px;
86   bottom: 15px;
87   z-index: 996;
88   width: 44px;
89   height: 44px;
90   border-radius: 50px;
91   transition: all 0.4s;
92   border: 2px solid #cda45e;
93 }
94
95 .back-to-top i {
96   font-size: 28px;
97   color: #cda45e;
98   line-height: 0;
99 }
100
101 .back-to-top:hover {
102   background: #cda45e;
103   color: #1a1814;
104 }
105
106 .back-to-top:hover i {
107   color: #444444;
108 }
109
```

Figure 7.4 CSS Code – Part 2

The given below figure 7.5 highlights the Python Code

```
1 import json
2 import plotly
3 import pandas as pd
4 import numpy as np
5
6 from nltk.stem import WordNetLemmatizer
7 from nltk.tokenize import word_tokenize
8
9 from flask import Flask
10 from flask import render_template, request
11 from plotly.graph_objs import Bar
12 from sklearn.externals import joblib
13 from sqlalchemy import create_engine
14
15
16 app = Flask(__name__)
17
18 def tokenize(text):
19     tokens = word_tokenize(text)
20     lemmatizer = WordNetLemmatizer()
21
22     clean_tokens = []
23     for tok in tokens:
24         clean_tok = lemmatizer.lemmatize(tok).lower().strip()
25         clean_tokens.append(clean_tok)
```

Figure 7.5 Python Code – Part 1

The given below figure 7.6 highlights the Python Code – Part 2

```
18 def tokenize(text):
19     tokens = word_tokenize(text)
20     lemmatizer = WordNetLemmatizer()
21
22     clean_tokens = []
23     for tok in tokens:
24         clean_tok = lemmatizer.lemmatize(tok).lower().strip()
25         clean_tokens.append(clean_tok)
26
27     return clean_tokens
28
29 # load data
30 engine = create_engine('sqlite:///../data/Disaster_ETL.db')
31 df = pd.read_sql_table('messages', engine)
32
33 # load model
34 model = joblib.load("../models/model.pkl")
35
36
37
38 app = Flask(__name__)
39
40 @app.route('/')
41 @app.route('/first')
42 def first():
43     return render_template('first.html')
44 @app.route('/login')
45 def login():
```

Figure 7.6 Python Code - Part 2

The given below figure 7.7 highlights the Python Code – Part 3

```
40 @app.route('/')
41 @app.route('/first')
42 def first():
43     return render_template('first.html')
44 @app.route('/login')
45 def login():
46     return render_template('login.html')
47 def home():
48     return render_template('home.html')
49 @app.route('/upload')
50 def upload():
51     return render_template('upload.html')
52 @app.route('/preview',methods=["POST"])
53 def preview():
54     if request.method == 'POST':
55         dataset = request.files['datasetfile']
56         df = pd.read_csv(dataset,encoding = 'unicode_escape')
57         df.set_index('Id', inplace=True)
58         return render_template("preview.html",df_view = df)
59
60 @app.route('/prediction1')
61 def prediction1():
62     return render_template('home.html')
63 @app.route('/chart')
64 def chart():
65     return render_template('chart.html')
66 @app.route('/prediction')
67 def prediction():
```

Figure 7.7 Python Code – Part 3

The given below figure 7.8 highlights the Python Code – Part 4

```
40 @app.route('/')
41 @app.route('/first')
42 def first():
43     return render_template('first.html')
44 @app.route('/login')
45 def login():
46     return render_template('login.html')
47 def home():
48     return render_template('home.html')
49 @app.route('/upload')
50 def upload():
51     return render_template('upload.html')
52 @app.route('/preview',methods=["POST"])
53 def preview():
54     if request.method == 'POST':
55         dataset = request.files['datasetfile']
56         df = pd.read_csv(dataset,encoding = 'unicode_escape')
57         df.set_index('Id', inplace=True)
58         return render_template("preview.html",df_view = df)
59
60 @app.route('/prediction1')
61 def prediction1():
62     return render_template('home.html')
63 @app.route('/chart')
64 def chart():
65     return render_template('chart.html')
66 @app.route('/prediction')
67 def prediction():
```

Figure 7.8 Python Code – Part 4

The given below figure 7.9 highlights the Python Code – Part 5

```
61 def prediction1():
62
63 @app.route('/chart')
64 def chart():
65     return render_template('chart.html')
66 @app.route('/prediction')
67 def prediction():
68     return render_template("home.html")
69 @app.route('/crime')
70 def crime():
71     return render_template("crime.html")
72 @app.route('/crimes')
73 def crimes():
74     return render_template("crimes.html")
75 @app.route('/total')
76 def total():
77     return render_template("total.html")
78 @app.route('/theft')
79 def theft():
80     return render_template('theft.html')
81
82 @app.route('/predict',methods=['POST'])
83 def predict():
84     df= pd.read_csv("some.csv", encoding="latin-1")
85     # Features and Labels
86     df['label'] = df['l']
87     X = df['t']
88     y = df['label']
89     print(X)
```

Figure 7.9 Python Code – Part 5

The given below figure 7.10 highlights the Python Code – Part 6

```
89 print(X)
90 # Extract Feature With CountVectorizer
91 cv = CountVectorizer(ngram_range=(1, 2))
92 X = cv.fit_transform(X) # Fit the Data
93 from sklearn.model_selection import train_test_split
94 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.10, random_state=0)
95 #Naive Bayes Classifier
96 from sklearn.naive_bayes import MultinomialNB
97
98 clf = svm.SVC(kernel='linear')
99 clf.fit(X_train,y_train)
00 clf.score(X_test,y_test)
01
02 if request.method == 'POST':
03     message = request.form['message']
04     data = [message]
05     vect = cv.transform(data).toarray()
06     my_prediction = clf.predict(vect)
07     return render_template('result.html',prediction = my_prediction)
08
09
10
11 if __name__ == '__main__':
12     app.run(debug=True)
```

Figure 7.10 Python Code – Part 6

The given below figure 7.11 highlights the Python Code – Part 7

```
1 from sklearn.feature_extraction.text import TfidfVectorizer
2 from sklearn.metrics.pairwise import cosine_similarity
3 corpus=[]
4 with open('reference.txt') as f1:
5     corpus.append(f1.read())
6     f1.close()
7 with open('test.txt') as f1:
8     corpus.append(f1.read())
9     f1.close()
10 vectorizer=TfidfVectorizer()
11 trsfm=vectorizer.fit_transform(corpus)
12 score=cosine_similarity(trsfm[0],trsfm)[0][1]*100
13 print(score)
```

Figure 7.11 Python Code – Part 7

7.1 Implementation of code regarding Flask

```
import encodings

from flask import Flask, render_template, request, redirect, url_for

from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.metrics.pairwise import cosine_similarity

import pandas as pd

import numpy as np

# from keybert import KeyBERT

from nltk.stem import PorterStemmer

from nltk.stem import WordNetLemmatizer

import language_tool_python

import json

import pandas as pd

import numpy as np

from flask import Flask

from flask import render_template, request

from plotly.graph_objs import Bar

import joblib

from rake_nltk import Rake

tool = language_tool_python.LanguageTool('en-US')

rake_nltk_var = Rake();

kw_model = KeyBERT(model='all-mpnet-base-v2')
```

```

def calculate_cosine_similarity(corpus):

    vectorizer=TfidfVectorizer()

    trsfm=vectorizer.fit_transform(corpus)

    score=cosine_similarity(trsfm[0],trsfm)[0][1]*10

    return round(score,2)

def stemmer(keywords_list):

    ps = PorterStemmer()

    for i in range(len(keywords_list)):

        keywords_list[i] = ps.stem(keywords_list[i])

    return keywords_list

def lemmatize(keywords_list):

    lemmatizer = WordNetLemmatizer()

    for i in range(len(keywords_list)):

        keywords_list[i] = lemmatizer.lemmatize(keywords_list[i])

    return keywords_list

corpus=[]

app=Flask(    name    )

@app.route('/')

@app.route('/first')

def first():

    return render_template('first.html')

@app.route('/login')

```

```

def login():

    return render_template('login.html')

def home():

    return render_template('home.html')

@app.route('/upload')

def upload():

    return render_template('upload.html')

@app.route('/preview',methods=["POST"])

def preview():

    if request.method == 'POST':

        dataset = request.files['datasetfile']

        df = pd.read_csv(dataset,encoding = 'utf-8')

        df.set_index('Id', inplace=True)

        return render_template("preview.html",df_view = df)

@app.route('/index')

def index():

    return render_template('index.html')

@app.route('/chart')

def chart():

    return render_template('chart.html')

@app.route('/success', methods=['POST','GET'])

```



```

def success():

    if request.method=='POST':

        f=None

        f=request.files['file']

        if f == None:

            return render_template('erroredirect.html',message='empty_file')

        fname=f.filename

        if fname.lower().endswith(('png', '.jpg', '.jpeg', '.tiff', '.bmp', '.gif')):

            return render_template('erroredirect.html',message='image_file')

        answer=f.read().decode('utf-8')

        matches = tool.check(answer)

        keywords_correct_answer_list = None

        keywords_answer = kw_model.extract_keywords(answer,

        keyphrase_ngram_range=(0,1),

        stop_words='english',

        highlight=False,

        keywords_answer_list= list(dict(keywords_answer).keys())

        rake_nltk_var.extract_keywords_from_text(answer)

        keywords_answer_list = rake_nltk_var.get_ranked_phrases()

```

```

f.close()

f.save(f.filename)

with open('reference.txt',encoding='utf-8') as fgt:

    corpus.append(fgt.read())

    correct_answer=corpus[0]

    keywords_correct_answer = kw_model.extract_keywords(correct_answer,

keyphrase_ngram_range=(0,1),
stop_words='english',

highlight=False,

keywords_correct_answer_list =

list(dict(keywords_correct_answer).keys())

rake_nltk_var.extract_keywords_from_text(correct_answer)

keywords_correct_answer_list = rake_nltk_var.get_ranked_phrases()

fgt.close()
common_keywords = 0

keywords_answer_list = stemmer(keywords_answer_list)

keywords_correct_answer_list = stemmer(keywords_correct_answer_list)

keywords_answer_list = lemmatize(keywords_answer_list)

keywords_correct_answer_list = lemmatize(keywords_correct_answer_list)

keywords_answer_list_set = set(keywords_answer_list)

keywords_correct_answer_list_set = set(keywords_correct_answer_list)

print(keywords_answer_list)

print(keywords_correct_answer_list)

```

```

for ka in keywords_answer_list_set:

    for kca in keywords_correct_answer_list_set:

        if ka == kca:

            common_keywords+=1

complete_list = keywords_answer_list + keywords_correct_answer_list

unique_keywords = len(np.unique(complete_list))

keywords_match_score = (common_keywords/unique_keywords)*10

corpus.append(answer)

cosine_sim_score = calculate_cosine_similarity(corpus)

score=((6/10)*(cosine_sim_score))+((4/10)*(keywords_match_score))

if score >= 10:

    score = 10

corpus.clear()

if len(matches)>0:

    score = score - len(matches)

if score<0:

    score = 0
    print("Errors\t",len(matches))

print('Cosine_sim_score:\t',cosine_sim_score)

print('keyword_match_score:\t',keywords_match_score)

return

render_template('success.html',name=fname,answer=answer,score=score,correct_answer= correct_answer,matches=len(matches))

```

```

    if request.method=='GET':

return render_template('index.html')

if name == '    main    ':

    app.run(debug=True)

    from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.metrics.pairwise import cosine_similarity

corpus=[]

with open('reference.txt') as f1:

    corpus.append(f1.read())

    f1.close()

with open('test.txt') as f1:

    corpus.append(f1.read())

    f1.close()

vectorizer=TfidfVectorizer()

trsfm=vectorizer.fit_transform(corpus)

score=cosine_similarity(trsfm[0],trsfm)[0][1]*100

print(score)

```

CHAPTER 8

RESULT & DISCUSSION

The desktop view of the project highlights in figure 8.1

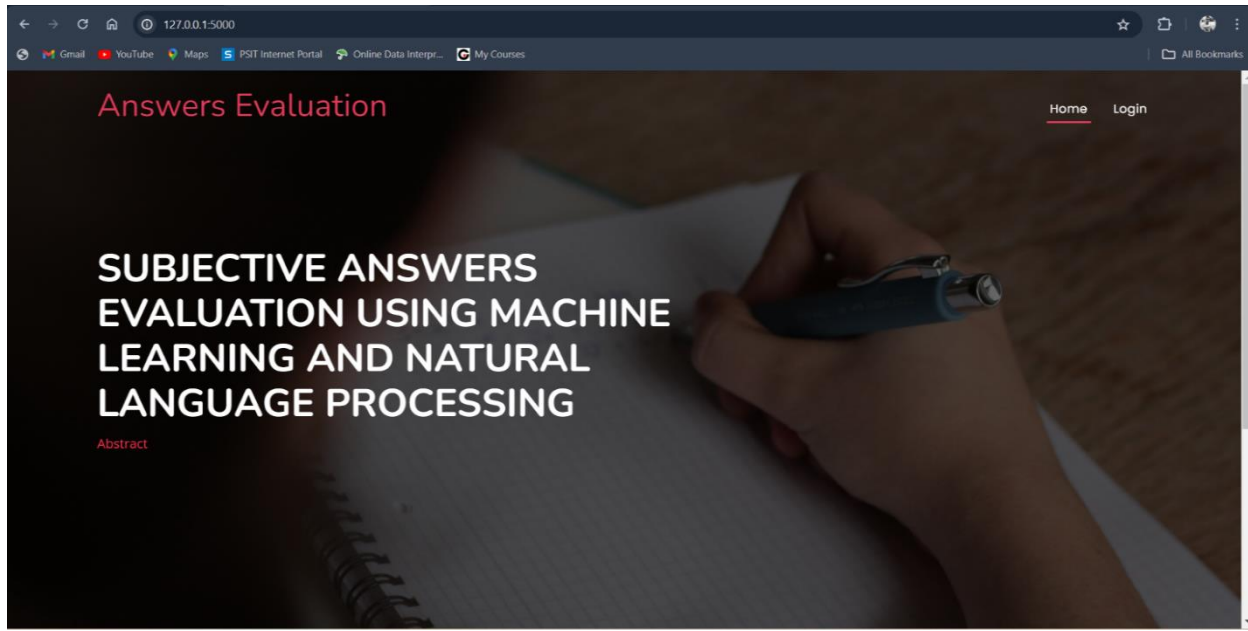


Figure 8.1 Desktop View

The uploading and prediction window highlights in figure 8.2

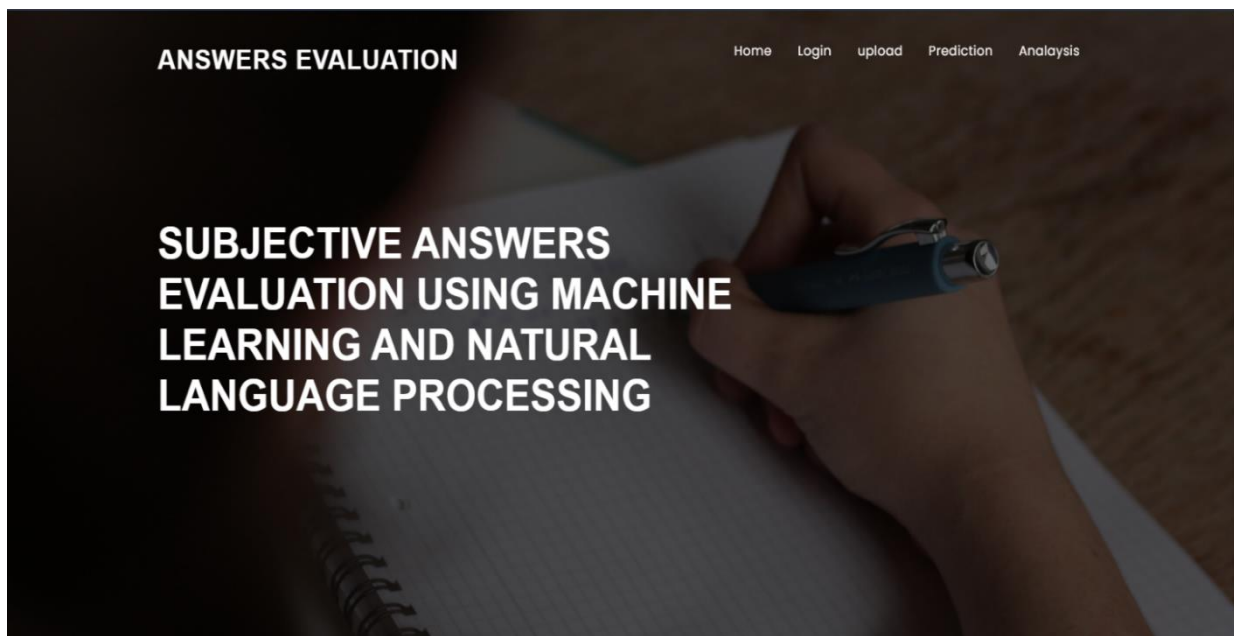


Figure 8.2 Uploading and Prediction Window

The login window highlights in figure 8.3

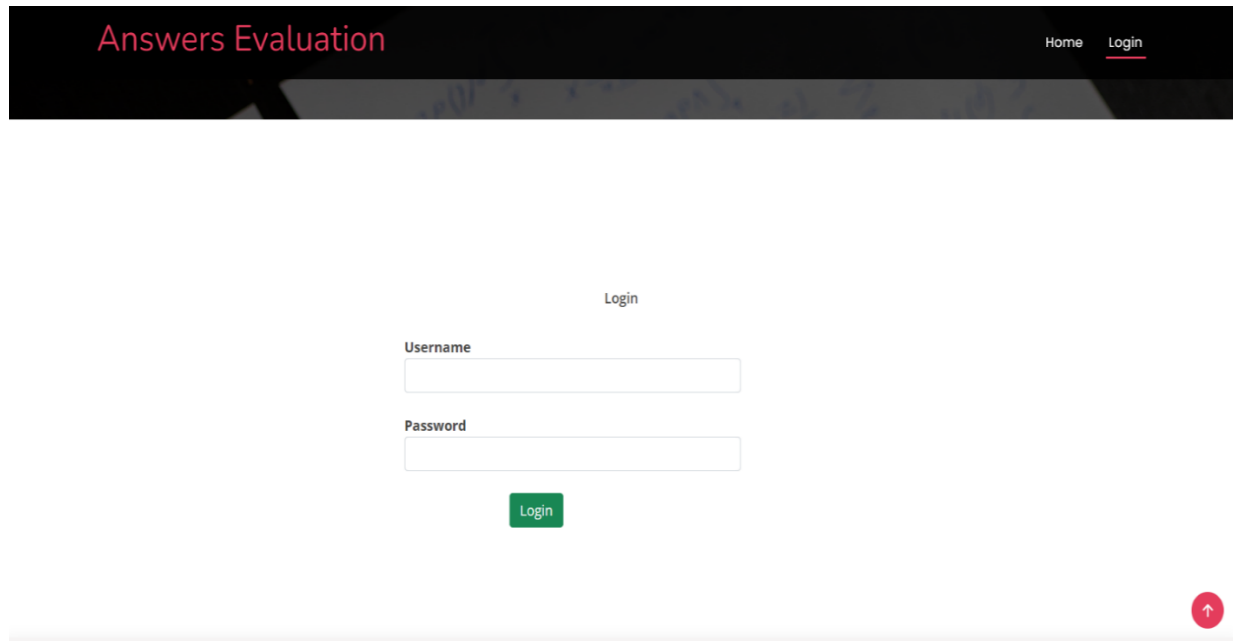


Figure 8.3 Login Window

The score appearing window highlights in figure 8.4

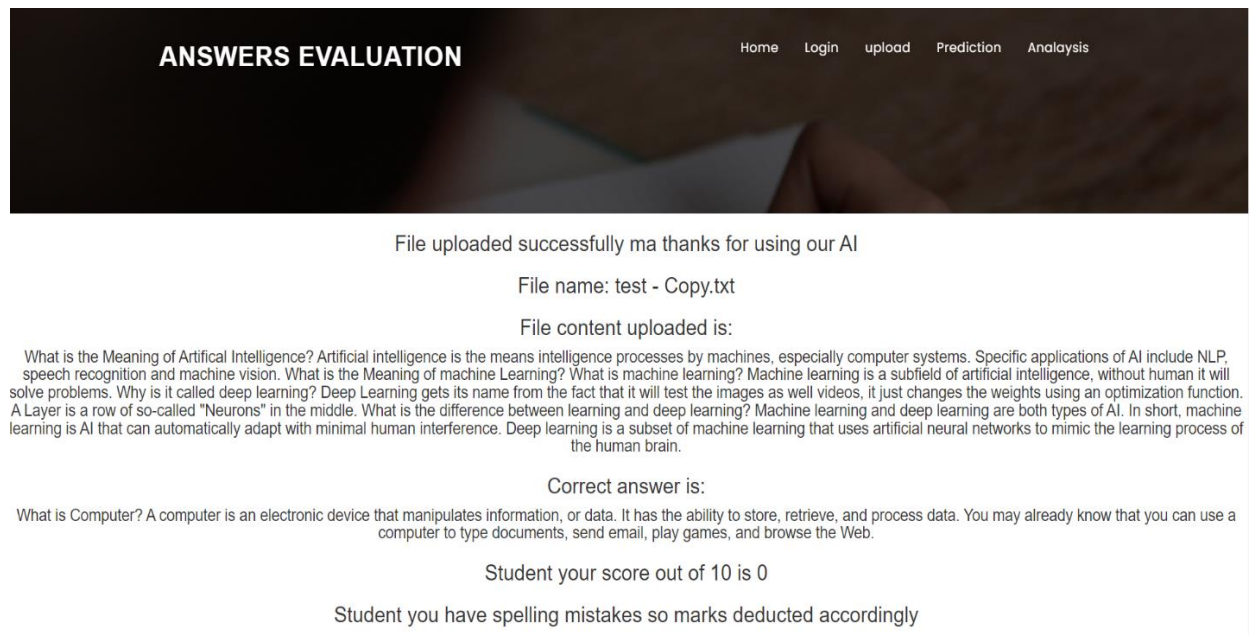


Figure 8.4 Score Appearing Window

CHAPTER 9

CONCLUSION & FUTURE SCOPE

The creation of machine learning models for subjective response evaluation marks a significant progress in educational technology. Traditional techniques of assessing subjective responses, such as human grading, are time-consuming, inconsistent, and prone to prejudice. The application of machine learning techniques provides a solution that can improve the speed, uniformity, and fairness of the grading process.

Our findings show that machine learning models, particularly those that use natural language processing (NLP) approaches, can successfully analyze subjective responses. Models such as BERT (Bidirectional Encoder Representations from Transformers) and its derivatives have demonstrated promising results in comprehending the context and semantics of student replies. These models may be trained to evaluate the accuracy, completeness, and relevance of replies using predetermined criteria.

High degrees of accuracy and consistency in grading can be attained by machine learning models, which are on par with those of human evaluators. These models are able to pick up on the subtleties of subject-specific knowledge and assessor expectations by utilizing big datasets of graded responses.

Bias Reduction includes subjectivity, weariness, and inadvertent biases are among the biases that human graders may introduce into their work. Automated grading systems can assist in mitigating these biases. As a result, pupils from all backgrounds experience an assessment procedure that is more egalitarian.

Scalability includes large volumes of answers can be handled by machine learning models with ease, which makes them appropriate for use in large-scale educational evaluations and standardized testing.

Feedback Generation implies these models are not just for grading; they may also give students thorough feedback that will assist them identify their areas of weakness and where they can grow. This individualized feedback is helpful in improving the educational process.

9.1 Future Scope

- 1) Subjective answer evaluation with machine learning has a huge future potential and could lead to major breakthroughs in the field of education. To increase and broaden these models' capabilities, there are a number of areas are investigated. Enhanced Model Training: More complex models can be produced by consistently improving training datasets and methodologies. Models can generalize more effectively across multiple disciplines and educational levels by incorporating diverse information from different educational situations. Contextual Understanding In the future, models could be created that are more adept at comprehending the larger context of inquiries and responses.
- 2) Subjective answer evaluation will become available to a worldwide audience with the development of models capable of evaluating responses in several languages. In order to do this, models must be adjusted for linguistic and cultural variances and trained on multilingual datasets. Students can receive immediate feedback by integrating machine learning models into learning platforms for real-time answer evaluation. This can allow adaptive learning systems that adapt to the demands of each individual learner and improve interactive learning environments. It is crucial to guarantee that machine learning is used in education in an ethical manner. This entails resolving concerns about data privacy, openness in the grading system, and possible effects on instructional strategies. Future studies should concentrate on creating moral standards and optimal procedures for implementing these technologies.
- 3) Collaborative Learning to support group projects and peer evaluation, collaborative learning systems can be combined with machine learning models. Models can help assess individual contributions, group dynamics, and project outcomes as a whole. Integration with Educational Tools: To offer a more comprehensive educational experience, subjective response evaluation models can be integrated with learning management systems, intelligent tutoring systems, and virtual classrooms, among other educational technology. A smooth and all-encompassing approach to student assessment and support may be offered by this integration.

- 4) Longitudinal Research: Longitudinal research can yield important insights into the long-term effects of machine learning-based grading on student learning outcomes, motivation, and educational equality. The creation of fair and efficient assessment procedures can be influenced by the findings of these investigations.

In summary, the incorporation of machine learning into the subjective assessment of answers has revolutionary possibilities for the field of education. We can design more efficient, equitable, and customized learning environments that are advantageous to both teachers and students by tackling present issues and looking into future possibilities. To fully utilize these technologies and influence the direction of education in the future, more research and development in this area will be essential.

REFERENCES

- [1] J. Wang and Y. Dong, “Measurement of text similarity: A survey,” *Information*, vol. 11, no. 9, p. 421, Aug. 2020.
- [2] M. Han, X. Zhang, X. Yuan, J. Jiang, W. Yun, and C. Gao, “A survey on the techniques, applications, and performance of short text semantic similarity,” *Concurrency Comput., Pract. Exper.*, vol. 33, no. 5, Mar. 2021.
- [3] M. S. M. Patil and M. S. Patil, “Evaluating Student descriptive answers using natural language processing,” *Int. J. Eng. Res. Technol.*, vol. 3, no. 3, pp. 1716–1718, 2014.
- [4] P. Patil, S. Patil, V. Miniyar, and A. Bandal, “Subjective answer evaluation using machine learning,” *Int. J. Pure Appl. Math.*, vol. 118, no. 24, pp. 1–13, 2018.
- [5] J. Muangprathub, S. Kajornkasirat, and A. Wanichsombat, “Document plagiarism detection using a new concept similarity in formal concept analysis,” *J. Appl. Math.*, vol. 2021, pp. 1–10, Mar. 2021.
- [6] X. Hu and H. Xia, “Automated assessment system for subjective questions based on LSI,” in *Proc. 3rd Int. Symp. Intell. Inf. Technol. Secur. Informat.*, Apr. 2010, pp. 250–254.
- [7] M. Kusner, Y. Sun, N. Kolkin, and K. Weinberger, “From word embeddings to document distances,” in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 957–966.
- [8] C. Xia, T. He, W. Li, Z. Qin, and Z. Zou, “Similarity analysis of law documents based on Word2vec,” in *Proc. IEEE 19th Int. Conf. Softw. Qual. Rel. Secur. Companion (QRS-C)*, Jul. 2019, pp. 354–357.
- [9] H. Mittal and M. S. Devi, “Subjective evaluation: A comparison of several statistical techniques,” *Appl. Artif. Intell.*, vol. 32, no. 1, pp. 85–95, Jan. 2018.
- [10] L. A. Cutrone and M. Chang, “Automarking: Automatic assessment of open questions,” in *Proc. 10th IEEE Int. Conf. Adv. Learn. Technol.*, Sousse, Tunisia, Jul. 2010, pp. 143–147.
- [11] G. Srivastava, P. K. R. Maddikunta, and T. R. Gadekallu, “A two-stage text feature selection algorithm for improving text classification,” *Tech. Rep.*, 2021.
- [12] H. Mangassarian and H. Artail, “A general framework for subjective information extraction

from unstructured English text,” *Data Knowl. Eng.*, vol. 62, no. 2, pp. 352–367, Aug. 2007.

[13] B. Oral, E. Emekligil, S. Arslan, and G. Eryigit, “Information extraction ~ from text intensive and visually rich banking documents,” *Inf. Process. Manage.*, vol. 57, no. 6, Nov. 2020, Art. no. 102361.

[14] H. Khan, M. U. Asghar, M. Z. Asghar, G. Srivastava, P. K. R. Maddikunta, and T. R. Gadekallu, “Fake review classification using supervised machine learning,” in *Proc. Pattern Recognit. Int. Workshops Challenges (ICPR)*. New York, NY, USA: Springer, 2021, pp. 269–288.

[15] S. Afzal, M. Asim, A. R. Javed, M. O. Beg, and T. Baker, “URLdeepDetect: A deep learning approach for detecting malicious URLs using semantic vector models,” *J. Netw. Syst. Manage.*, vol. 29, no. 3, pp. 1–27, Mar. 2021.

[16] N. Madnani and A. Cahill, “Automated scoring: Beyond natural language processing,” in *Proc. 27th Int. Conf. Comput. Linguistics (COLING)*, E. M. Bender, L. Derczynski, and P. Isabelle, Eds. Santa Fe, NM, USA: Association for Computational Linguistics, Aug. 2018, pp. 1099–1109.

[17] Z. Lin, H. Wang, and S. I. McClean, “Measuring tree similarity for natural language processing based information retrieval,” in *Proc. Int. Conf. Appl. Natural Lang. Inf. Syst. (NLDB) (Lecture Notes in Computer Science)*, vol. 6177, C. J. Hopfe, Y. Rezgui, E. Métais, A. D. Preece, and H. Li, Eds. Cardiff, U.K.: Springer, 2010, pp. 13–23.

[18] G. Grefenstette, “Tokenization,” in *Syntactic Wordclass Tagging*. Springer, 1999, pp. 117–133.

[19] K. Sirts and K. Peekman, “Evaluating sentence segmentation and word Tokenization systems on Estonian web texts,” in *Proc. 9th Int. Conf. Baltic (HLT) (Frontiers in Artificial Intelligence and Applications)* vol. 328, U. Andrius, V. Jurgita, K. Jolantai, and K. Danguole, Eds. Kaunas, Lithuania: IOS Press, Sep. 2020, pp. 174–181.

[21] K. K. Kandaswamy, K.-C. Chou, T. Martinetz, S. Möller, P. N. Suganthan, S. Sridharan, and G. Pugalenti, “AFP-pred: A random forest approach for predicting

- antifreeze proteins from sequence-derived properties," *J. Theor. Biol.*, vol. 270, no. 1, pp. 5662, Feb. 2011.
- [22] V. F. Rodriguez-Galiano, B. Ghimire, J. Rogan, M. Chica-Olmo, and J. P. Rigol-Sanchez, "An assessment of the effectiveness of a random forest classifier for land-cover classification," *ISPRS J. Photogramm. Remote Sens.*, vol. 67, pp. 93104, Jan. 2012.
- [23] L. Lin, J. Jiakai, S. Guangwen, L. Weiwei, Y. Hongjie, and L. Wenjuan, "Hotspot prediction of public property crime based on spatial differentiation of crime and built environment," *J. Geo-Inf. Sci.*, vol. 21, no. 11, pp. 16551668, 2019.
- [24] Z. Jun and H. Wenbo, "Recent advances in Bayesian machine learning," *J. Comput. Res. Develop.*, vol. 52, no. 1, pp. 1626, 2015.
- [25] J. T. Huang, J. Li, and Y. Gong, "An analysis of convolutional neural networks for speech recognition," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, South Brisbane, QLD, Australia, Apr. 2015, pp. 49894993.
- [26] Z. Feiyan, J. Linpeng, and D. Jun, "Review of convolutional neural network," *Chin. J. Comput.*, vol. 40, no. 6, pp. 12291251, 2017.
- [27] Y. Yang, J. Dong, X. Sun, E. Lima, Q. Mu, and X. Wang, "A CFCC-LSTM model for sea surface temperature prediction," *IEEE Geosci. Remote Sens. Lett.*, vol. 15, no. 2, pp. 207211, Feb. 2018.
- [28] X. Hong, R. Lin, C. Yang, N. Zeng, C. Cai, J. Gou, and J. Yang, "Predicting Alzheimer's disease using LSTM," *IEEE Access*, vol. 7, pp. 8089380901, 2019.
- [29] L. Mou, P. Zhao, and Y. Chen, "Short-term traffic flow prediction: A long short-term memory model enhanced by temporal information," in *Proc. 19th COTA Int. Conf. Transp. Prof. CICTP Transp. China-Connect.World*, 2019, pp. 24112422.
- [30] L. E. Cohen and M. Felson, "Social change and crime rate trends: A routine activity approach," *Amer. Sociol. Rev.*, vol. 44, no. 4, p. 588, Aug. 1979.