

**A MINI PROJECT REPORT
ON
TASK-CENTRIC AUTONOMOUS MACHINE LEARNING
MODELING APPROACH**

Submitted in partial fulfillment of the requirement

for the award of the degree of

**BACHELOR OF TECHNOLOGY
IN
Computer Science and Engineering**

By

B. SIRI (21P61A0531)

B.MANASA (21P61A0532)

B.SAI ADITHYA (21P61A0533)

Under the esteemed guidance of

Dr.A.L.Sreenivasulu

Professor

Dept. of CSE

Counselling Code : **VBIT**



(A UGC Autonomous Institution, Approved by AICTE, Accredited by NBA & NAAC-A Grade, Affiliated to JNTUH)

**VIGNANA BHARATHI INSTITUTE OF
TECHNOLOGY**

(A UGC Autonomous Institution, Approved by AICTE, Affiliated to JNTUH,

Accredited by NBA & NAAC) Aushapur (V), Ghatkesar (M), Medchal(dist.)

2024-2025



VIGNANA BHARATHI
Institute of Technology

Counselling Code: **VBIT**



(A UGC Autonomous Institution, Approved by AICTE, Accredited by NBA and NAAC-A Grade, Affiliated to JNTUH)

**DEPARTMENT
OF
COMPUTER SCIENCE AND ENGINEERING**

CERTIFICATE

This is to certify that the mini project titled **Task-Centric Autonomous Machine Learning Modeling Approach** submitted to the **Vignana Bharathi Institute of Technology**, affiliated to **JNTUH**, by **B. Siri (21P61A0531)**, **B. Manasa (21P61A0532)**, **B. Sai Adithya (21P61A0533)** is a bonafide work carried out by them.

The results embodied in this report have not been submitted to any other University for the award of any degree.

Guide

Signature:

Dr.A.L. Sreenivasulu
Professor
Dept. of CSE

Head of the Department

Signature:

Dr. Raju Dara
Professor
Dept. of CSE

EXTERNAL EXAMINER

DECLARATION

We, **B.Siri, B.Manasa, B.Sai Adithya** bearing hall ticket numbers **21P61A0531, 21P61A0532, 21P61A0533** hereby declare that the mini project report entitled “**Task-Centric Autonomous Machine Learning Modeling Approach**” under the guidance of **DR.A.L.Sreenivasulu**, Department of Computer Science and Engineering, **Vignana Bharathi Institute of Technology, Hyderabad**, have submitted to Jawaharlal Nehru Technological University Hyderabad, Kukatpally, in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Business System.

This is a record of bonafide work carried out by us and the results embodied in this project have not been reproduced or copied from any source. The results embodied in this project report have not been submitted to any other university or institute for the award of any other degree or diploma.

B. SIRI (21P61A0531)

B. MANASA (21P61A0532)

B. SAI ADITHYA (21P61A0533)

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of the people who made it possible and whose encouragement and guidance has been a source of inspiration throughout the course of the project.

We are extremely thankful to our beloved Chairman, **Dr.N.Goutham Rao** and Secretary, **Dr.G.Manohar Reddy** who took keen interest to provide us the infrastructural Facilities for carrying out the project work.

It is great pleasure to convey our profound sense of gratitude to our principal **Dr. P.V. S Srinivas, Dr. Raju Dara**, Head of the CSE Department, Vignana Bharathi Institute of Technology for having been kind enough for arranging the necessary facilities for executing the project in the college.

We would like to express our sincere gratitude to our Guide, **Dr.A.L. Sreenivasulu**, Professor, **CSE Dept, Vignana Bharathi Institute of Technology**, whose guidance and valuable suggestions have been indispensable to bring about the completion of our project.

We wish to acknowledge special thanks to the Project Coordinator **Mr G. Arun and Dr.N. Swapna**, Associate Professors of **CSE Dept, Vignana Bharathi Institute of Technology**, for assessing seminars, inspiration, moral support and giving us valuable suggestions in our project.

We would also like to express our gratitude to all the staff members and lab faculty, department of **Computer Science and Engineering, Vignana Bharathi Institute of Technology** for the constant help and support.

We wish a deep sense of gratitude and heartfelt thanks to management for providing excellent lab facilities and tools. Finally, we thank all those whose guidance helped us in this regard.

ABSTRACT

Recently, many researchers are intensely engaged in investigation on the artificial intelligence technology that recognizes, learns, inferences, and acts on external information in a wide range of fields by combining technologies of computing, big data and machine learning algorithms. The artificial intelligence technology is currently used in almost all industries, and many machine learning experts are working on integrating and standardizing various machine learning tools so that non-experts can easily apply them to their domain. The researchers are also studying an autonomous machine learning as well as ontology construction for standardizing the machine learning concepts. In this paper, we classify typical problem-solving steps for autonomous machine learning as tasks, and present a problem-solving process. We propose the modeling method of an autonomous machine learning using a process of the task execution on machine learning such as workflow. The proposed task ontology-based machine learning model defines a task-based process grouping scheme of UML activities. And it will automatically generate and extend the machine learning models by transformation rules based on common elements and structures (relationships and processes between elements).

Keywords:

Artificial Intelligence Technology, Task Ontology, Autonomous Machine Learning, Ontology Construction.



VISION

To become, a Center for Excellence in Computer Science and Engineering with a focused Research, Innovation through Skill Development and Social Responsibility.

MISSION

DM-1: Provide a rigorous theoretical and practical framework across State-of-the-art infrastructure with an emphasis on software development.

DM-2: Impact the skills necessary to amplify the pedagogy to grow technically and to meet interdisciplinary needs with collaborations.

DM-3: Inculcate the habit of attaining the professional knowledge, firm ethical values, innovative research abilities and societal needs.

PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

PEO-01: Domain Knowledge: Synthesize mathematics, science, engineering fundamentals, pragmatic programming concepts to formulate and solve engineering problems using prevalent and prominent software.

PEO-02: Professional Employment: Succeed at entry- level engineering positions in the software industries and government agencies.

PEO-03: Higher Degree: Succeed in the pursuit of higher degree in engineering or other by applying mathematics, science, and engineering fundamentals.

PEO-04: Engineering Citizenship: Communicate and work effectively on team-based engineering projects and practice the ethics of the profession, consistent with a sense of social responsibility.

PEO-05: Lifelong Learning: Recognize the significance of independent learning to become experts in chosen fields and broaden professional knowledge.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO-01: Ability to explore emerging technologies in the field of computer science and engineering.

PSO-02: Ability to apply different algorithms in different domains to create innovative products.

PSO-03: Ability to gain knowledge to work on various platforms to develop useful and secured applications to the society.

PSO-04: Ability to apply the intelligence of system architecture and organization in designing the new era of computing environment.

PROGRAM OUTCOMES (POs)

PO-01: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO-02: Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO-03: Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and cultural, societal, and environmental considerations.

PO-04: Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO-05: Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.

PO-06: The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO-07: Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

.

PO-08:Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO-09: Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO-10: Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO-11: Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO-12: Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

LIST OF FIGURES

S.NO	NAME	Page No
4. 1	Architecture Diagram	10
4.2.1.1	Use Case Diagram	13
4.2.2.1	Sequence Diagram	15
4.2.3.1	Activity Diagram	17
4.2.4.1	Class Diagram	19
4.2.5.1	Deployment Diagram	19
7.1	Input&Output	37

LIST OF INPUT / OUTPUT SCREENS

S.No	Names	Page No
7.1	Home Page	32
7.2	User Login	32
7.3	User Register	33
7.4	User HomePage	33
7.5	User Upload File	34
7.6	User View Details	34
7.7	Detection of Images	35
7.8	Final Output	36
7.9	Admin Dashboard	37

TABLE OF CONTENTS

CHAPTER-1:

INTRODUCTION	1-2
1.1 Introduction to Task-Centric Autonomous Machine Learning Modeling Approach	1
1.2 Motivation	1
1.3 Overview of Existing System	1
1.4 Overview of Proposed System	1
1.5 Problem Definition	2
1.6 System Features	2

CHAPTER 2:

LITERATURE SURVEY	3-6
-------------------	-----

CHAPTER 3:

REQUIREMENTS ANALYSIS	7-9
3.1 Operating Environment	7
3.2 Functional Requirements	8
3.3 Non-functional Requirements	8
3.4 System Analysis	9

CHAPTER 4:

SYSTEM DESIGN	10-23
4.1 Architecture Diagram	10-12
4.2 UML Diagrams	13-20

CHAPTER 5:

IMPLEMENTATION	21-28
5.1 Explanation of key functions	21
5.2 Method of Implementation	22-23
5.3 Modules	24-28

CHAPTER 6:

TESTING AND VALIDATION	29-31
6.1 Testing process	34
6.2 Test Cases	35-36

CHAPTER 7:

INPUT/OUTPUT SCREENS	32-37
----------------------	-------

CHAPTER 8:

CONCLUSION AND FURTHER ENHANCEMENT	38
8.1 Conclusion	38
8.2 Future Scope	38

CHAPTER 9:

REFERENCES	39-40
------------	-------

1.INTRODUCTION

1.1 Introduction

Autonomous Machine Learning (AutoML) systems aim to simplify and enhance the development of machine learning models through automation. This project, "Autonomous Machine Learning Modeling using a Task Ontology," explores how task ontology can guide the automatic selection, configuration, and evaluation of machine learning models. By formalizing tasks and mapping them to suitable algorithms, the system enhances efficiency and reduces the manual effort required for model development. The focus lies on optimizing the model lifecycle, including preprocessing, feature selection, and hyperparameter tuning, while ensuring adaptability across diverse application domains.

1.2 Motivation

The motivation for this project stems from the increasing demand for scalable and efficient machine learning solutions. Traditional ML model development is resource-intensive, requiring domain expertise and significant trial-and-error. By leveraging task ontology, this project seeks to address these challenges, enabling faster deployment of accurate models in diverse fields such as healthcare, finance, and social media. Additionally, the automation facilitated by task ontology reduces human error and enhances reproducibility in ML workflows.

1.3 Overview of Existing System

Existing AutoML systems, such as Auto-WEKA and Google AutoML, automate various stages of model development. These systems typically rely on algorithm search spaces and heuristic optimization techniques like Bayesian optimization. However, their reliance on generic approaches limits adaptability to domain-specific problems. While they significantly reduce manual intervention, the absence of a structured ontology leads to inefficiencies in selecting algorithms tailored to specific tasks, particularly in dynamic or novel problem spaces. [1]

1.4 Overview of Proposed System

The proposed system introduces a task ontology-driven framework for AutoML. This framework formalizes the relationships between machine learning tasks (e.g., classification, regression) and corresponding algorithms, preprocessing techniques, and evaluation metrics. By encoding this knowledge, the system dynamically recommends optimal pipelines tailored to the given task. Advanced techniques like ensemble learning, meta-learning, and feature selection are incorporated to refine the modeling process further. The framework also emphasizes adaptability, enabling seamless integration across diverse application domains while reducing computational overhead.[2][3]

1.5 Problem Definition

Traditional AutoML systems struggle with inefficiencies due to their lack of task-specific knowledge representation. This limitation results in suboptimal algorithm selection and prolonged search processes, hindering model performance and deployment timelines. Moreover, the generic nature of existing systems fails to address domain-specific requirements, leading to challenges in scalability and applicability. Thus, there is a pressing need for a structured, ontology-driven approach that aligns model configurations with the unique demands of diverse machine learning tasks.

1.6 System Features

The proposed system incorporates the following features:

- **Task Ontology Integration:** Formalized mappings between tasks, algorithms, and evaluation metrics.
- **Automated Pipeline Generation:** Dynamic creation of ML workflows, including preprocessing, feature selection, and model training.
- **Adaptive Learning:** Continuous learning from prior tasks to improve future model recommendations.
- **Resource Efficiency:** Reduced computational complexity through optimized search spaces and task-specific heuristics.
- **Explainability:** Enhanced transparency in model recommendations using interpretable techniques like SHAP values and feature importance scores.
- This structured, ontology-driven approach aims to overcome the limitations of existing AutoML systems, ensuring robust and efficient model development tailored to diverse applications.

2.LITERATURE SURVEY

Autonomous Machine Learning Modeling using a Task Ontology:

This study focuses on developing a structured ontology for autonomous machine learning. It emphasizes creating task-oriented models that enable refinement and mapping to other domain ontologies, streamlining the machine learning process and enhancing flexibility.[1]

Ontology-Based Fault Tree Analysis Algorithms in a Fuzzy Environment for Autonomous Ships:

This research applies ontology-based approaches for fault tree analysis in the context of autonomous ships. The study highlights advanced methodologies for improving analysis reliability and applicability in a fuzzy decision-making environment. This framework utilizes machine learning to automatically assess the quality of resumes.[2]

3. Machines Learn Better with Better Data Ontology: Lessons from Philosophy of Induction and Machine Learning Practice:

The work explores how data ontology improves machine learning performance. It integrates concepts from the philosophy of induction and emphasizes utilizing UML diagrams for knowledge organization and reuse[3].

Using Ontology to Guide Reinforcement Learning Agents in Unseen Situations:

This paper discusses the application of ontologies to enhance the adaptability of reinforcement learning agents. The approach facilitates efficient experimentation and deployment in scenarios where agents encounter unfamiliar situations.[4]

Ontology Knowledge-Based Framework for Machine Learning Concept:

This study explores ontology-based knowledge design to organize machine learning concepts and techniques from diverse sources like books and articles. It focuses on structuring a database across four domains, achieving high precision (99.65%) and recall (95.90%) in experimental evaluations. The findings suggest its potential application in related systems and future research.[5]

Task Ontology: Ontology for Building Conceptual Problem-Solving Models:

This research focuses on task ontologies designed to build conceptual models for problem-solving. It highlights techniques to improve task decomposition while maintaining adaptability and flexibility in implementation.[6]

7.Expose: An Ontology for Data Mining Experiments:

This study explores an ontology named Expose, designed to organize and reuse knowledge in data mining experiments. It emphasizes structured task decomposition to simplify and streamline the experimental process. The approach aims to enhance efficiency and consistency in data mining workflows.[7]

8.TensorFlow: A System for Large-Scale Machine Learning:

This study explores TensorFlow, a scalable and flexible framework for large-scale machine learning . It emphasizes its versatility in handling diverse tasks and its suitability for extensive computational environments. The framework is positioned as a key tool for advancing machine learning applications.[8]

9. A Translation Approach to Portable Ontology Specifications:

This study explores a translation approach for creating portable ontology specifications to facilitate knowledge sharing. It emphasizes ensuring interoperability across diverse systems. The approach aims to standardize ontology usage in varied computational environments.[9]

10. Models for Representing Task Ontologies:

This study explores the use of UML activity diagrams to capture task control-flow and UML class diagrams to represent knowledge roles within tasks. It highlights the underrepresentation of task ontologies compared to domain ontologies and proposes combining these models with domain ontologies. The approach aims to comprehensively describe knowledge in application contexts.[10]

11. ML Schema: Exposing the Semantics of Machine Learning with Schemas and Ontologies:

This study explores the ML-Schema, a top-level ontology for standardizing information on machine learning algorithms, datasets, and experiments. It reviews state-of-the-art interchange formats and introduces ML-Schema as a canonical format developed over seven years. The study highlights its potential to enhance interpretability and interoperability across platforms and workflows.[11]

12. What Is an Ontology? :

This study explores the contrasting philosophical and computational meanings of "ontology," focusing on its role in knowledge engineering. It revisits and formalizes the definition of computational ontologies as "explicit specifications of conceptualizations." The paper emphasizes the importance of shared explicit specifications for effective knowledge representation.[12]

13. Ontology based data access in statoil

This study explores the contrasting philosophical and computational meanings of "ontology," focusing on its role in knowledge engineering. It revisits and formalizes the definition of computational ontologies as "explicit specifications of conceptualizations." The paper emphasizes the importance of shared explicit specifications for effective knowledge representation[13].

14. Ontology Development Methodology: A Systematic Review and Case Study

This paper compares five ontology development methodologies and includes a case study of the TITO ontology developed for the University of Tabuk's IT department. Future work focuses on evaluating and extending to other departments.[14]

15. An Ontological-Based Model to Data Governance for Big Data

This paper presents an ontological approach to data governance, utilizing semantic techniques and automatic ontology-based reasoning to manage big data complexity. The system, tested on Telefonica's global video service, shows promising results in simplifying data management through distributed components and a Shared Knowledge Plane.[15]

2.1 Drawbacks of Literature Survey

- **Lack of Practical Implementation:**

Many studies focus on theoretical frameworks without detailing real-world implementation or validation processes.

- **Narrow Domain Focus:**

Some papers focus on specific domains (e.g., autonomous ships, big data governance), limiting their generalizability.

- **Scalability Concerns:**

Limited evidence on how the proposed methods scale in dynamic or large-scale environments.

- **Interoperability Challenges:**

Insufficient exploration of seamless integration of ontologies with existing systems.

3. REQUIREMENTS ANALYSIS

3.1 Operating Environment:

An operational environment encompasses the hardware and software infrastructure needed to develop, test, and execute a system or application. It ensures compatibility and supports efficient functionality under specified conditions.

1.Hardware Requirements:

- **RAM: Min of 4 GB**

At least 4 GB of RAM is necessary to run the application efficiently, handle multitasking, and support resource-intensive operations like data processing.

- **Hard Disk: 1 TB**

A 1 TB hard disk provides ample storage space for the application, including data files, logs, models, and backups.

- **Operating System: Windows / Linux / MacOS**

The system supports multiple operating systems to ensure flexibility, enabling cross-platform development and deployment.

2.Software Requirements:

- **ProgrammingLanguages:Python,HTML,CSS,JavaScript**

Python is used for backend logic, while HTML, CSS, and JavaScript manage front-end development, creating interactive and responsive interfaces.[8]

- **Libraries:**

These libraries handle diverse tasks such as natural language processing (NLTK, Spacy), machine learning (TensorFlow/Keras), data visualization (Matplotlib), image processing (OpenCV), and data formatting (JSON).

- **WebFramework:**

Django is the chosen framework for backend web development, offering robust features for building secure and scalable web applications.[5]

- **IDEs:**

Integrated Development Environments like VS Code, PyCharm, and Colab facilitate efficient coding, debugging, and collaboration through user-friendly interfaces and tools.

3.2 Functional Requirements:

- **Task Ontology Development:**

The system should define and structure a task ontology for machine learning to organize knowledge related to ML modeling processes.

- **Autonomous Modeling:**

The system should automate the generation of machine learning models by leveraging the task ontology for selecting appropriate algorithms, preprocessing steps, and evaluation metrics.

- **Knowledge Reuse:**

The system should enable reuse of prior machine learning knowledge by integrating task ontologies with historical models, datasets, and experiment results.

- **User Interaction:**

The system should provide interfaces for users to input modeling goals and constraints, and it should offer model recommendations based on the ontology.[9]

3.3 Non-Functional Requirements:

- **Performance:**

The system should quickly generate machine learning pipelines, ensuring results within a reasonable time frame, e.g., under 10 seconds for typical use cases.

- **Scalability:**

The system should scale to support a growing library of tasks, algorithms, and datasets within the ontology.

- **Reliability:**

The system should ensure robust operation, with fallback mechanisms in case ontology data is incomplete or inconsistent.

- **Security:**

The system should safeguard sensitive dataset information and restrict access to ontology modification to authorized personnel only.

- **Maintainability:**

The task ontology and autonomous components should be modular and easy to update as new machine learning methods and tasks are introduced.[10]

3.4 System Analysis:

1. Problem Identification:

Traditional machine learning processes involve significant manual effort in model selection, parameter tuning, and workflow configuration. This creates inefficiencies and inconsistencies, especially when dealing with diverse tasks. There is also a lack of standardization in representing machine learning tasks, which hampers reusability and collaboration across projects.

2. Objectives:

- To develop an autonomous system that utilizes task ontologies to streamline machine learning workflow generation.
- To enable efficient and standardized representation of tasks, facilitating knowledge reuse.
- To enhance consistency, reduce human intervention, and improve model interpretability.

3. Requirements:

- Task Ontology Development: Create a comprehensive ontology to represent machine learning tasks, including preprocessing, model selection, and evaluation metrics.
- Automated Workflow Generation: Use the ontology to automatically generate workflows tailored to specific machine learning problems.
- Interoperability: Ensure compatibility with existing machine learning libraries and frameworks.
- User Interface: Provide tools for users to define tasks and constraints and visualize the generated workflows and results.[4]

4. Design of Optimal Solution:

- Architecture: Develop a modular system with components for ontology management, workflow generation, and execution. Use a centralized ontology repository for task representation.
- Data Flow: Establish workflows that transform user-defined tasks into machine learning pipelines via ontology-based reasoning.
- Predictive Modelling: Integrate algorithms and parameter optimization techniques that are dynamically selected based on the task ontology.[6]

5. Implementation Plan:

- Prototype Development: Build a prototype to validate ontology-driven workflow generation.
- Testing and Validation: Test the system on benchmark datasets and real-world tasks to assess performance and accuracy.
- Scalability: Incorporate mechanisms for adding new tasks, algorithms,

4. SYSTEM DESIGN

The system design focuses on creating a flexible, adaptive architecture that dynamically selects and executes tasks based on the input data and desired outcomes. It integrates stages like image preprocessing, feature extraction, object detection, and post-processing, with each stage defined as a task in a task ontology. This ontology enables autonomous decision-making, allowing the system to choose the appropriate models, methods, and workflows. The design ensures reusability, scalability, and efficient handling of diverse tasks, while allowing the system to adapt to new datasets and techniques without manual intervention.

4.1 Architecture Diagram

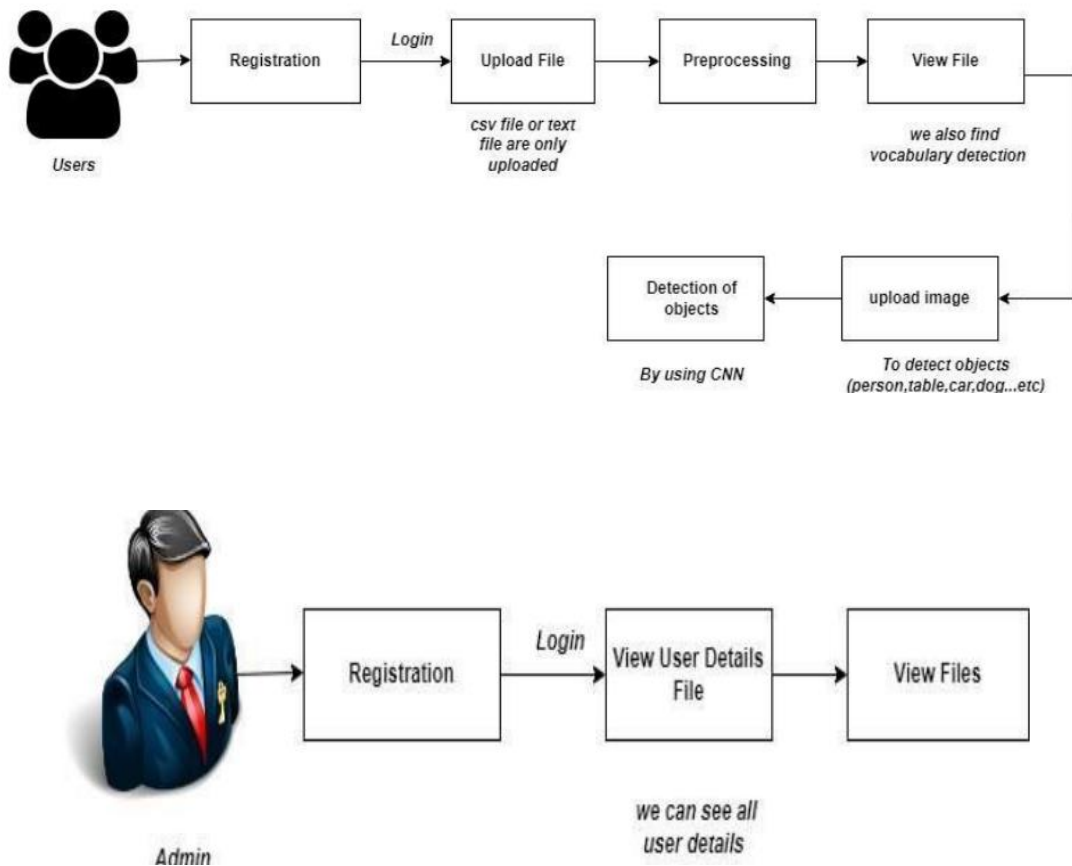


Fig 4.1. Project Architecture[8]

The system architecture outlined in the image represents a machine learning-based system that autonomously handles object detection and vocabulary processing using task automation and ontology-driven models. The system is designed to facilitate interaction between users and an administrator, supporting functionalities such as file uploads, preprocessing, object detection.

User Login:

1. Registration:

- This is the entry point for new users in the system.
- Users create an account by providing required information (e.g., username, email, and password).
- The registration process ensures only verified individuals can use the platform, thus maintaining

2. Login

- Once registered, users log in with their credentials to access the system.
- The login process verifies user identity, ensuring unauthorized access is prevented.

3. Upload File:

- Users can upload files, specifically in CSV or text format, for further analysis.
- These files may include data related to specific tasks such as natural language processing (NLP) or object recognition.
- Example use cases:
- A CSV file might contain data for training a vocabulary model.
- A text file could be used for semantic analysis or object detection tasks.

4. Preprocessing:

- After uploading, the system preprocesses the data.

Preprocessing Tasks:

- Cleaning: Removing noise, irrelevant data, or special characters.
- Tokenization: Splitting text into words or phrases for analysis.
- Feature Extraction: Identifying key features in the data for vocabulary detection or object recognition. This is the entry point for new users in the system.
- Users create an account by providing required information (e.g., username, email, and password).
- The registration process ensures only verified individuals can use the platform, thus maintaining

5. View File:

Users can view their processed files to verify the results.

Vocabulary Detection:

- The system analyzes text data to identify vocabulary terms, calculate word frequencies, and detect linguistic patterns.
- This is useful for text mining, sentiment analysis, or machine translation tasks

6. Detection of Objects:

The system supports image upload for object detection.

- Using Convolutional Neural Networks (CNN):
- CNNs, a type of deep learning model, analyze the image to detect and classify objects such as persons, tables, cars, and dogs.
- This involves:
- Extracting features like edges, shapes, and textures.

Admin Module

1.Registration:

- The administrator registers separately to gain control over the system.
- Admin privileges allow managing user accounts, files, and overseeing the system's activities.

2.Login:

- Admins log in with their credentials to access the admin dashboard.
- This ensures administrative tasks are performed securely.

3.View User Details

- Admins can access detailed profiles of all registered users.
- Information available might include:
- Username, email, and registration date.
- Uploaded files and processing activity logs.

4.View Files

- This feature is essential for tracking user engagement and resolving issues if needed
- The administrator can view all files uploaded by users.
- This helps monitor the type of data being processed, ensuring it complies with system guidelines.
- It also enables admins to review output results if users encounter errors.

4.2UML Diagrams

4.2.1 Use Case Diagram for Task-Centric Autonomous Machine Learning Modeling Approach.

A Use Case Diagram visually represents the interactions between users, admins, and the system, highlighting their roles and the system's functionalities. It defines how users and admins perform tasks like submitting requests, managing accounts, and overseeing system operations within an autonomous machine learning framework.

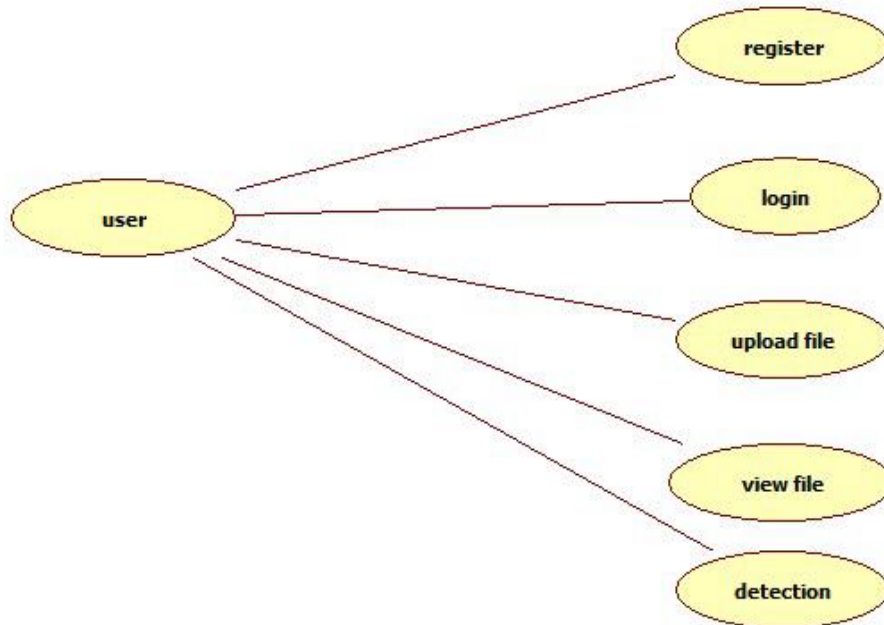


Fig 4.2.1.1

- The Use Case Diagram involves two main actors: users and admins.
- Users interact with the system to perform tasks such as submitting requests to the machine learning model, tracking task statuses, viewing results, and managing their accounts.
- These interactions are limited to functionalities that align with their role, focusing on accessing and utilizing the model's outputs.

Admins, on the other hand, have a more comprehensive role in managing the system. They can manage user accounts, monitor the system's performance, define or update the task ontology, prioritize or terminate tasks, and access logs for auditing and debugging purposes. Admins also oversee access control to ensure that only authorized users can interact with the system.

The system includes interconnected use cases such as validating tasks upon submission, ensuring compliance with the task ontology, and allowing admins to view logs while monitoring system.

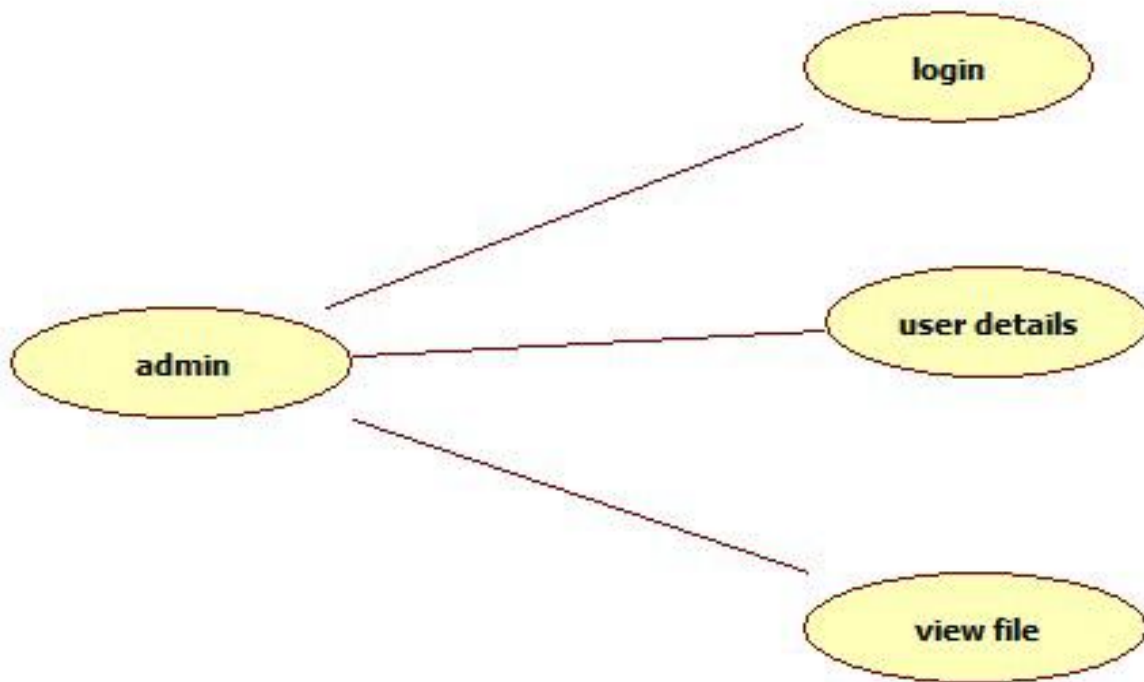


Fig 4.2.1.2

1. Actor:

The admin is the primary actor, responsible for interacting with the system to manage various tasks.

2. Use Cases:

Login: The admin logs into the system to authenticate and gain access to functionalities.

User Details: The admin can view or manage user information, such as account details, roles, or permissions.

View File: The admin has the capability to access and review files, likely for auditing or managing tasks submitted by users.

3. Relationships:

The lines connecting the admin to each use case represent direct interactions between the admin and these system functionalities.

This diagram focuses on admin-specific use cases and highlights their critical role in managing users and files within the system.

4.2.2 Sequence Diagram for Task-Centric Autonomous Machine Learning Modeling Approach

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows, as parallel vertical lines ("lifelines"), different processes or objects that live simultaneously, and as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

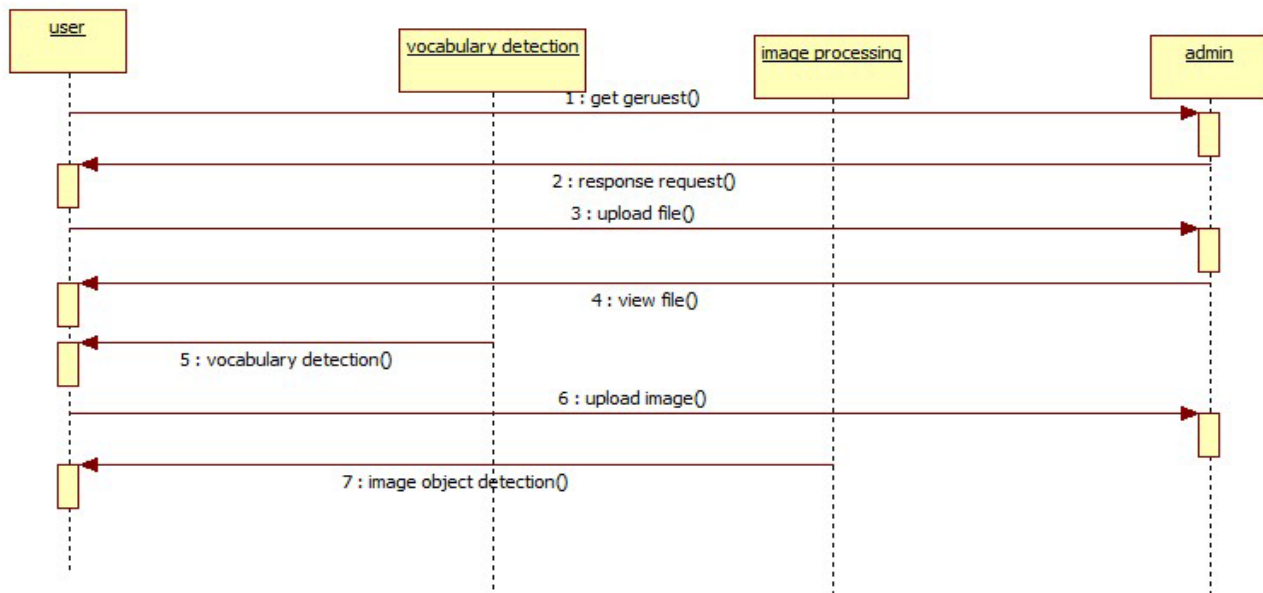


Fig 4.2.2.1

Key Components:

1. Actors and Components:

- User: The individual interacting with the system.
- Vocabulary Detection: A module that processes vocabulary-related tasks.
- Image Processing: A module handling image-based operations.
- Admin: An administrative entity that interacts with the system.

2. Lifelines: Each box represents an actor or component, and its dashed line shows its active timeline in the sequence.

3. Messages: The arrows represent interactions or method calls between the components. The labels on the arrows indicate the function or action invoked.

Flow Explanation:

Step 1: The User sends a get request() to the Vocabulary Detection component.

Step 2: The Vocabulary Detection module processes the request and sends a response request() back to the user.

Step 3: The User uploads a file using the upload file() method to the Vocabulary Detection

Step 4: The Vocabulary Detection module allows the user to view file().

Step 5: A vocabulary detection process is initiated within the module.

Step 6: The User uploads an image (upload image()) to the Image Processing module

Step 7: The Image Processing module runs an image object detection() operation.

4.2.3 Activity Diagram for Task-Centric Autonomous Machine Learning Modeling Approach.

Activity diagrams are used to model the Dynamic aspects of a system focusing on the activities or actions or the relationships that occur between them

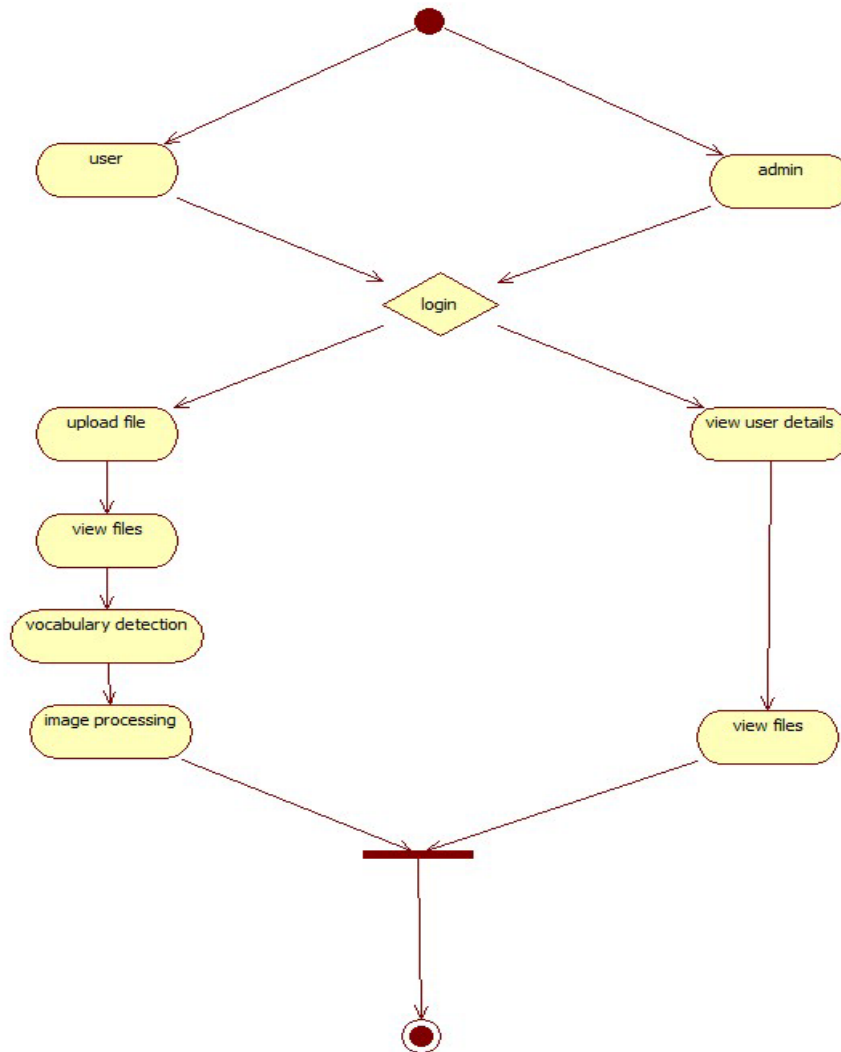


Fig 4.2.3.

The activity diagram represents the workflow of a system involving two primary actors:

- users
- admins

It outlines the sequence of tasks and decisions within the system.

Login Decision

The process begins with a login decision point, where the system determines if the actor is a user or an admin. Based on this decision, the workflow branches into separate paths.

User Actions

If the actor is a user, they can perform the following tasks in sequence:

1. Upload File: Users can upload their files to the system.
2. View Files: Uploaded files can be accessed and managed.
3. Vocabulary Detection: The system processes the files for vocabulary-related tasks.
4. Image Processing: Users can further process the files for image-based tasks.

Admin Actions

For admins, the workflow includes:

1. View User Details: Admins can monitor and access user-specific information.
2. View Files: Admins can view all files in the system for management purposes.

4.2.4 Class Diagram for Task-Centric Autonomous Machine Learning Modeling Approach.

A class diagram is a type of UML (Unified Modeling Language) diagram used to visually represent the structure of a system by modeling its classes, attributes, methods, and relationships.



Fig 4.2.4.1

The class diagram represents a system with two main classes: User and Admin, highlighting their roles and interactions. The User class includes methods such as `login()`, `uploadFiles()`, `viewFiles()`, and `detection()` for basic functionalities like logging in, uploading, and viewing files. The Admin class, which inherits from the User class, adds specialized functionalities such as `viewAllUsers()` and `viewFiles()`, enabling system-wide management capabilities like overseeing users and accessing files. The dashed line with a triangular arrow denotes an inheritance relationship, indicating that Admins possess all User functionalities alongside their unique privileges. This structure reflects a role-based access system, ensuring modularity and extensibility in design.

4.2.5 Deployment Diagram

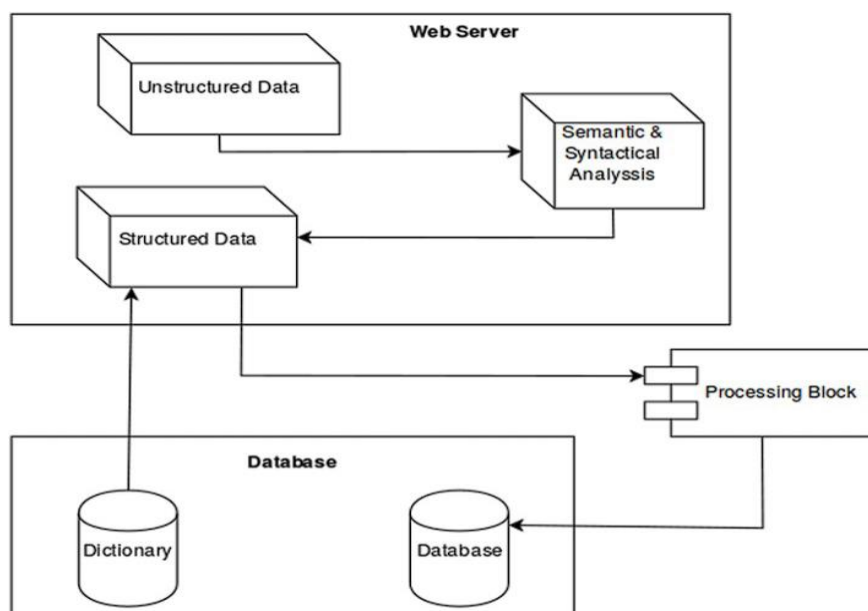


Fig. 4.2.5.1 Deployment Diagram

- This deployment diagram illustrates the architecture of a system designed for processing and analyzing data, which aligns with the requirements of an autonomous machine-learning model using task ontology. At its core, the diagram shows how unstructured data flows into a web

server for semantic and syntactical analysis. This analysis converts raw, unstructured data into structured formats, which are essential for machine learning algorithms that rely on well-organized inputs to generate meaningful insights. The semantic and syntactical analysis ensures that the data is properly parsed, contextualized, and ready for further processing.

- Once the data is structured, it is stored in a database system consisting of two main components: a dictionary and the database itself. The dictionary component likely serves as a knowledge base or reference guide for task ontology. This aids the system in understanding domain-specific terms, relationships, and contextual meanings that are crucial for autonomous decision-making. The database acts as the central repository for all processed data, ensuring efficient storage, retrieval, and organization, which supports the machine learning model's training and inference phases.
- The processing block in the diagram represents the core computational unit where advanced tasks such as model training, task-specific inference, and decision-making occur. The connection between the structured data and the processing block implies that the machine learning model uses this data as input for training and deployment. This block likely implements algorithms tailored for autonomous learning, task recognition, and ontology-based reasoning, enabling the system to adapt dynamically to new tasks and datasets.
- Finally, the integration of the web server, database, and processing block highlights the system's modularity and scalability. The web server handles data preprocessing and communication, while the database ensures robust data management, and the processing block provides computational power for model operations. This architecture is well-suited for autonomous machine learning, as it leverages task ontology to interpret and respond to tasks flexibly and intelligently, making it applicable to various domains such as robotics, natural language processing, and autonomous systems.

5.IMPLEMENTATION

5.1 Explanation of Key Functions:

1. Autonomous Model Generation

- Task Understanding: Leverages task ontology to define machine learning objectives, input-output relationships, and constraints.
- Model Selection: Automatically identifies the most suitable algorithms (e.g., regression, classification, clustering) for the given task using the structured knowledge from the ontology.
- Pipeline Construction: Assembles end-to-end workflows including data preprocessing, feature engineering, model training, and evaluation autonomously.

2. Task Ontology Integration

- Knowledge Representation: Encodes domain knowledge and task-specific requirements into a structured format to guide machine learning processes.
- Ontology Mapping: Maps input data to predefined task categories, ensuring alignment between data characteristics and modeling approaches.
- Semantic Consistency: Ensures consistent interpretation of terms and relationships across tasks, improving decision accuracy.

3. Performance Optimization

- Automated Hyperparameter Tuning: Utilizes heuristic or search-based optimization techniques (e.g., Grid Search, Bayesian Optimization) for improving model performance.
- Feedback Mechanism: Iteratively refines models based on performance metrics and error analysis.

4. Usability and Accessibility

- User-Friendly Interface: Provides an interactive dashboard for users to visualize model results and explore task-specific insights.
- Explainability: Offers explanations for chosen models, features, and predictions, enhancing transparency and trust in autonomous decision-making.

5.2 Methods of Implementation

1. Data Collection

- **APIs and Integrations:** Collect structured and unstructured data from diverse sources based on predefined task ontology attributes.
- **Domain-Specific Datasets:** Utilize domain-relevant datasets aligned with the ontology for training and testing.

2. Ontology Design and Mapping

- **Ontology Creation:** Design a hierarchical ontology with domain-specific tasks, categories, and relationships (e.g., using OWL or RDF formats).
- **Data Preprocessing:** Map raw input data to ontology categories, ensuring semantic relevance and consistency.

3. Autonomous Workflow Construction

- **Preprocessing Module:** Automate data cleaning, normalization, and feature engineering steps based on task-specific requirements.
- **Model Selection Module:** Implement an ontology-driven recommender system to suggest appropriate algorithms and techniques.
- **Execution Framework:** Use libraries such as TensorFlow, Scikit-learn, and SpaCy for model training, evaluation, and deployment.

4. Model Evaluation and Optimization

- **Evaluation Metrics:** Employ task-specific evaluation metrics (e.g., accuracy, F1-score, RMSE) for iterative refinement.
- **Performance Monitoring:** Integrate tools for real-time performance monitoring to ensure model reliability over time.

5. Deployment and Integration

- **API Development:** Expose the system as APIs for seamless integration with other applications.
- **Visualization Tools:** Use visualization frameworks like Dash or Tableau for presenting insights derived from the models.

6. Feedback and Continuous Improvement

- **Feedback Loop:** Incorporate user feedback and new data to enhance the task ontology and improve model selection.
- **Automated Retraining:** Periodically retrain models using updated datasets and evolving task definitions

The Autonomous Machine Learning Model for Object Detection and Text Analysis system consists of several key modules that work together to process and analyze the data, ultimately identifying objects and analyzing the content based on task ontologies. Each module plays a critical role in ensuring accurate object detection and text analysis

The following are the 8 modules of our project:

- Data Collection
- Preprocessing
- Object Detection
- Text Analysis
- Task Ontology Integration
- Model Training
- Evaluation and Output
- Visualization and User Interface [5]

5.3.1MODULES OVERVIEW

The system processes data through modules for collection, preprocessing, and analysis. Object detection uses YOLO, while NLP techniques extract insights from text, with results structured and presented via a user-friendly interface.

1. **Data Collection:** This module is responsible for gathering images and text data, typically from user inputs or pre-collected datasets. The images and texts are then formatted into a form suitable for processing, which may include labeling or other relevant metadata.
2. **Preprocessing:** This step involves preprocessing the images and text data. For images, it includes normalization, resizing, and augmentation. For text, preprocessing steps include tokenization, lemmatization, and stopword removal. The goal is to clean the data and prepare it for analysis.
3. **Object Detection:** In this module, machine learning models like Convolutional Neural Networks (CNNs) or other object detection algorithms (YOLO, Faster R-CNN) are employed to detect objects within images. The model identifies and labels objects based on predefined classes.
4. **Text Analysis:** This module uses NLP techniques to analyze text data. This may include sentiment analysis, keyword extraction, and other NLP tasks like named entity recognition (NER) to identify important details such as skills, qualifications, or job titles.
5. **Task Ontology Integration:** Task ontologies are integrated to map the extracted information

to a predefined task structure. This helps categorize the objects and text in a way that aligns with the intended purpose of the analysis, such as matching candidate skills with job roles.\

6. **Model Training:** Machine learning models are trained on annotated datasets to classify objects and analyze text. The system learns to recognize patterns in both images and text data through supervised or unsupervised learning methods.
7. **Evaluation and Output:** This module evaluates the model's performance and produces output based on the analysis. It ranks the candidates, images, or detected objects and generates results for visualization or further action.
8. **Visualization and User Interface:** The results are visualized in a user-friendly interface, which may include dashboards, graphs, or tables. The user can interact with the interface to explore results, refine analyses, or view insights from the detected objects and analyzed text

5.4 Source Code

```
from django.shortcuts import render, redirect
from django.contrib import messages
from random import randint
from nltk.corpus import wordnet as wn
import cv2
import os
import numpy as np

# URL Routing
urlpatterns = [
    url(r'^admin/', admin.site.urls), # Admin site URL
    url(r'^$', index, name="index"), # Home page
    url(r'^registration/', registration, name="registration"), # User registration
    url(r'^adminlogin/', adminlogin, name="adminlogin"), # Admin login page
]

# Admin functionalities
def adminloginaction(request):
    """Admin login validation"""
    if request.method == "POST":
        login = request.POST.get('username') # Get admin username
        pswd = request.POST.get('password') # Get admin password
        if login == 'admin' and pswd == 'admin': # Validate credentials
            return render(request, 'admin/adminhome.html') # Redirect to admin home

def activateuser(request):
    """Activate user account with a random 8-digit activation key"""
    if request.method == 'GET':
        usid = request.GET.get('usid') # Get user ID from request
        authkey = random_with_N_digits(8) # Generate an 8-digit activation key
        registrationmodel.objects.filter(id=usid).update(authkey=authkey, status='activated')

def random_with_N_digits(n):
    """Generate a random number with n digits"""
    return randint(10**(n-1), (10**n)-1)

# User registration
def registration(request):
    """Handle user registration"""
    if request.method == 'POST':
        form = registrationform(request.POST) # Initialize registration form
```

```

if form.is_valid():
    form.save() # Save user data
    messages.success(request, 'Successfully registered') # Show success message
    return redirect('trainer') # Redirect to trainer page

# Vocabulary extraction with NLTK
def findvocabulary(request):
    """Extract vocabulary from a text file and fetch definitions using WordNet"""
    if request.method == "GET":
        file = request.GET.get('id') # Get file path from request
        raw = open(settings.MEDIA_ROOT + '/' + file).read() # Read file content
        tokens = word_tokenize(raw) # Tokenize file content
        words = set(w.lower() for w in nltk.corpus.words.words()) # Get valid words
        voc = set(tokens) & words # Extract vocabulary
        katti = {x: wn.synsets(x)[0].definition() for x in voc if wn.synsets(x)} # Fetch
definitions

# YOLO-based object detection
def detection(request):
    """Perform object detection using YOLO"""
    if request.method == "POST":
        images = request.FILES.get('imgfile') # Get uploaded image
        img = Image.open(images) # Open the uploaded image
        img = np.array(img) # Convert PIL image to NumPy array (required for OpenCV)

        # Load YOLO model files
        labelsPath = os.path.sep.join(["yolo-coco/coco.names"]) # Load YOLO class labels
        weightsPath = os.path.sep.join(["yolo-coco/yolov3.weights"]) # Load YOLO
weights
        configPath = os.path.sep.join(["yolo-coco/yolov3.cfg"]) # Load YOLO config

        # Load YOLO model
        net = cv2.dnn.readNetFromDarknet(configPath, weightsPath) # Initialize YOLO
model
        layer_names = net.getLayerNames()
        output_layers = [layer_names[i - 1] for i in net.getUnconnectedOutLayers()] #
YOLO output layers

        # Process detections
        for output in layerOutputs:
            for detection in output:
                scores = detection[5:] # Extract class probabilities
                class_id = np.argmax(scores) # Get class ID with the highest confidence
                confidence = scores[class_id] # Get corresponding confidence
                if confidence > 0.5: # Filter weak detections

```

```
# Scale bounding box coordinates to original image dimensions
box = detection[0:4] * np.array([W, H, W, H])
(centerX, centerY, width, height) = box.astype("int")
x = int(centerX - (width / 2))
y = int(centerY - (height / 2))

# Append to lists
boxes.append([x, y, int(width), int(height)])
confidences.append(float(confidence))
class_ids.append(class_id)

# Apply Non-Maximum Suppression (NMS) to suppress weak overlapping boxes
idxs = cv2.dnn.NMSBoxes(boxes, confidences, 0.5
```


6.TESTING AND VALIDATION

6.1 Testing Process

Test strategy and approach

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

Test objectives

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

Features to be tested

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

Integration Testing

- Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.
- The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

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

Acceptance Testing

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

Test Results: All the test cases mentioned above passed successfully.

6.2 Test Cases

1. User Registration

- Description: Test the ability for a user to register successfully.
- Expected Result: The user should be registered successfully.
- Actual Result: Pass.
- Remarks (if fails): If the user is not registered, review input details and system logs.

2. User Login

- Description: Validate the login functionality with correct credentials.
- Expected Result: Upon entering the correct username and password, the user should access the valid page.
- Actual Result: Pass.
- Remarks (if fails): If the username or password is incorrect, prompt the user with an appropriate error message.

3. Admin Rights Validation

- Description: Verify if the admin rights are correctly accepted for a user.
- Expected Result: Admin rights should be granted as expected.
- Actual Result: Pass.
- Remarks (if fails): Ensure the user is registered and authorized for admin access.

4. User File Upload

- Description: Check the functionality for uploading files.
- Expected Result: The user should be able to choose and upload files successfully.
- Actual Result: Pass.
- Remarks (if fails): If the user cannot select or upload files, verify file type and size constraints.

5. Vocabulary Detection

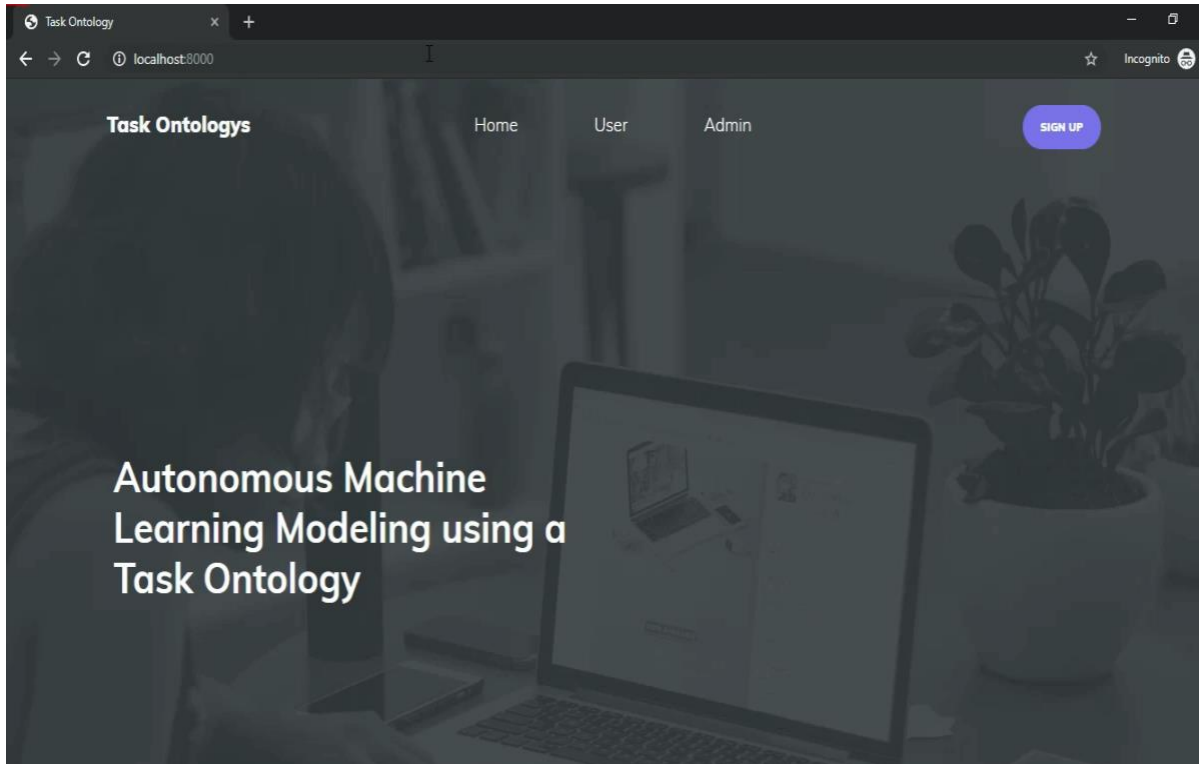
- **Description: Ensure the system can detect and verify vocabulary** accurately.
- Expected Result: All vocabulary should be detected and verified successfully.
- Actual Result: Pass.
- Remarks (if fails): If vocabulary detection fails, check the system's dictionary and configuration.

6. Image Processing

- **Description: Validate the image processing functionality.**
- Expected Result: The system should process and verify images as expected.
- Actual Result: Pass.
- Remarks (if fails): If image processing is unavailable, inspect system logs for errors or missing dependencies.

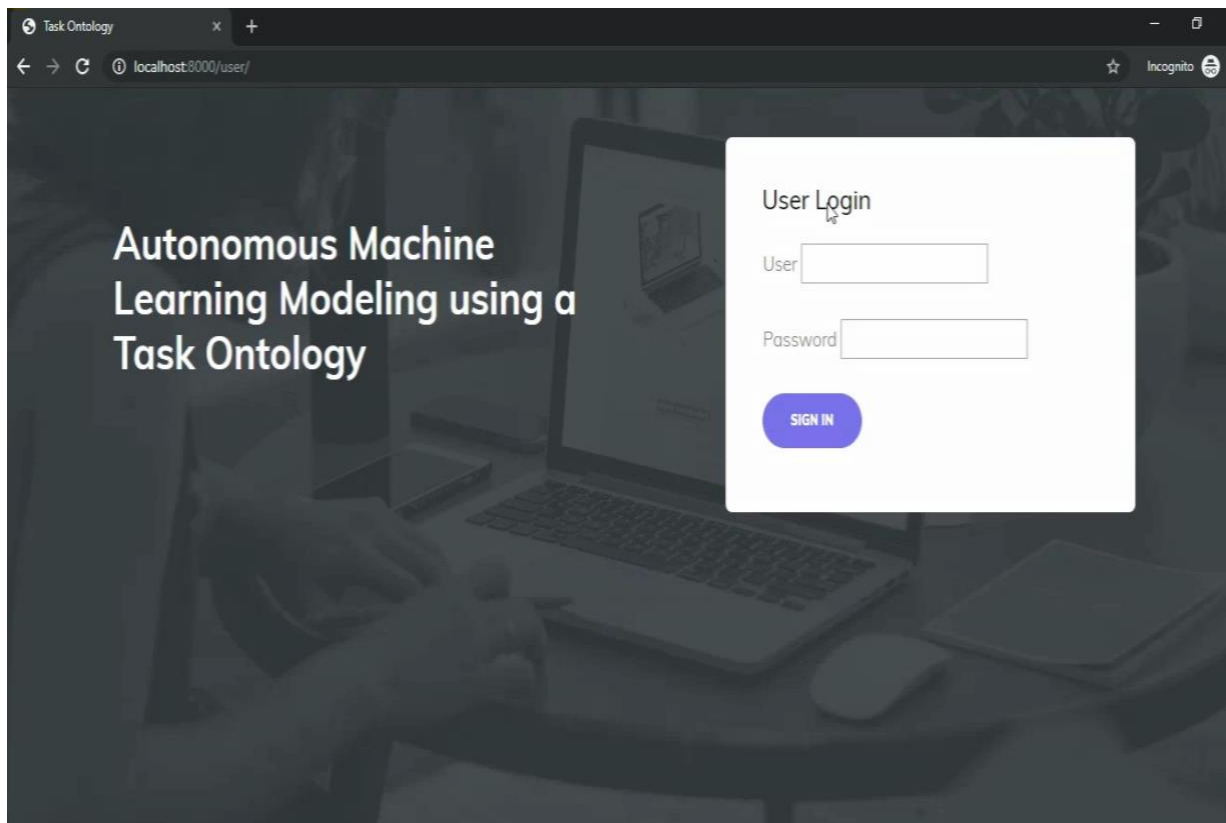
7.Results and Performance Evaluation

Home Page



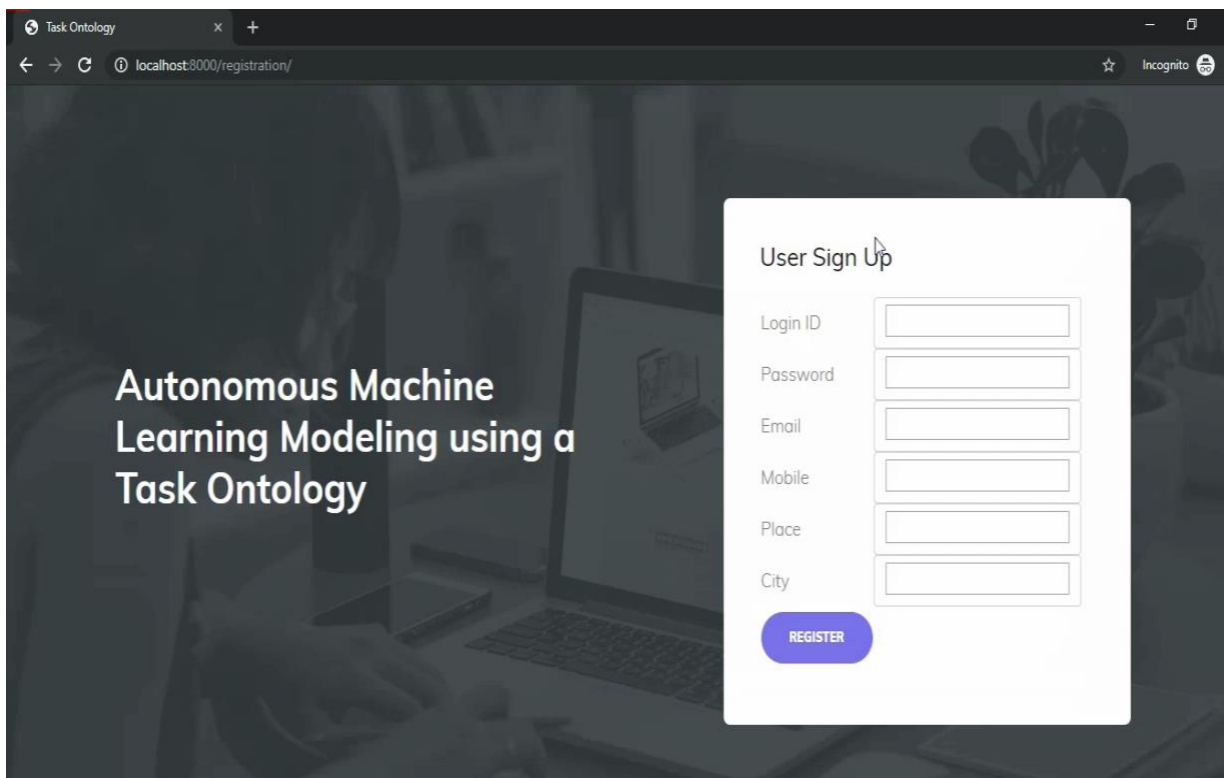
The Home Page contains user and admin access. This Home Page provides the user to sign up into the platform

User Login



This interface enables the user to log into the platform with their username and password respectively.

User Register

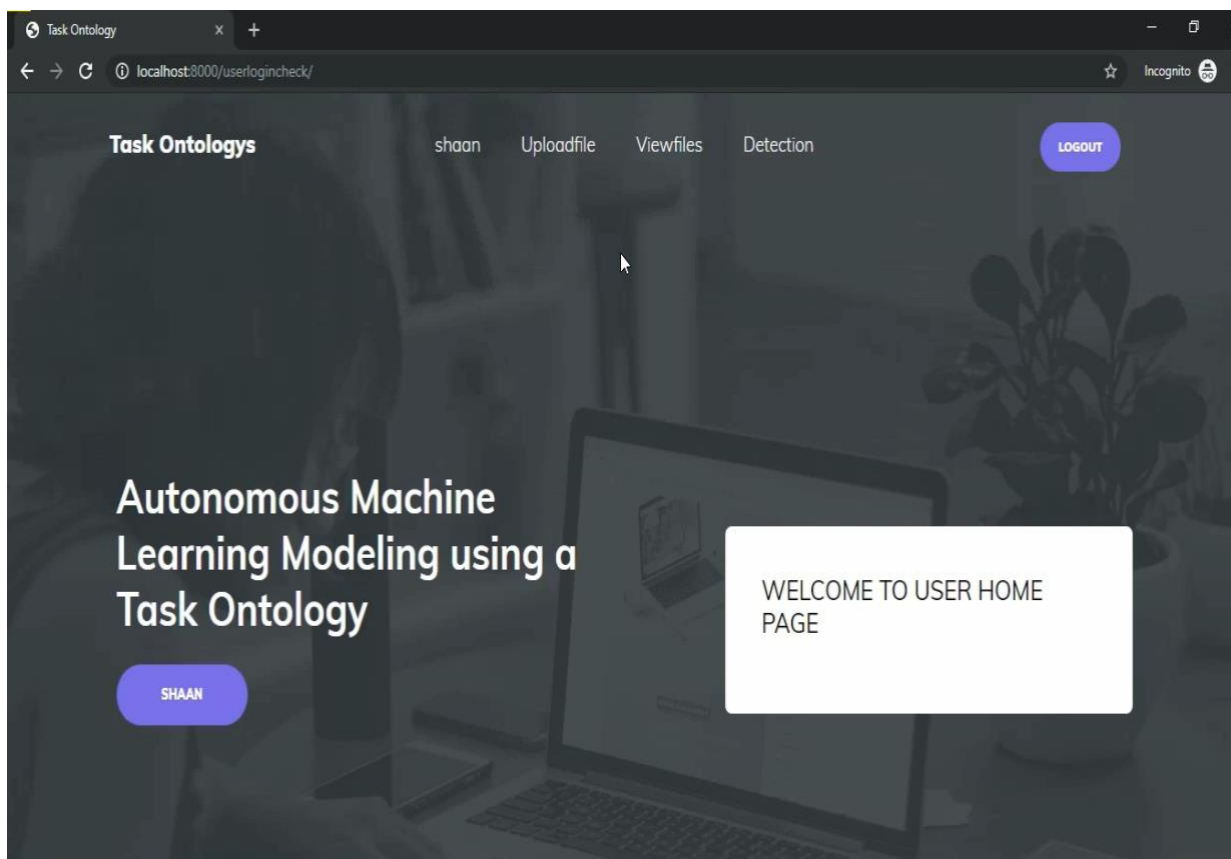


The screenshot shows a web browser window with the address bar displaying 'localhost:8000/registration/'. The page features a dark background with a laptop and a person's hands. A white modal box titled 'User Sign Up' is centered on the screen. It contains the following fields and buttons:

- Login ID
- Password
- Email
- Mobile
- Place
- City
- REGISTER (blue button)

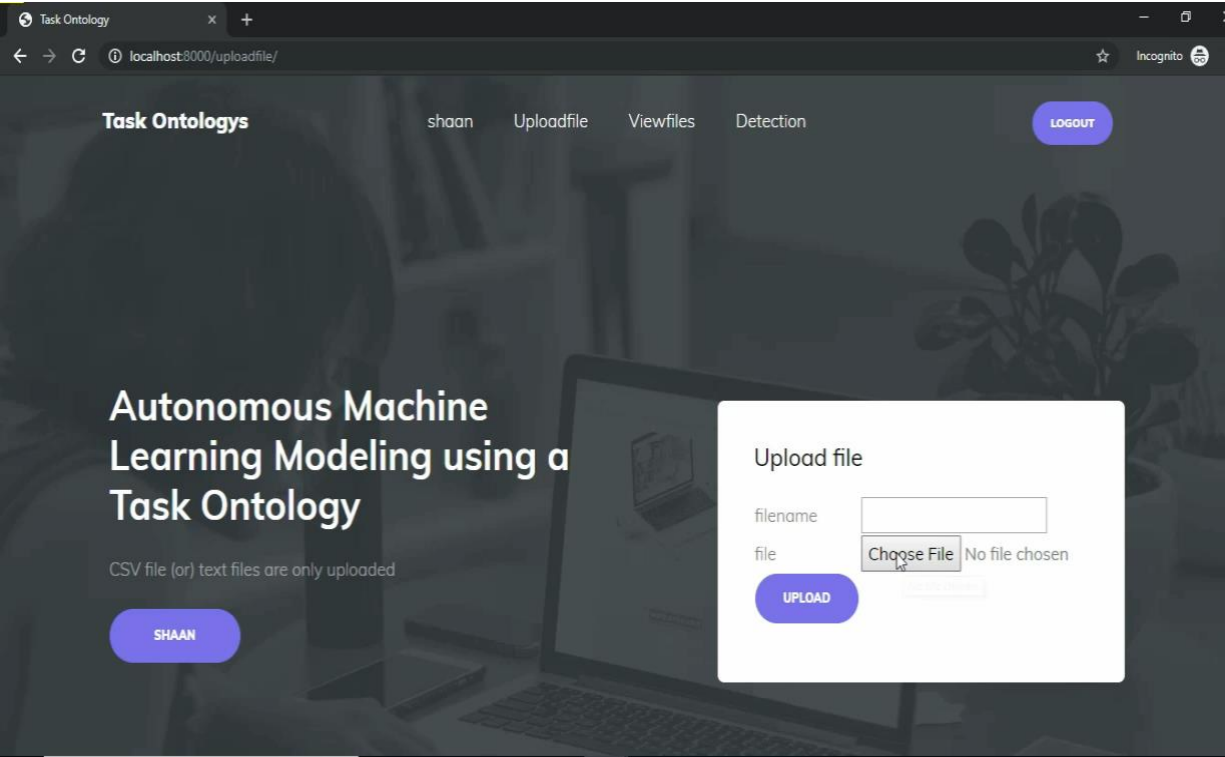
This interface enables user to sign up into platform by giving necessary information to register their new account.

User Home Page



This interface is the User Home Page, which allows the user to get into the platform

User Upload File



This interface allows user to upload the necessary file,to fetch the required results

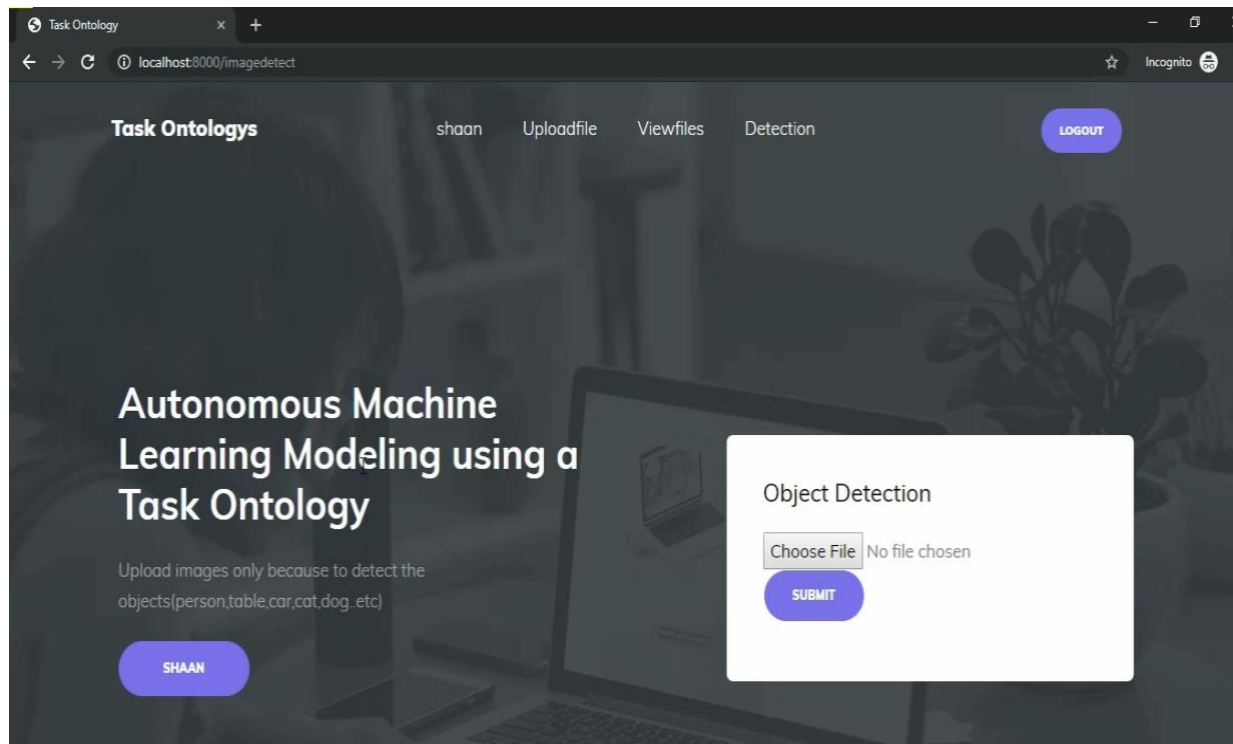
User View details

A screenshot of the 'Task Ontology' web application showing the 'Users Upload Files' page. The browser's address bar shows 'localhost:8000/viewuserfiles/'. The page title is 'Users Upload Files'. Below the title is a table with 5 columns: 'S.No', 'Filename', 'File', 'Download', and 'FindVocabulary'. The table contains 7 rows of data. Each row has a 'VIEW' button in the 'Download' column and a 'FIND VOCABULARY' button in the 'FindVocabulary' column. The background of the application shows a person working on a laptop.

S.No	Filename	File	Download	FindVocabulary
1	Document	files/pdfs/meghana_fgsevgM.txt	VIEW	FIND VOCABULARY
2	pdf	files/pdfs/Research_on_Application_of_Artificial_Intelligence_in_Medical_Education.pdf	VIEW	FIND VOCABULARY
3	Data	files/pdfs/1_Amalgam_Cloud_Methodology_for_Safe_Accredited_Deduplication.docx	VIEW	FIND VOCABULARY
4	Text	files/pdfs/project.txt	VIEW	FIND VOCABULARY
5	alzebra	files/pdfs/albarrey.txt	VIEW	FIND VOCABULARY
6	namitha	files/pdfs/hagalsmi.txt	VIEW	FIND VOCABULARY
7	springfiled.txt	files/pdfs/springfield.txt	VIEW	FIND VOCABULARY

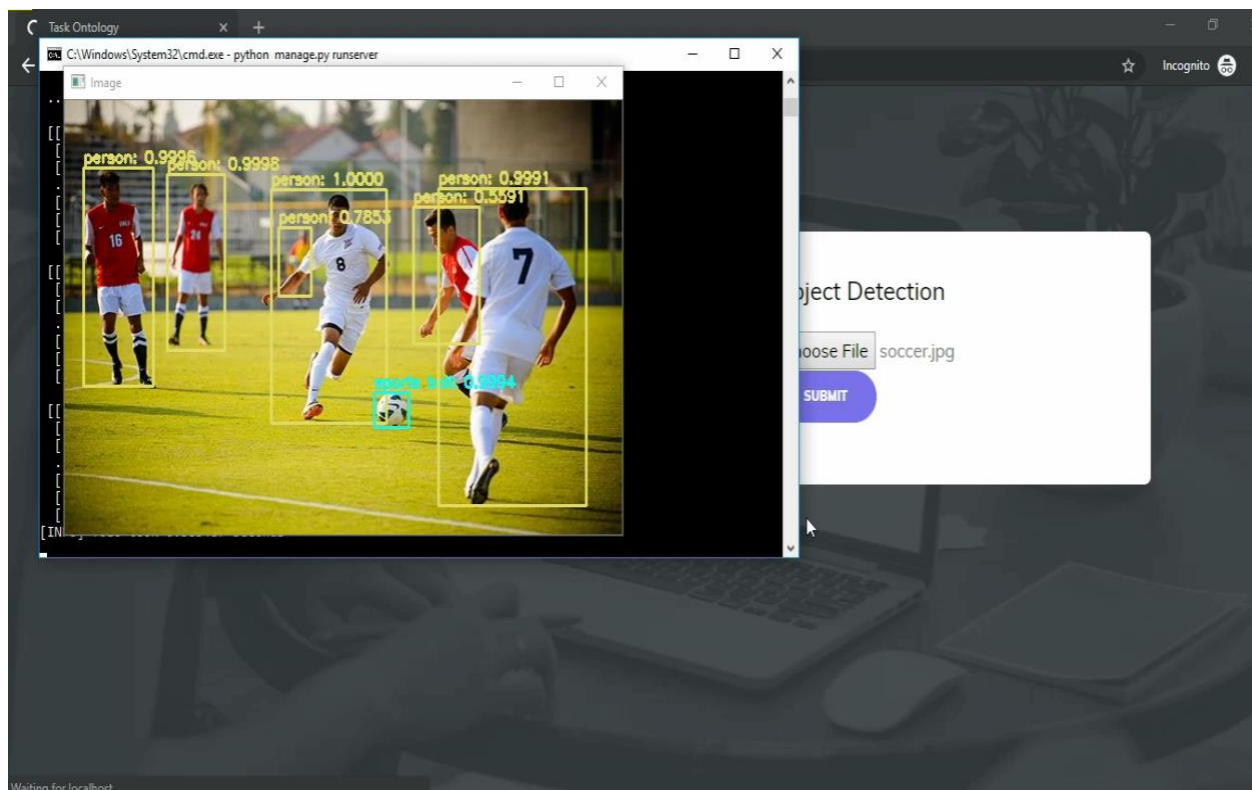
This interface gives the review of the files uploaded by the user.

Detection of Images



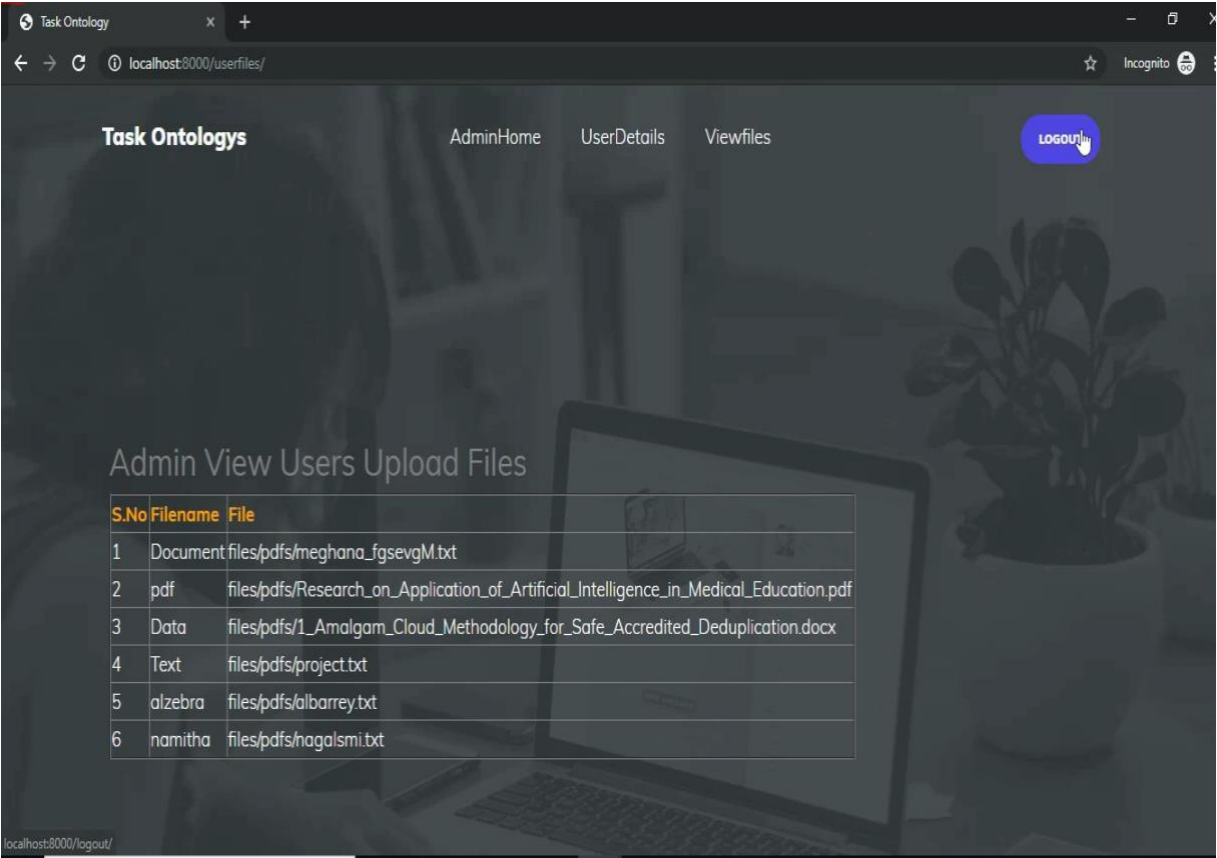
This Interface allows the user to upload images in order to detect the items and demographics in it.

Image Detection



This interface gives the results by identifying each object in the given file.

Admin Dashboard



This interface provides the admin dashboard, where admin can access the user inputs.

8.CONCLUSION AND FURTHER ENHANCEMENT

8.1 CONCLUSION:

The project effectively integrates object detection and text analysis to create a versatile, intelligent system. Using the YOLO algorithm for object detection, the model accurately identifies and localizes objects within images, while Natural Language Toolkit (NLTK) facilitates sophisticated text analysis. It demonstrates the seamless fusion of visual and textual data processing, offering a robust solution for real-world applications such as automated surveillance, content moderation, and intelligent data extraction.

The architecture leverages state-of-the-art technologies like Django for deployment, ensuring scalability and user interaction. The integration of autonomous decision-making through task ontology establishes a framework that is both adaptive and capable of handling complex datasets with minimal human intervention.

This project highlights the potential of combining machine learning techniques to address multifaceted challenges, paving the way for further advancements in artificial intelligence and autonomous systems. The successful implementation validates the practicality of the approach and sets the stage for future enhancements, including real-time processing and integration with IoT devices for broader applications.

8.2 FUTURE SCOPE:

Future enhancements for the project "Autonomous Machine Learning Modeling using a Task Ontology" could include the integration of more advanced machine learning techniques, such as reinforcement learning, to allow for autonomous adaptation to new tasks. Additionally, incorporating multi-modal data handling would enable the system to process and combine diverse data types, improving model accuracy. Cloud-based deployment would scale the system efficiently, while explainability tools like LIME or SHAP would increase trust in its predictions. Finally, incorporating AutoML for model tuning and optimization could streamline the process for non-experts.

9. REFERENCES

- [1]Kyoung Soon Hwang; Ki Sun Park; Sang Hyun Lee; Kwang Il Kim; Keon Myung Lee, “Autonomous Machine Learning Modeling using a Task Ontology” 2018 Joint 10th International Conference on Soft Computing and Intelligent Systems (SCIS) and 19th International Symposium on Advanced Intelligent Systems (ISIS)
- [2] Bekir Sahin; Anis Yazidi; Dumitru Roman; Ahmet Soylu,” Ontology-Based Fault Tree Analysis Algorithms in a Fuzzy Environment for Autonomous Ships”,2021 pp- 2169-3536
- [3] Li, D. “ Machines Learn Better with Better Data Ontology: Lessons from Philosophy of Induction and Machine Learning Practice.”*Minds & Machines* **33**, 429–450 (2023).
- [4] Ghanadbashi, S., Golpayegani, F,”Using ontology to guide reinforcement learning agents in unseen situations”. *Appl Intell* **52**, 1808–1824 (2022).
- [5] S. Kanjana, S. Maleerat, “Ontology Knowledge-Based Framework for Machine Learning Concept”, iiWAS '16 Proceedings of the 18th International Conference on Information Integration and Webbased Applications and Services, pp. 50-53.
- [6] Mitsuru, S. Kazuhisa, K. Osamu, M. Riichiro, “Task ontology: Ontology for building conceptual problem solving models”, In proceeding of ECAI98 Workshop on Applications of Ontologies and Problem-Solving model, pp.126-133, ECA.
- [7] V. Joaquin, S. Larisa, “Expose: An Ontology for Data Mining Experiments”, Third Generation Data Mining Workshop at ECML PKDD.
- [8] A. Martín, B. Paul Barha, C. Jianmin, C. Zhifeng, D. Andy, D. Jeffrey, et al., “TensorFlow: A System for Large-Scale Machine Learning," Operating Systems Design, and Implementation, Vol. 16, pp. 265-283, 2016.
- [9] Thomas R. Gruber, “ A Translation Approach to Portable Ontology Specifications” Knowledge System Laboratory, Technical Report KSL 92-71.
- [10] A. F. Martins, R. A. F. De, “Models for Representing Task Ontologies”, Proceeding of the 3rd

Workshops on Ontologies and their Application.

[11] Gustavo Correa Publio, Diego Esteves, Agnieszka Ławrynowicz, Panče Panov, Larisa Soldatova, Tommaso Soru, Joaquin Vanschoren, Hamid Zafar, “ ML Schema: Exposing the Semantics of Machine Learning with Schemas and Ontologies”.

[12] Guarino, N., Oberle, D., Staab, S. (2009). What Is an Ontology? In: Staab, S., Studer, R. (eds) Handbook on Ontologies. International Handbooks on Information Systems. Springer, Berlin, Heidelberg.

[13] Evgeny Kharlamov,Dag Hovland,Martin Georg Skjæveland,Dimitris Bilidas, “Ontology based data access in statoil”.

[14] Y. Alfaifi, "Ontology Development Methodology: A Systematic Review and Case Study," *2022 2nd International Conference on Computing and Information Technology (ICCIT)*, Tabuk, Saudi Arabia, 2022, pp. 446-450, doi: 10.1109/ICCIT52419.2022.9711664.

[15] A. Castro, V. A. Villagr , P. Garc a, D. Rivera and D. Toledo, "An Ontological-Based Model to Data Governance for Big Data," in *IEEE Access*, vol. 9, pp. 109943-109959, 2021, doi: 10.1109/ACCESS.2021.3101938