A Project Report on

PORT AQUA DEMAND ANALYSIS AND RECOMMENDATION SYSTEM

Submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

Submitted by

D. Lakshmi Neha	**(20MH1A0577)**
Y. Praveen Kumar	**(20MH1A05C4)**
CH. Shyam Bhaskar	**(20MH1A0575)**
V. Ram Ganesh	**(20MH1A05C6)**

Under the esteemed supervision of

Mr. Dr. B. V. Rama Krishna M.Tech.,Ph.D.

Professor

Department of Computer Science & Engineering

ADITYA COLLEGE OF ENGINEERING (A)

Approved by AICTE,Permenantly Affiliated to JNTUK, Accredited by NBA & NAAC Recocginzed by UGC under Section 2(f) and 12(B) of UGC act, 1956 Aditya Nagar, ADB Road, Surampalem , Kakinada District, A.P 533437

(2020 - 2024)

ADITYA COLLEGE OF ENGINEERING

An AUTONOMOUS Institution

Approved by AICTE, Permanently Affiliated to JNTUK, Accredited by NBA & NAAC
Recognized by UGC under Sections 2(f) and 12(B) of UGC Act, 1956
Aditya Nagar, ADB Road, Surampalem - 533 437, E.G.Dist., Ph: 99631 76662.

Institute Vision, Mission

INSTITUTE VISION:

To induce higher planes of learning by imparting technical education with

- International standards
- Applied research
- Creative Ability
- Value based instruction and

to emerge as a premiere institute.

INSTITUTE MISSION:

Achieving academic excellence by providing globally acceptable technical education by forecasting technology through

- Innovative Research and development
- Industry Institute Interaction
- Empowered Manpower

PRINCIPAL
PRINCIPAL
Aditya College of Engineering
SURAMPALEM - 533 437

ADITYA COLLEGE OF ENGINEERING

An AUTONOMOUS Institution

Approved by AICTE, Permanently Affiliated to JNTUK, Accredited by NBA & NAAC
Recognized by UGC under Sections 2(f) and 12(B) of UGC Act, 1956
Aditya Nagar, ADB Road, Surampalem - 533 437, E.G.Dist., Ph: 99631 76662.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Department Vision, Mission

DEPARTMENT VISION:

To be recognized as computer science & engineering hub striving to meet the growing needs of the Industry and society.

DEPARTMENT MISSION:

- Imparting Quality Education through state-of-the-art infrastructure with industry collaboration.
- Enable teaching and learning process with disseminate knowledge.
- Organize skill based, Industrial & Society events for overall Development.

HOD CSE
Head of the Department
Computer Science & Engineering
Aditya College of Engineering
SURAMPALEM-533 437

ADITYA COLLEGE OF ENGINEERING (A)

Approved by AICTE,Permenantly Affiliated to JNTUK, Accredited by NBA & NAAC

Recocginzed by UGC under Section 2(f) and 12(B) of UGC act, 1956

Aditya Nagar, ADB Road, Surampalem , Kakinada District, A.P-533 437

Department of Computer Science and Engineering

CERTIFICATE

This is to certify the project report entitled "**Port Aqua Market Demand Analysis and Recommendation System**" is a bonified work carried out by **D. Lakshmi Neha** bearing with reg No: **20MH1A0577**, **Y. Praveen Kumar** bearing with reg No: **20MH1A05C4**, **CH. Shyam Bhaskar** bearing with reg No: **20MH1A0575, V. Ram Ganesh** bearing with reg No: **20MH1A05C6** at the college for the award of Bachelor of Technology in COMPUTER SCIENCE AND ENGINEERING from ADITYA COLLEGE OF ENGINEERING (A) during the academic year of 2020-2024.

Project Guide
Mr. Dr. B.V.Rama Krishna, M.Tech., Ph.D.
Dept. Of CSE
Professor

Head of the Department
Dr.G.S.N.Murthy, M.Tech., Ph.D.
Dept. Of CSE
Professor

EXTERNAL EXAMINER

DECLARATION

We hereby declare that this project entitled **"Port Aqua Market Demand Analysis and Recommendation System"** has been undertaken by us and this work has been submitted to **ADITYA COLLEGE OF ENGINEERING (A)**, Surampalem affiliated to JNTUK, Kakinada, in partial fulfillment of the requirements for the award of the degree of **BACHELOR OF TECHNOLOGY** in **COMPUTER SCIENCE AND ENGINEERING**.

We further declare that this project work has not been submitted in full or part to any other University or educational institute for the award of any degree or diploma.

PROJECT ASSOCIATES

D. Lakshmi Neha	**(20MH1A0577)**
Y. Praveen Kumar	**(20MH1A05C4)**
CH. Shyam Bhaskar	**(20MH1A0575)**
V. Ram Ganesh	**(20MH1A05C6)**

ACKNOWLEDGEMENT

It is with immense pleasure that we would like to express our indebted gratitude to my **Project Supervisor**, **Mr. Dr. B. V. Rama Krishna**, **Professor** who has guided us a lot and encouraged us in every step of project work, his valuable moral support and guidance has been helpful in successful completion of this Project.

We wish to express our sincere thanks to **Dr. G S N Murthy**, **Head of the Department of CSE**, for his valuable guidance given to us throughout the period of the project work.

We feel elated to thank **Dr A. Ramesh**, **Principal** of Aditya College of Engineering for his cooperation in completion of our project and throughout our course.

We feel elated to thank **Dr. P.S.V.V.S. Ravi Kumar**, **Dean of Academics** of Aditya College of Engineering for his cooperation in completion of our project work.

We wish to express our sincere thanks to all faculty members, and lab programmers for their valuable guidance given to us throughout the period of the project.

We avail this opportunity to express our deep sense and heart full thanks to the **Management** of **Aditya College of Engineering (A)** for providing a great support for us by arranging the trainees, and facilities needed to complete our project and for giving us the opportunity for doing this work.

PROJECT ASSOCIATES

D. Lakshmi Neha	**(20MH1A0577)**
Y. Praveen Kumar	**(20MH1A05C4)**
CH. Shyam Bhaskar	**(20MH1A0575)**
V. Ram Ganesh	**(20MH1A05C6)**

ABSTRACT

The presented project introduces an innovative integrated system designed to revolutionize fish species classification and recommendation processes within the fisheries sector. Leveraging cutting-edge deep learning techniques, notably ResNet50 for image classification, the system aims to accurately identify various fish species from uploaded images. Once classified, the system offers users comprehensive information about each fish species, encompassing habitat details, nutritional value, and commercial significance. This wealth of information empowers users to make informed decisions regarding fish consumption while promoting sustainable fishing practices. Moreover, the integrated system incorporates sophisticated recommendation algorithms based on Singular Value Decomposition (SVD), ensuring tailored fish recommendations aligned with regional availability and consumer preferences. By leveraging SVD, the system provides personalized recommendations that enhance user satisfaction and promote responsible consumption habits. Furthermore, the system's intuitive user interface facilitates seamless interaction, allowing users to easily upload images, access classification results, and explore personalized recommendations. Through this user-centric approach, the system enhances consumer knowledge and decision-making capabilities, fostering a deeper understanding of fish species and their environmental impact. Beyond its immediate functionalities, the project underscores a broader commitment to environmental consciousness and ecosystem preservation. By merging technological innovation with sustainability principles, the integrated system contributes to market efficiency, consumer education, and stakeholder collaboration within the fisheries sector. Through ongoing collaboration with stakeholders and continuous improvement efforts, the project seeks to address evolving challenges and opportunities in fish species classification and recommendation, thereby driving positive outcomes for both industry stakeholders and marine ecosystems. This empowers users to make informed choices about fish consumption and promotes sustainable fishing practices. Additionally, the system employs sophisticated recommendation algorithms based on Singular Value Decomposition (SVD) to offer personalized fish recommendations aligned with regional availability and consumer preferences.

LIST OF FIGURES

S.No.	NAME OF FIGURE	PAGE No.
1.	Red Mullet	24
2.	Horse Mackerel	25
3.	Gill-Head Bream	26
4.	Black Sea Spart	27
5.	Dataset for Demand Analysis	28
6.	Dataset for Recommendation	28
7.	System Architecture	36
8.	Use case diagram	39
9.	Class diagram	41
10.	Sequence diagram	43
11.	Activity diagram	45
12.	Component diagram	47
13.	Collaboration diagram	48
14.	Deployment diagram	49
15.	Entity Relationship diagram	50
16.	Data Flow diagram	51
17.	Output Screens	93

LIST OF TABLES

S.No.	NAME OF TABLE	PAGE No.
1	List of use cases and actors associated	38
2	Use cases and descriptions	38 - 39

INDEX

CHAPTER	PAGE NO
ABSTRACT	VI
LIST OF FIGURES	VII
LIST OF TABLES	VIII
Chapter 1: INTRODUCTION	
1.1 Introduction to Port Aqua Market	1
1.2 Demand Analysis	3
1.3 Recommendation System	5
1.4 Introduction to Deep Learning	8
1.4.1 Types of Deep Learning	9
1.4.2 Advantages	10
1.4.3 Disadvantages	11
1.5 Motivation of the Work	12
Chapter 2: LITERATURE SURVEY	14
Chapter 3: PROBLEM STATEMENT AND METHODOLOGY	
3.1 Problem Definition	17
3.2 Existing System	18
3.2.1 Disadvantages	18
3.3 Proposed System	20
2.3.1 Advantages	20
3.4 Modules Division	21
3.4.1 Dataset Information	22
3.4.1.1 Attribute Information	23
3.4.1.2 Reading the CSV file	28
3.4.2 Data Pre-Processing	29

3.4.3 Training Data 30

3.4.4 Testing Data 32

3.4.5 User Interface 32

Chapter 4: SYSTEM STUDY

4.1 Feasibility Study 34

4.1.1 Technical Feasibility 34

4.1.2 Economic Feasibility 34

4.1.3 Operation Feasibility 35

4.1.4 Social Feasibility 35

Chapter 5: SYSTEM DESIGN

5.1 System Architecture 36

5.2 UML Diagrams 37

5.2.1 Usecase Diagram 39

5.2.2 Class Diagram 41

5.2.3 Sequence Diagram 43

5.2.4 Activity Diagram 45

5.2.5 Component Diagram 47

5.2.6 Collaboration Diagram 48

5.2.7 Deployment Diagram 49

5.2.8 Entity Relationship Diagram 49

5.2.9 DataFlow Diagram 50

Chapter 6: REQUIREMENT ANALYSIS

6.1 System Requirements 53

6.2 Functional Requirements 53

6.3 Non-Functional Requirements 54

6.3.1 User Interface and Human Factors 54

6.3.2 Software Requirements 56

6.3.3 Hardware Requirements 56
6.3.4 Usability 56
6.3.5 Reliability 57
6.3.6 Performance 57
6.3.7 Supportability 57
6.3.8 Physical Environment 57
6.3.9 Security Requirements 58
6.3.9.1 Access requirements 58
6.3.9.2 Integrity requirements 58
6.3.9.4 Private requirements 59
6.3.10 Resource Requirements 59
Chapter 7: IMPLEMENTATION
7.1 Software Used 60
7.1.1 Python 60
7.1.2 Introduction to Flask frame work 73
7.1.3 Python Libraries 75
7.2 Source Code 77
7.2.1 Deep Learning Model 77
7.2.2 Performance 85
Chapter 8: TESTING
8.1 Testing Levels 87
8.1.1 Unit Testing 87
8.1.2 Integration Testing 87
8.1.3 Functional Testing 87
8.1.4 System Testing 88
8.1.5 Py Test 88
8.2 Testing Methods 89
8.2.1 Static Testing 89
8.2.1.1 Inspect 89

8.2.1.2 Walkthrough 89
8.2.1.3 Technical Reviews 90
8.2.2 Dynamic Testing 90
8.2.2.1 Black Box Testing 90
8.2.2.2 White Box Testing 91
Chapter 9: OUTPUT SCREENS
9.1 Home Page 93
9.2 About Page 94
9.3 Register Page 94
9.4 Login Page 95
9.5 Upload Page 95
9.6 Classification Result 96
9.7 Demand Analysis Result 96
9.8 Recommendation Page 97
9.9 Recommendation Result 98
CONCLUSION 99
FUTURE ENCHANCEMENT 100
PUBLISHED WORK 101
BIBLIOGRAPHY 114

Chapter 1

INTRODUCTION

1.1 Introduction to Port Aqua Market

The project introduces an integrated system designed to revolutionize fish species classification, information dissemination, and recommendation within the fisheries sector. With the utilization of cutting-edge deep learning techniques, notably ResNet50 for image classification, the system accurately identifies fish species from uploaded images. This breakthrough in classification is coupled with a comprehensive information display mechanism, offering users detailed insights into each classified fish species, including habitat details, nutritional value, and commercial significance.

In an era defined by rapid technological evolution, this project represents a bold endeavor to redefine the trajectory of the fisheries sector. Through its visionary integration of advanced deep learning techniques and ecological principles, the system paves the way for a more sustainable and resilient future. By providing users with nuanced insights into fish species' ecological roles, nutritional attributes, and economic value, the project empowers individuals to make informed choices that align with environmental conservation and economic prosperity.

Moreover, the system incorporates sophisticated recommendation algorithms, such as Singular Value Decomposition (SVD), to provide tailored fish recommendations based on geographical data, regional availability, and consumer preferences. By offering pertinent suggestions aligned with local consumption patterns and sustainable fishing practices, the system empowers users to make informed decisions while promoting species diversity and environmental consciousness.

In the landscape of modern innovation, this project serves as a testament to the transformative potential of interdisciplinary collaboration and visionary thinking. By seamlessly integrating deep learning techniques with ecological principles, the system transcends traditional boundaries, forging new pathways towards a more sustainable and equitable future. Through its holistic approach to fish species classification and recommendation, the project embodies the spirit of innovation, resilience, and environmental stewardship. The project's scope extends beyond mere technological innovation; it aims to foster positive change within the fisheries sector by merging advanced technology with environmental awareness.

The integrated system presented in this project represents a significant leap forward in the realm of fish species classification and recommendation. Grounded in advanced deep learning methodologies, particularly ResNet50 for image classification, this system is poised to redefine how fish species are identified and understood within the fisheries sector. At its core, the project seeks to bridge the gap between technological innovation and environmental stewardship, offering a comprehensive solution that addresses key challenges facing the industry.

Fish species classification stands as a fundamental pillar of fisheries management and conservation efforts. By accurately identifying various fish species from uploaded images, the system lays the groundwork for informed decision-making regarding fishing practices and consumption habits. Through the meticulous application of deep learning techniques, such as ResNet50, the system achieves unparalleled levels of accuracy and reliability in species classification, setting a new standard for industry best practices.

Beyond classification, the system goes a step further by providing users with detailed information about each classified fish species. From habitat details to nutritional value and commercial significance, users gain access to a wealth of knowledge that empowers them to make conscientious choices regarding fish consumption. This holistic approach not only enhances consumer awareness but also fosters a deeper appreciation for the ecological complexities of marine ecosystems.

Central to the system's functionality is the integration of sophisticated recommendation algorithms, anchored by Singular Value Decomposition (SVD). By analyzing user preferences and regional availability, the system generates personalized fish recommendations that align with individual consumer needs and sustainability goals. This personalized approach represents a paradigm shift in how fish consumption is approached, emphasizing diversity, locality, and environmental responsibility.

At its essence, the project is driven by a commitment to sustainability and ecosystem preservation. By promoting species diversity and local consumption patterns, the system actively contributes to the preservation of marine ecosystems and the long-term viability of fisheries resources. Moreover, by raising consumer awareness and fostering environmental consciousness, the project catalyzes a cultural shift towards more responsible consumption habits.

The system's intuitive user interface serves as a gateway to its functionality, ensuring seamless interaction and accessibility for users of all backgrounds. Through a user-friendly design and intuitive navigation, the system democratizes access to information and empowers users to engage meaningfully with fish species classification and recommendation processes.

Collaboration lies at the heart of the project's success. By engaging with stakeholders across the fisheries sector, including fishermen, consumers, and regulatory bodies, the system ensures that its design and functionality align with the diverse needs and perspectives within the industry. This collaborative approach fosters a sense of ownership and investment among stakeholders, driving continuous improvement and innovation within the sector.

Looking ahead, the project holds immense potential for future enhancements and scalability. Advancements in image recognition technology, expanded data sources, and partnerships with research institutions and government agencies promise to further elevate the system's capabilities and impact. As the project continues to evolve, it remains steadfast in its commitment to driving positive change within the fisheries sector and beyond.

1.2 Demand Analysis

Demand analysis plays a pivotal role in understanding market dynamics and consumer preferences within the fisheries sector. At its core, demand analysis involves the systematic examination of factors influencing the quantity of fish demanded by consumers, ranging from economic variables to seasonal trends and cultural preferences. By delving into the intricacies of demand patterns, stakeholders can glean valuable insights that inform strategic decision-making and resource allocation.

One of the primary objectives of demand analysis is to uncover the underlying drivers of fish consumption and purchasing behavior. This entails scrutinizing a myriad of factors, including income levels, price elasticity, taste preferences, and demographic trends. By leveraging sophisticated analytical techniques, such as regression analysis and time-series modeling, researchers can discern the complex interplay between these variables and elucidate demand relationships.

Regression analysis stands as a cornerstone of demand analysis, offering a powerful framework for quantifying the impact of independent variables on fish consumption. Through regression

modeling, analysts can estimate demand functions that delineate how changes in factors such as income, prices, and advertising expenditures affect the quantity of fish demanded. Moreover, techniques like elasticity estimation enable stakeholders to assess the sensitivity of demand to price fluctuations, thereby informing pricing strategies and market positioning.

Time-series modeling represents another indispensable tool in demand analysis, particularly for capturing temporal variations and seasonal fluctuations in fish demand. By analyzing historical consumption data over time, analysts can identify recurring patterns and cyclical trends, facilitating the prediction of future demand levels with greater accuracy. Techniques such as seasonal decomposition and autoregressive integrated moving average (ARIMA) modeling enable researchers to discern underlying patterns amidst the noise of fluctuating demand dynamics.

Furthermore, machine learning algorithms have emerged as potent tools for demand analysis, offering the ability to uncover complex patterns and nonlinear relationships within vast datasets. Techniques such as decision trees, random forests, and neural networks excel at capturing intricate demand patterns that may elude traditional analytical methods. By harnessing the predictive power of machine learning, stakeholders can anticipate demand shifts, identify emerging trends, and tailor marketing strategies to meet evolving consumer preferences.

In addition to quantitative analysis, demand analysis often incorporates qualitative methodologies, such as consumer surveys and focus groups, to glean nuanced insights into consumer behavior and preferences. Qualitative research methods provide a rich understanding of consumer motivations, attitudes, and purchasing drivers, complementing the quantitative analysis with qualitative depth and context.

Moreover, demand analysis extends beyond individual consumer behavior to encompass broader market dynamics and industry trends. Market segmentation techniques enable stakeholders to identify distinct consumer segments with unique preferences and demand profiles, allowing for targeted marketing strategies and product offerings tailored to specific market niches.

Environmental factors also play a crucial role in shaping fish demand, with sustainability concerns and eco-labeling increasingly influencing consumer purchasing decisions. Demand analysis can shed light on the growing preference for sustainably sourced seafood and the impact

of eco-certification schemes on consumer behavior, guiding industry efforts to promote responsible fishing practices and environmental stewardship.

Furthermore, technological advancements, such as online platforms and e-commerce solutions, have revolutionized the way consumers access and purchase seafood products. Demand analysis in the digital age encompasses understanding the drivers of online purchasing behavior, optimizing digital marketing strategies, and leveraging data analytics to personalize the online shopping experience and enhance customer engagement.

In summary, demand analysis serves as a cornerstone of strategic decision-making within the fisheries sector, offering invaluable insights into consumer behavior, market dynamics, and industry trends. By combining quantitative and qualitative methodologies, leveraging advanced analytical techniques, and staying attuned to evolving consumer preferences and market forces, stakeholders can navigate the complexities of the seafood market landscape and drive sustainable growth and innovation within the industry.

1.3 Recommendation System

Recommendation systems represent a cornerstone of modern e-commerce and content platforms, empowering users with personalized suggestions tailored to their preferences and interests. At the heart of these systems lie sophisticated algorithms designed to analyze user behavior, preferences, and item attributes to deliver relevant and engaging recommendations. By leveraging vast datasets and advanced machine learning techniques, recommendation systems play a pivotal role in enhancing user satisfaction, driving engagement, and facilitating content discovery across a diverse array of domains, including e-commerce, streaming services, and social media platforms.

One of the key objectives of recommendation systems is to mitigate information overload by curating personalized recommendations that align with each user's tastes and preferences. Traditional approaches to recommendation systems encompass collaborative filtering, content-based filtering, and hybrid methods that combine elements of both. Collaborative filtering algorithms analyze user-item interactions and similarities between users or items to generate recommendations based on past behavior and preferences. Content-based filtering, on the other hand, leverages item attributes and user profiles to recommend items with similar characteristics

to those previously liked or interacted with by the user. Hybrid methods combine collaborative and content-based approaches to capitalize on the strengths of each method and provide more accurate and diverse recommendations.

Matrix factorization techniques, such as Singular Value Decomposition (SVD) and Alternating Least Squares (ALS), represent foundational algorithms in collaborative filtering-based recommendation systems. These techniques decompose the user-item interaction matrix into lower-dimensional matrices representing latent factors, such as user preferences and item attributes. By approximating the original matrix using these latent factors, matrix factorization algorithms can predict user-item interactions and generate personalized recommendations with high accuracy and scalability. SVD, in particular, has gained widespread adoption in recommendation systems due to its effectiveness in capturing latent factors and handling sparse and high-dimensional data.

Another prominent algorithm in recommendation systems is the k-nearest neighbors (kNN) algorithm, which belongs to the class of memory-based collaborative filtering methods. kNN algorithms identify similar users or items based on predefined similarity metrics, such as cosine similarity or Pearson correlation coefficient, and recommend items that similar users have interacted with or liked in the past. kNN algorithms offer simplicity and interpretability, making them popular choices for recommendation tasks, particularly in scenarios with small to medium-sized datasets.

In recent years, deep learning techniques have emerged as powerful tools for recommendation systems, enabling the modeling of complex patterns and nonlinear relationships in user-item interactions. Neural network architectures, such as multi-layer perceptrons (MLPs), recurrent neural networks (RNNs), and convolutional neural networks (CNNs), have demonstrated remarkable performance in recommendation tasks by learning hierarchical representations of user preferences and item attributes from raw input data. Deep learning-based recommendation systems excel at capturing intricate patterns in user behavior, handling sparse and noisy data, and generating personalized recommendations at scale.

Graph-based recommendation algorithms represent another innovative approach to recommendation systems, particularly in social networking and content recommendation platforms. Graph-based algorithms model user-item interactions as a graph structure, where users

and items are represented as nodes, and interactions as edges between nodes. Techniques such as graph neural networks (GNNs) and random walk-based algorithms leverage graph topology and node embeddings to propagate information and infer user preferences or item relevance within the graph structure. Graph-based recommendation systems excel at capturing complex relationships and user-item interactions in interconnected networks, facilitating accurate and context-aware recommendations.

In addition to algorithmic advancements, contextual information and user feedback play crucial roles in enhancing the performance and relevance of recommendation systems. Context-aware recommendation techniques leverage contextual factors such as time, location, device type, and user context to deliver recommendations tailored to the user's current situation and preferences. By incorporating contextual information into the recommendation process, these techniques can adapt recommendations in real-time to changing user needs and environmental factors, thereby enhancing user satisfaction and engagement.

Moreover, reinforcement learning (RL) techniques have garnered increasing interest in recommendation systems for their ability to optimize long-term user engagement and exploration-exploitation trade-offs. RL-based recommendation systems model the recommendation process as a sequential decision-making problem, where an agent learns to recommend items to maximize user engagement or utility over time. By interacting with users and receiving feedback on recommended items, RL agents can adapt their recommendation policies dynamically to learn and exploit user preferences effectively.

Furthermore, ensemble methods, such as stacking and boosting, have emerged as effective strategies for improving recommendation system performance by combining predictions from multiple base models. Ensemble techniques leverage the diversity of individual models and their complementary strengths to achieve superior recommendation accuracy and robustness. By aggregating predictions from diverse algorithms or model variants, ensemble methods mitigate the risk of overfitting, reduce prediction variance, and enhance the generalization ability of recommendation systems across diverse user preferences and item domains.

In summary, recommendation systems represent a cornerstone of personalized user experiences in digital platforms, offering tailored suggestions that cater to individual preferences and interests. By leveraging a diverse array of algorithms, including collaborative filtering, matrix

factorization, deep learning, graph-based techniques, and contextual recommendation methods, recommendation systems can deliver accurate, diverse, and engaging recommendations across various domains. As data volumes grow and user expectations evolve, continued innovation in recommendation algorithms and techniques is essential to meet the demands of an increasingly personalized and dynamic digital landscape.

1.4 Introduction to Deep Learning

Deep learning is a branch of machine learning which is based on artificial neural networks. It is capable of learning complex patterns and relationships within data. In deep learning, we don't need to explicitly program everything. It has become increasingly popular in recent years due to the advances in processing power and the availability of large datasets. Because it is based on artificial neural networks (ANNs) also known as deep neural networks (DNNs). These neural networks are inspired by the structure and function of the human brain's biological neurons, and they are designed to learn from large amounts of data. Deep Learning is a subfield of Machine Learning that involves the use of neural networks to model and solve complex problems. Neural networks are modeled after the structure and function of the human brain and consist of layers of interconnected nodes that process and transform data. The key characteristic of Deep Learning is the use of deep neural networks, which have multiple layers of interconnected nodes. These networks can learn complex representations of data by discovering hierarchical patterns and features in the data. Deep Learning algorithms can automatically learn and improve from data without the need for manual feature engineering. Deep Learning has achieved significant success in various fields, including image recognition, natural language processing, speech recognition, and recommendation systems. Some of the popular Deep Learning architectures include Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Deep Belief Networks (DBNs). Training deep neural networks typically requires a large amount of data and computational resources. However, the availability of cloud computing and the development of specialized hardware, such as Graphics Processing Units (GPUs), has made it easier to train deep neural networks. In summary, Deep Learning is a subfield of Machine Learning that involves the use of deep neural networks to model and solve complex problems. Deep Learning has achieved significant success in various fields, and its use is expected to continue to grow as more data becomes available, and more powerful computing

resources become available.

1.4.1 Types of Deep Learning

Deep learning encompasses various architectures and techniques tailored for different tasks and data types. Here's a broad overview of some common types:

Convolutional Neural Networks (CNNs): Widely used for image recognition and classification tasks due to their ability to capture spatial hierarchies of features through convolutional layers.

Recurrent Neural Networks (RNNs): Suited for sequential data processing tasks such as natural language processing, time series analysis, and speech recognition, thanks to their recurrent connections that maintain memory over time.

Long Short-Term Memory Networks (LSTMs): A type of RNN designed to address the vanishing gradient problem, allowing for better modeling of long-range dependencies in sequential data.

Generative Adversarial Networks (GANs): Comprising a generator and a discriminator, GANs are adept at generating realistic data samples and have applications in image generation, style transfer, and data augmentation.

Autoencoders: Used for unsupervised learning tasks such as data denoising, dimensionality reduction, and feature learning, by learning to encode input data into a compact latent representation and then reconstructing it.

Deep Belief Networks (DBNs): Consisting of multiple layers of stochastic latent variables, DBNs are employed for feature learning, collaborative filtering, and classification tasks, trained layer by layer using unsupervised learning techniques like restricted Boltzmann machines (RBMs).

Transformers: Particularly prominent in natural language processing tasks, transformers leverage self-attention mechanisms to model relationships between different elements in a sequence, achieving state-of-the-art results in tasks like language translation, text summarization, and sentiment analysis.

Capsule Networks: Introduced as an alternative to CNNs, capsule networks aim to capture

hierarchical structures in images more effectively by representing entities as capsules and preserving spatial relationships between them.

1.4.2 Advantages of Deep Learning

Deep learning has several advantages over traditional machine learning methods, some of the main ones include:

1. **Automatic feature learning:** Deep learning algorithms can automatically learn features from the data, which means that they don't require the features to be hand-engineered. This is particularly useful for tasks where the features are difficult to define, such as image recognition.
2. **Handling large and complex data:** Deep learning algorithms can handle large and complex datasets that would be difficult for traditional machine learning algorithms to process. This makes it a useful tool for extracting insights from big data.
3. **Improved performance:** Deep learning algorithms have been shown to achieve state-of-the-art performance on a wide range of problems, including image and speech recognition, natural language processing, and computer vision.
4. **Handling non-linear relationships:** Deep learning can uncover non-linear relationships in data that would be difficult to detect through traditional methods.
5. **Handling structured and unstructured data:** Deep learning algorithms can handle both structured and unstructured data such as images, text, and audio.
6. **Predictive modeling:** Deep learning can be used to make predictions about future events or trends, which can help organizations plan for the future and make strategic decisions.
7. **Handling missing data:** Deep learning algorithms can handle missing data and still make predictions, which is useful in real-world applications where data is often incomplete.
8. **Handling sequential data:** Deep learning algorithms such as Recurrent Neural Networks (RNNs) and Long Short-term Memory (LSTM) networks are particularly suited to handle sequential data such as time series, speech, and text. These algorithms have the ability to maintain context and memory over time, which allows them to make predictions or

decisions based on past inputs.

9. **Scalability:** Deep learning models can be easily scaled to handle an increasing amount of data and can be deployed on cloud platforms and edge devices.

10. **Generalization:** Deep learning models can generalize well to new situations or contexts, as they are able to learn abstract and hierarchical representations of the data. Deep learning has several advantages over traditional machine learning methods, including automatic feature learning, handling large and complex data, improved performance, handling non-linear relationships, handling structured and unstructured data, predictive modeling, handling missing data, handling sequential data, scalability and generalization ability.

1.4.3 Disadvantages of Deep Learning

While deep learning has many advantages, there are also some disadvantages to consider:

1. **High computational cost:** Training deep learning models requires significant computational resources, including powerful GPUs and large amounts of memory. This can be costly and time-consuming.

2. **Overfitting:** Overfitting occurs when a model is trained too well on the training data and performs poorly on new, unseen data. This is a common problem in deep learning, especially with large neural networks, and can be caused by a lack of data, a complex model, or a lack of regularization.

3. **Lack of interpretability:** Deep learning models, especially those with many layers, can be complex and difficult to interpret. This can make it difficult to understand how the model is making predictions and to identify any errors or biases in the model.

4. **Dependence on data quality:** Deep learning algorithms rely on the quality of the data they are trained on. If the data is noisy, incomplete, or biased, the model's performance will be negatively affected.

5. **Data privacy and security concerns:** As deep learning models often rely on large amounts of data, there are concerns about data privacy and security. Misuse of data by malicious actors can lead to serious consequences like identity theft, financial loss and invasion of

privacy.

6. **Lack of domain expertise:** Deep learning requires a good understanding of the domain and the problem you are trying to solve. If the domain expertise is lacking, it can be difficult to formulate the problem and select the appropriate algorithm.
7. **Unforeseen consequences:** Deep learning models can lead to unintended consequences, for example, a biased model can discriminate against certain groups of people, leading to ethical concerns.
8. **Limited to the data its trained on:** Deep learning models can only make predictions based on the data it has been trained on. They may not be able to generalize to new situations or contexts that were not represented in the training data.
9. **Black box models:** some deep learning models are considered as "black-box" models, as it is difficult to understand how the model is making predictions and identifying the factors that influence the predictions.

1.5 Motivation of the Work

The motivation behind this project is deeply rooted in the recognition of the critical challenges facing the fisheries sector, which extend far beyond the realms of mere economic considerations. At its core, the project seeks to address the multifaceted issues plaguing the industry, ranging from the inaccurate identification of fish species to the dissemination of inadequate information and the pervasive presence of unsustainable fishing practices. These challenges not only pose immediate threats to the delicate balance of marine ecosystems but also jeopardize the livelihoods of millions of people worldwide who depend on fisheries for sustenance and economic stability.

In response to these challenges, the project draws inspiration from the transformative potential of cutting-edge technologies, particularly advanced deep learning techniques and recommendation algorithms. By harnessing the power of these technologies, the project endeavors to create an integrated system capable of revolutionizing how fish species are identified, information is disseminated, and fishing practices are conducted. Through this holistic approach, the project aims to lay the groundwork for a more sustainable and equitable fisheries sector—one that is

firmly rooted in environmental stewardship, social responsibility, and economic resilience.

Central to the motivation behind this project is the firm belief in the pivotal role that technology can play in driving positive change within the fisheries sector. By leveraging advanced deep learning techniques, the project seeks to enhance the accuracy and efficiency of fish species identification, thereby mitigating the risks associated with misidentification and enabling more precise management strategies. Furthermore, by integrating recommendation algorithms, the project aims to empower stakeholders with tailored insights and recommendations that reflect regional nuances, consumer preferences, and ecological considerations.

Beyond its technical aspirations, the project is underpinned by a profound sense of responsibility towards environmental conservation and sustainable development. By promoting the preservation of ecosystems and the adoption of sustainable fishing practices, the project seeks to safeguard the long-term viability of fisheries resources for future generations. Moreover, by fostering greater market efficiency and empowering consumers with knowledge, the project endeavors to create a more transparent and resilient fisheries sector—one that is capable of adapting to evolving challenges and opportunities in an increasingly interconnected world.

Ultimately, the motivation behind this project transcends individual interests and organizational boundaries, encompassing a collective commitment to the greater good of society and the planet as a whole. By striving to address the pressing challenges facing the fisheries sector, the project aspires to catalyze positive change on a global scale, fostering environmental consciousness, promoting social equity, and unlocking new opportunities for sustainable growth and prosperity. In doing so, the project endeavors to leave a lasting legacy of innovation, resilience, and stewardship—a legacy that will endure for generations to come.

Chapter 2

LITERATURE SURVEY

Related work:

[1] S. Ogunlana, O. Olabode, S. Oluwadare, and G. Iwasokun, "Fish Classification Using Support Vector Machine," African Journal of Computing & ICT, vol. 8, pp. 75-82, 2015.

[2] S. Bermejo, "Fish age classification based on length, weight, sex and otolith morphological features," Fisheries Research, vol. 84, pp. 270-274, 2007.

[3] A. G. Cabreira, M. Tripode, and A. Madirolas, "Artificial neural networks for fish- species identification," ICES Journal of Marine Science, vol. 66, pp. 1119-1129, 2009.

[4] G. Ding, Y. Song, J. Guo, C. Feng, G. Li, B. He, and T. Yan, "Fish recognition using convolutional neural network," in OCEANS – Anchorage, 2017, Anchorage, AK, USA, USA, 2017, pp. 1-4.

[5] T. T. Hnin and K. T. Lynn, "Fish Classification Based on Robust Features Selection Using Machine Learning Techniques," in Genetic and Evolutionary Computing: Proceedings of the Ninth International Conference on Genetic and EvolutionaryComputing, August 26-28, 2015, Yangon, Myanmar - Volume 1, T. T. Zin, J. C.-W. Lin, J.-S. Pan, P. Tin, and M. Yokota, Eds., ed Cham: Springer International Publishing, 2016, pp. 237-245.

[6] B. S. Al Smadi, "Applications of Meta-Heuristic Algorithm with Back PropagationClassifier for Handling Class of General Fish Models," International Journal of Computer Science and Network Security (IJCSNS), vol. 16, p. 38, 2016.

[7] M. Alsmadi, K. Omar, and I. Almarashdeh, Fish Classification: Fish Classification Using Memetic Algorithms with Back Propagation Classifier: LAP LAMBERT Academic Publishing, 2012.

[8] M. Alsmadi, K. Omar, S. Noah, and I. Almarashdeh, "A hybrid memetic algorithm with back-propagation classifier for fish classification based on robust features extraction from PLGF and shape measurements," Information Technology Journal, vol. 10, pp. 944-954, 2011.

[9] M. K. Alsmadi, M. Tayfour, R. A. Alkhasawneh, U. Badawi, I. Almarashdeh, and F. Haddad, "Robust feature extraction methods for general fish classification," International Journal of Electrical & Computer Engineering (2088-8708), vol. 9, pp. 5192-5204, 2019.

[10] N. Castignolles, M. Cattoen, and M. Larinier, "Identification and counting of live fish by image analysis," in Image and Video Processing II, 1994, pp. 200-209.

[11] B. Zion, A. Shklyar, and I. Karplus, "Sorting fish by computer vision," Computers and Electronics in Agriculture, vol. 23, pp. 175-187, 1999.

[12] B. Zion, A. Shklyar, and I. Karplus, "In-vivo fish sorting by computer vision," Aquacultural Engineering, vol. 22, pp. 165-179, 2000.

[13] D. Lee, S. Redd, R. Schoenberger, X. Xu, and P. Zhan, "An automated fish species classification and migration monitoring system," in Industrial Electronics Society, 2003. IECON'03. The 29th Annual Conference of the IEEE, 2003, pp. 1080-1085.

[14] D.-J. Lee, J. K. Archibald, R. B. Schoenberger, A. W. Dennis, and D. K. Shiozawa, "Contour matching for fish species recognition and migration monitoring," in Applications of Computational Intelligence in Biology, ed: Springer, 2008, pp. 183-207.

[15] M. Nery, A. Machado, M. F. M. Campos, F. L. Pádua, R. Carceroni, and J. P. Queiroz-Neto, "Determining the appropriate feature set for fish classification tasks," in Computer Graphics and Image Processing, 2005. SIBGRAPI 2005. 18th Brazilian Symposium on, 2005, pp. 173-180.

[16] U. A. Badawi and M. K. Alsmadi, "A General Fish Classification Methodology Using Meta-Heuristic Algorithm With Back Propagation Classifier," Journal of Theoretical & Applied Information Technology, vol. 66, pp. 803-812, 2014.

[17] M. K. Alsmadi, "Hybrid Genetic Algorithm with Tabu Search with Back-Propagation Algorithm for Fish Classification: Determining the Appropriate Feature Set," International Journal of Applied Engineering Research, vol. 14, pp. 4387-4396, 2019.

[18] I. Sharmin, N. F. Islam, I. Jahan, T. A. Joye, M. R. Rahman, and M. T. Habib, "Machine vision based local fish recognition," SN Applied Sciences, vol. 1, p. 1529, 2019.

[19] H. S. Chhabra, A. K. Srivastava, and R. Nijhawan, "A Hybrid Deep Learning Approach for

Automatic Fish Classification," in Proceedings of ICETIT 2019, ed: Springer, 2020, pp. 427-436.

[20] E. Checcucci, R. Autorino, G. E. Cacciamani, D. Amparore, S. De Cillis, A. Piana, P. Piazzolla, E. Vezzetti, C. Fiori, and D. Veneziano, "Artificial intelligence and neural networks in urology: current clinical applications," Minerva Urologica e Nefrologica= The Italian Journal of Urology and Nephrology, vol. 72, pp. 49-57, 2019.

[21] P. Thorat, R. Tongaonkar, and V. Jagtap, "Towards Designing the Best Model for Classification of Fish Species Using," in Proceeding of International Conference on Computational Science and Applications: ICCSA 2019, 2020, p. 343.

[22] S. Cui, Y. Zhou, Y. Wang, and L. Zhai, "Fish Detection Using Deep Learning," Applied Computational Intelligence and Soft Computing, vol. 2020, p. 3738108, 2020/01/23 2020.

[23] A. Taheri-Garavand, A. Nasiri, A. Banan, and Y.-D. Zhang, "Smart deep learning-based approach for non-destructive freshness diagnosis of common carp fish," Journal of Food Engineering, vol. 278, p. 109930, 2020.

[24] Z. Zheng, C. Guo, X. Zheng, Z. Yu, W. Wang, H. Zheng, M. Fu, and B. Zheng, "Fish recognition from a vessel camera using deep convolutional neural network and data augmentation," in 2018 OCEANS-MTS/IEEE Kobe Techno-Oceans (OTO), 2018, pp. 1-5.

[25] T. Miyazono and T. Saitoh, "Fish Species Recognition Based on CNN Using Annotated Image," in IT Convergence and Security 2017, ed: Springer, 2018, pp. 156-163.

[26] M. Sung, S.-C. Yu, and Y. Girdhar, "Vision based real-time fish detection using convolutional neural network," in OCEANS 2017-Aberdeen, 2017, pp. 1-6.

[27] M. Alsmadi, K. B. Omar, S. A. Noah, and I. Almarashdeh, "Fish Recognition Based on Robust Features Extraction from Size and Shape Measurements Using Neural Network " Journal of Computer Science, vol. 6, pp. 1088-1094, 2010.

[28] M. K. Alsmadi, K. B. Omar, and S. A. Noah, "Fish classification based on robust features extraction from color signature using back-propagation classifier," Journal of Computer Science, vol. 7, p. 52, 2011.

Chapter 3

PROBLEM STATEMENT AND METHODOLOGY

3.1 Problem Definition

The problem statement addressed by this project encapsulates several critical challenges plaguing the fisheries sector, each of which has far-reaching implications for environmental sustainability, economic viability, and social welfare. At its core, the project seeks to confront the pressing need for an efficient and accurate system capable of addressing these challenges head-on, thereby laying the groundwork for a more resilient and equitable fisheries sector.

One of the primary challenges tackled by the project is the inadequacy of existing methods for fish species classification. Traditional approaches to identifying fish species may suffer from inherent limitations, such as low accuracy rates and the inability to handle diverse species datasets effectively. As a result, misidentification of fish species can occur, leading to erroneous management decisions, compromised conservation efforts, and diminished consumer confidence in seafood products. By addressing these limitations through the integration of advanced deep learning techniques, the project aims to enhance the accuracy and reliability of fish species classification, thereby mitigating the risks associated with misidentification and improving the overall integrity of fisheries management practices.

Furthermore, the project seeks to bridge the gap in information dissemination within the fisheries sector by providing comprehensive details about each classified fish species. Existing systems may offer limited or outdated information, hindering consumers' ability to make informed decisions about their seafood choices. This lack of transparency not only undermines consumer trust but also perpetuates misinformation and misconceptions about sustainable fishing practices. To address this challenge, the project endeavors to develop an integrated system that not only accurately identifies fish species but also delivers detailed information about their habitat, nutritional value, and commercial significance. By empowering consumers with knowledge, the project aims to promote transparency, accountability, and environmental stewardship within the fisheries sector.

Moreover, the project recognizes the importance of personalized recommendations in driving

sustainable fishing practices and consumer education. Existing recommendation systems may overlook regional variations in fish availability and consumer preferences, leading to inefficient resource utilization and suboptimal decision-making. By leveraging sophisticated recommendation algorithms, the project aims to provide tailored recommendations aligned with regional availability and consumer preferences, thereby promoting the responsible consumption of seafood and reducing the ecological footprint of fishing activities. Through this personalized approach, the project seeks to empower consumers to make informed choices that align with their values and contribute to the long-term sustainability of fisheries resources.

In essence, the problem statement addressed by this project underscores the need for a multifaceted solution that integrates advanced technology, environmental consciousness, and stakeholder collaboration. By developing an efficient and accurate system for fish species classification, information dissemination, and recommendation, the project aims to address key challenges within the fisheries sector and pave the way for a more sustainable and resilient future. Through its holistic approach, the project aspires to catalyze positive change across the entire seafood supply chain, from ocean to plate, ultimately benefiting ecosystems, communities, and economies worldwide.

3.2 Existing System

Existing methods for fish species classification typically rely on manual identification by experts or basic image processing techniques, lacking accuracy and efficiency. Similarly, fish recommendation systems often offer generic suggestions without considering regional availability or sustainability. These limitations highlight the need for a more sophisticated approach, which the proposed system aims to address through advanced deep learning techniques and tailored recommendations.

3.2.1 Disadvantages

While the proposed integrated system for fish species classification and recommendation offers numerous advantages, there are also some potential disadvantages to consider:

1. Complexity: Implementing advanced deep learning techniques and recommendation algorithms may introduce complexity to the system's development, deployment, and

maintenance. This complexity could potentially lead to increased costs and technical challenges.

2. Data Dependency: The effectiveness of the system relies heavily on the availability and quality of data, including labeled fish images for training the classification model and relevant regional data for generating recommendations. Limited or biased data could result in inaccurate classifications and recommendations.

3. Computational Resources: Deep learning models, particularly those as sophisticated as ResNet50, require significant computational resources for training and inference. This could pose challenges for users with limited computing power or access to cloud resources.

4. Bias and Fairness: The recommendation algorithms used in the system may inadvertently perpetuate biases present in the training data, leading to unfair or discriminatory recommendations. Ensuring fairness and mitigating bias in the recommendations requires careful attention to data selection and algorithm design.

5. User Acceptance: Users may be resistant to adopting new technology, especially if they are accustomed to traditional methods of fish species identification and recommendation. Effective user education and training may be necessary to facilitate widespread adoption of the system.

6. Privacy Concerns: Collecting and processing user data for personalized recommendations raises privacy concerns. Implementing robust privacy measures and obtaining user consent for data usage are essential to address these concerns and maintain user trust.

7. Environmental Impact: While the system aims to promote sustainability in fishing practices, there is a risk that increased access to information and recommendations could lead to higher demand for certain fish species, potentially exacerbating overfishing and environmental degradation.

Overall, while the proposed system offers significant benefits, addressing these potential disadvantages requires careful consideration and mitigation strategies to ensure the system's effectiveness, fairness, and ethical use.

3.3 Proposed System

The proposed system aims to revolutionize fish species classification, information dissemination, and recommendation by introducing an integrated solution that addresses key challenges in the fisheries sector. Leveraging advanced deep learning techniques, such as ResNet50, the system ensures accurate and efficient classification of fish species from uploaded images, providing users with detailed information about each species, including habitat details, nutritional value, and commercial significance. Additionally, the system incorporates recommendation algorithms like Singular Value Decomposition (SVD) to offer personalized fish recommendations based on geographical data, regional availability, and consumer preferences. With an intuitive user interface and educational components, the system enhances user experience and promotes environmental consciousness, fostering sustainable fishing practices and responsible consumption habits. Through stakeholder engagement and collaboration, the proposed system seeks to meet the diverse needs of fishermen, consumers, and regulatory bodies, ultimately contributing to ecosystem preservation, market efficiency, and consumer education within the fisheries sector.

3.3.1 Advantages:

The proposed integrated system for fish species classification and recommendation offers several advantages:

1. Accuracy: Utilizing advanced deep learning techniques ensures accurate classification of fish species from uploaded images, reducing errors associated with manual identification.

2. Efficiency: Automated classification and recommendation processes streamline workflows, saving time and resources for both users and stakeholders in the fisheries sector.

3. Comprehensive Information: Users gain access to detailed information about each classified fish species, including habitat details, nutritional value, and commercial significance, aiding informed decision-making.

4. Personalized Recommendations: Recommendation algorithms generate tailored fish recommendations based on geographical data, regional availability, and consumer preferences, enhancing user satisfaction and promoting sustainable consumption patterns.

5. User-Friendly Interface: An intuitive user interface makes the system accessible and easy to use for a wide range of users, fostering widespread adoption and engagement.

6. Environmental Consciousness: Educational components promote environmental consciousness among users, encouraging sustainable fishing practices and responsible consumption habits.

7. Stakeholder Collaboration: Collaboration with stakeholders across the fisheries sector ensures that the system meets the diverse needs of fishermen, consumers, and regulatory bodies, fostering industry-wide support and engagement.

8. Market Efficiency: By promoting species diversity and local consumption patterns, the system contributes to market efficiency within the fisheries sector, benefiting both producers and consumers.

Overall, the proposed system offers numerous advantages, including improved accuracy, efficiency, user satisfaction, and environmental sustainability, making it a valuable tool for stakeholders in the fisheries sector.

3.4 Modules Division

System

1. **Preprocessing:** Preprocessing of the image data is a critical step aimed at ensuring the data is in an optimal state for subsequent machine learning tasks. This process involves a series of tasks such as handling potential image artifacts, addressing missing or corrupted images, encoding categorical labels if applicable, and normalizing pixel values. By meticulously preparing the image data, the overarching goal is to enhance its quality and consistency, thereby facilitating effective utilization in the subsequent machine learning model.

2. **Data Splitting:** Following preprocessing, the data is typically divided into separate sets for training and testing purposes. The training set is utilized to train the machine learning model, while the testing set is employed to evaluate its performance. Although data splitting can be done randomly, it is sometimes essential to maintain the distribution of classes, particularly in classification problems, to ensure a balanced representation of different categories.

3. Model Training: Once the data is split, the machine learning model can be trained using the training set. This process involves feeding the training data into the model, allowing it to learn patterns and relationships inherent in the data. The choice of the model depends on various factors such as the nature of the problem (classification, regression, etc.) and the characteristics of the data. During training, hyperparameters may be tuned to optimize the model's performance and enhance its ability to generalize to unseen data.

4. Generating Results: Upon completion of the training phase, the trained model can be utilized to generate predictions on new, unseen data. This is achieved by calling the predict method, which applies the learned patterns and relationships to make predictions on the input data. The generated results provide valuable insights and enable informed decision-making in various applications.

User:

1. Data Loading: The user module involves the initial step of bringing raw data into the program. This process typically includes reading data from various file extensions such as JPG, PNG, etc. The raw data may contain images or other types of data relevant to the machine learning task at hand.

2. Choosing Algorithms: Algorithm selection plays a crucial role in determining the effectiveness and performance of the machine learning model. For classification tasks, common algorithms include logistic regression, decision trees, random forests, support vector machines, and neural networks. On the other hand, for regression tasks, popular algorithms include linear regression, decision trees, random forests, and gradient boosting algorithms. Experimentation with multiple algorithms is recommended, and cross-validation techniques can aid in selecting the most suitable model for the specific problem and dataset.

3.4.1 Dataset Information

The dataset used in the above project for fish species classification, information dissemination, and recommendation comprises a comprehensive collection of images of various fish species, along with associated metadata such as habitat details, nutritional value, and

commercial significance. This dataset serves as the foundation for training and testing the deep learning models used for fish species classification and recommendation.

Additionally, the project may utilize supplementary datasets containing information about regional availability, consumer preferences, market trends, and environmental factors relevant to the fisheries sector. These datasets augment the core fish species dataset, enabling the recommendation engine to generate personalized recommendations based on a wide range of criteria, including user location, dietary preferences, and sustainability considerations.

Furthermore, the project may incorporate real-time data streams or external APIs to enrich the dataset with up-to-date information about fish species availability, market prices, regulatory changes, and environmental conditions. This dynamic data integration ensures that the integrated system remains responsive to changing conditions and user needs, enhancing its relevance and effectiveness in promoting sustainable fishing practices and informed decision-making.

3.4.1.1 Datasets for Classification

The dataset is carefully curated to ensure diversity in terms of fish species, image quality, lighting conditions, and environmental backgrounds, reflecting real-world scenarios encountered by users of the integrated system. Additionally, the dataset may include metadata or annotations providing supplementary information about each fish species, such as habitat details, nutritional value, and commercial significance, which can enrich the user experience and support informed decision-making.

The dataset is carefully curated to ensure diversity in terms of fish species, image quality, lighting conditions, and environmental backgrounds, reflecting real-world scenarios encountered by users of the integrated system. Additionally, the dataset may include metadata or annotations providing supplementary information about each fish species, such as habitat details, nutritional value, and commercial significance, which can enrich the user experience and support informed decision-making.

Furthermore, the dataset may be augmented with additional images or samples to enhance the model's robustness and generalization capabilities. Augmentation techniques such as rotation, flipping, scaling, and noise addition may be applied to increase the variability of the training data

and prevent overfitting.

Overall, the dataset serves as the foundation for training and evaluating the deep learning model, enabling it to accurately classify fish species from uploaded images and provide users with valuable information about each species, contributing to the effectiveness and utility of the integrated system in the fisheries sector.

1.Red Mullet:

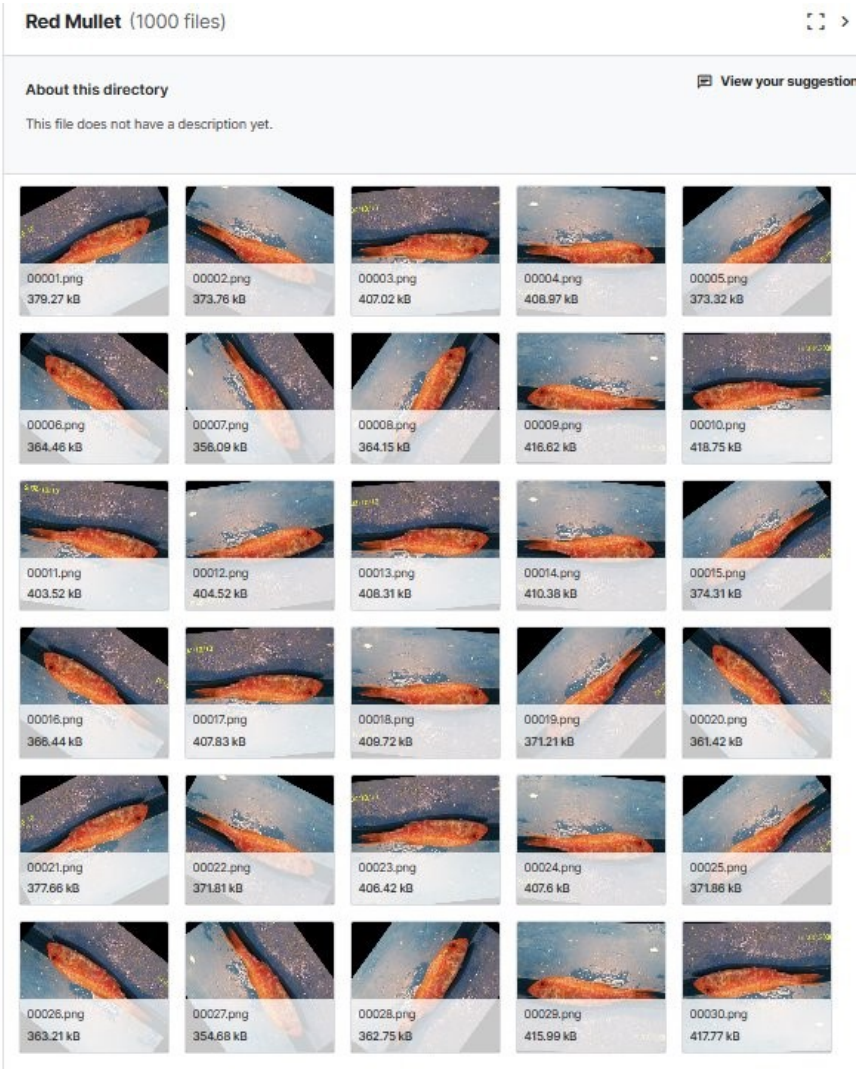


Fig.3.4.1.1.(a) Red Mullet

2.Hourse Mackerel:

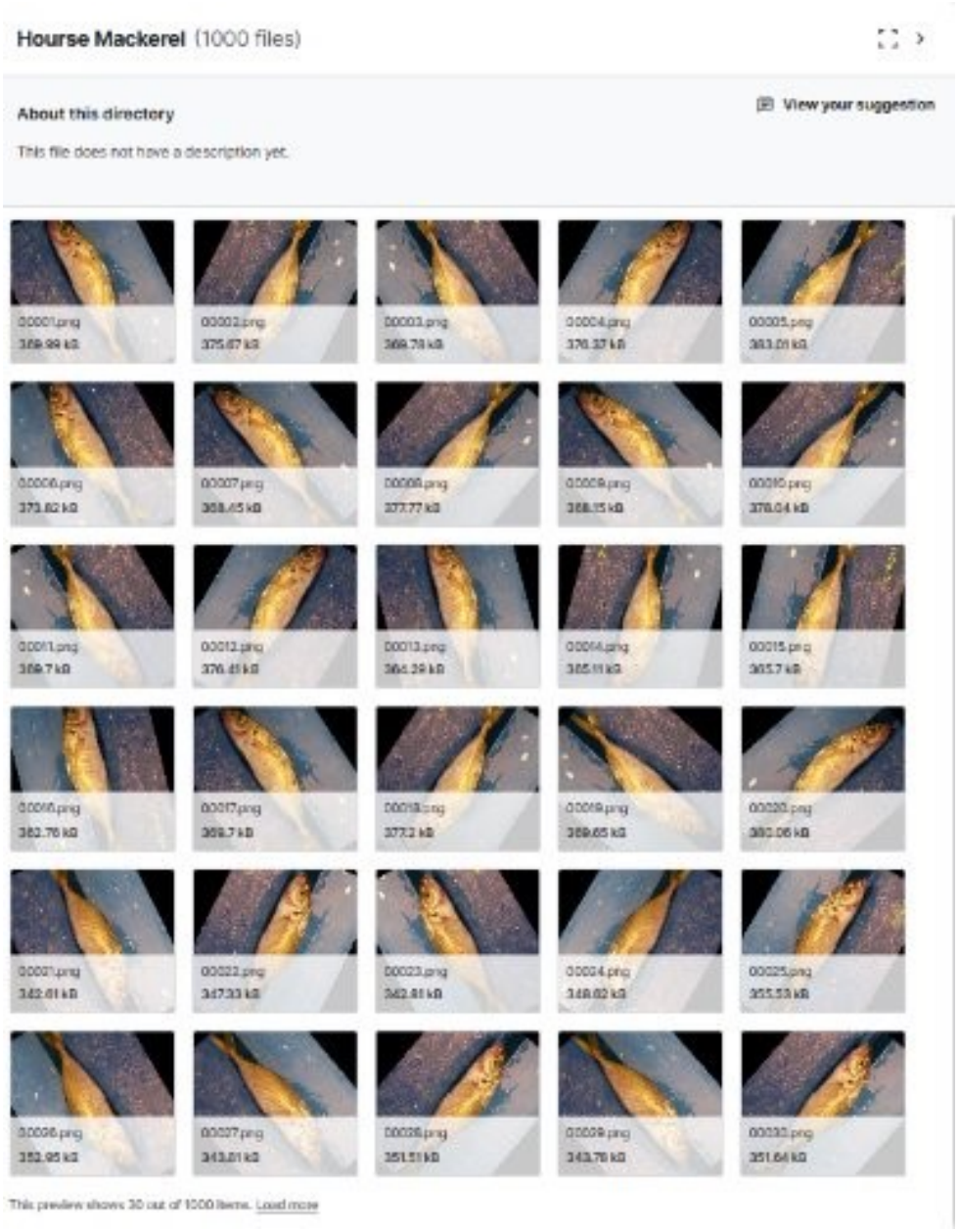


Fig.3.4.1.1.(b) Horse Mackerel

3. Gilt-Head Bream:

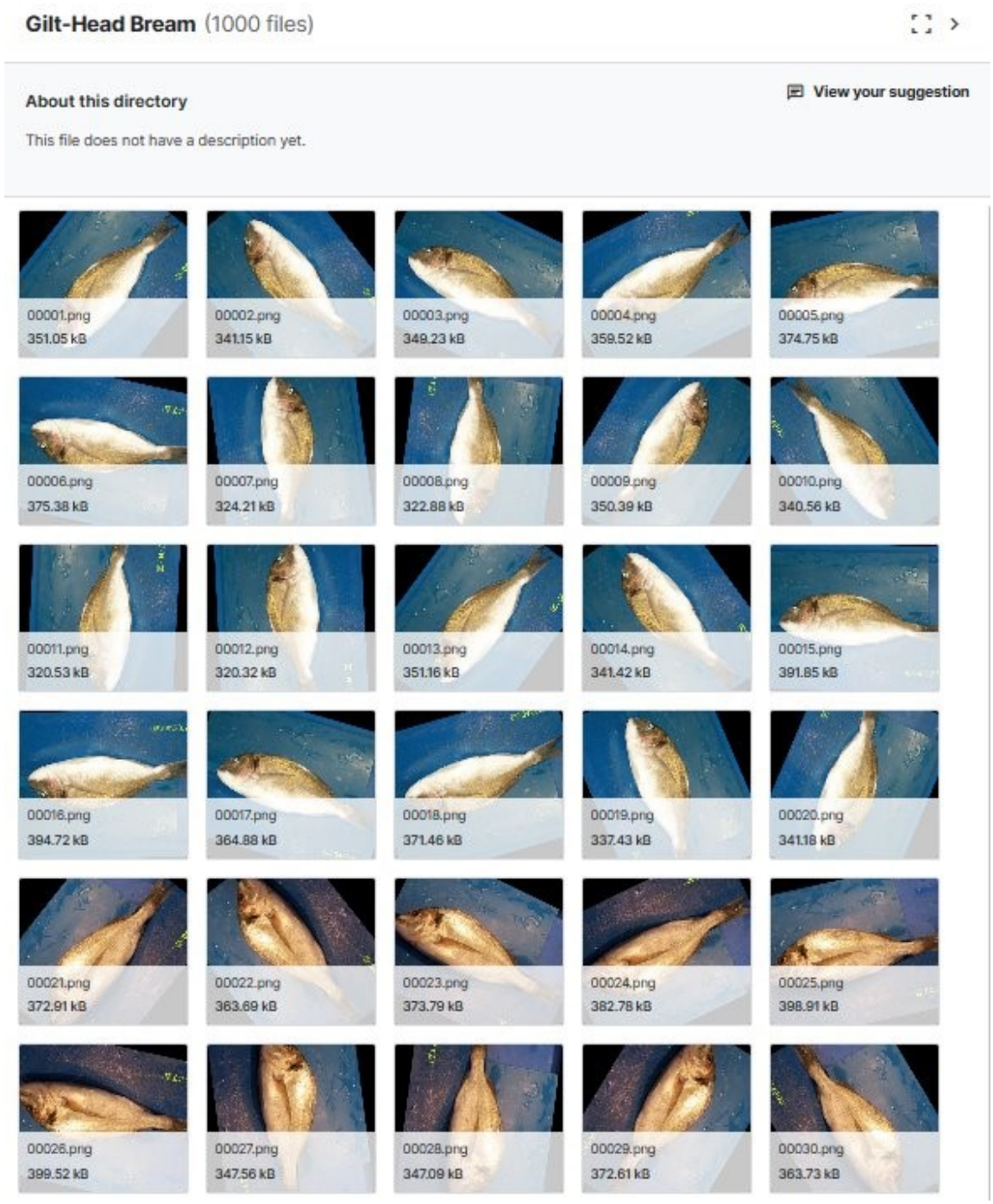


Fig.3.4.1.1.(c) Gilt-Head Bream

4.Black Sea Spart:

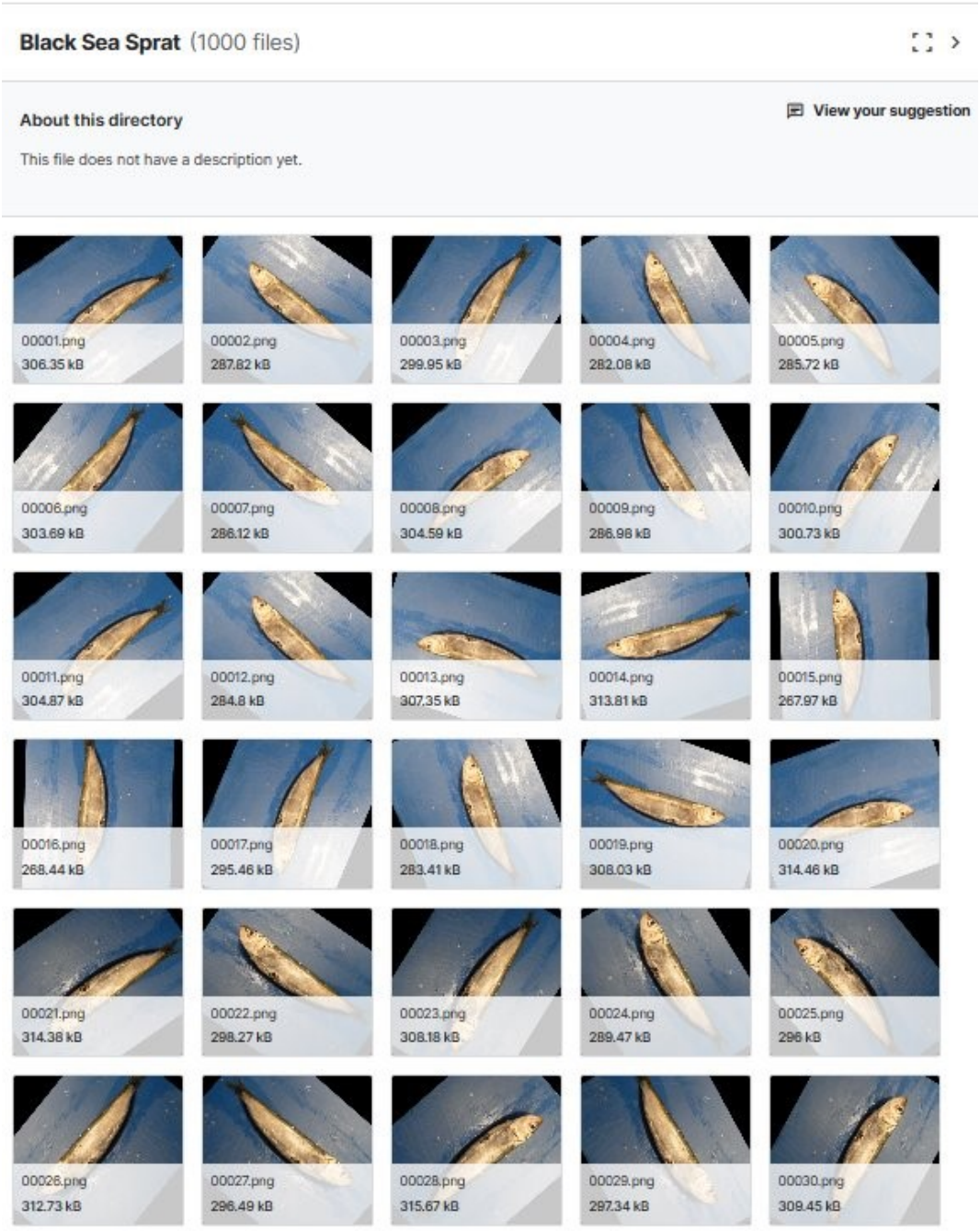


Fig.3.4.1.1.(d) Black Sea Sprat

3.4.1.2 Datasets for Demand Analysis

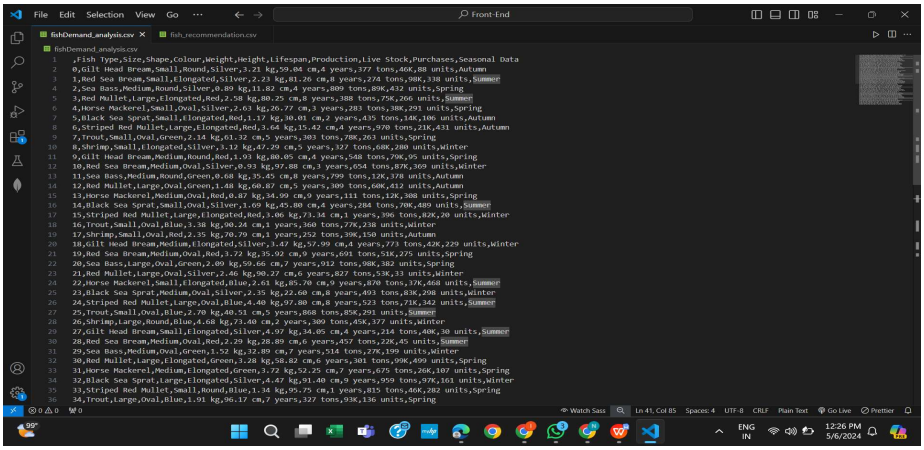


Fig.3.4.1.2 Datasets for Demand Analysis

3.4.1.3 Datasets for Recommendation System

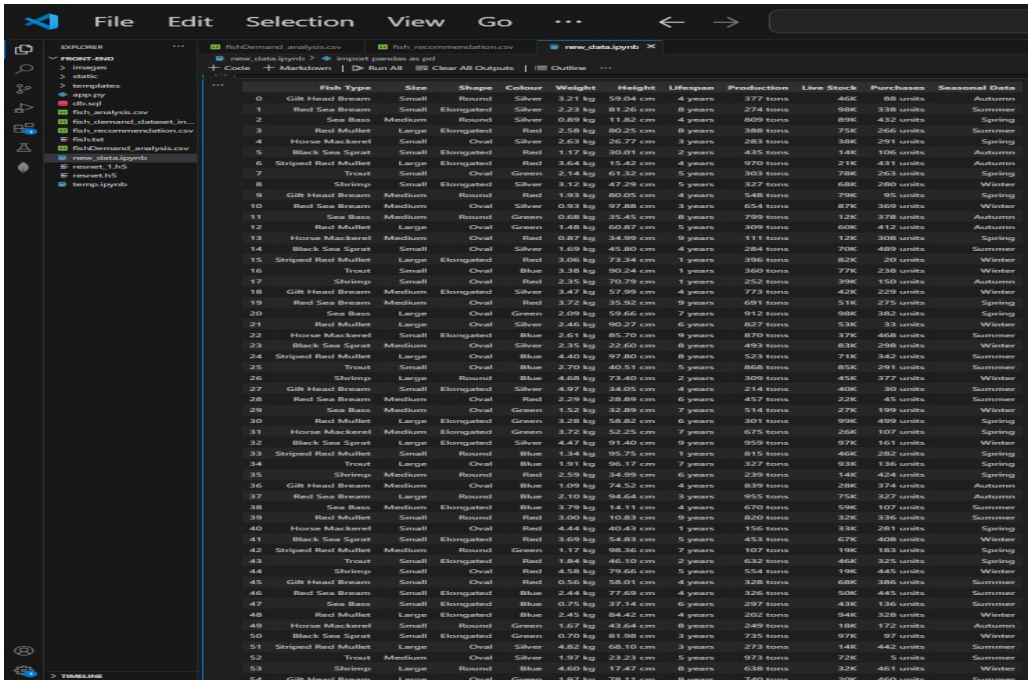


Fig.3.4.1.3 Datasets for Recommendation System

3.4.2 Data Pre-Processing

Data pre-processing is a crucial step in the above project, especially for preparing the input data for the fish species classification model. The pre-processing involves several tasks aimed at cleaning, transforming, and enhancing the raw data to make it suitable for training the deep learning model, particularly ResNet50.

1. Image Loading: The first step involves loading the raw image data into the system. This includes reading images from various file formats such as JPG, PNG, and JPEG. The images are then stored in memory or disk for further processing.

2. Image Cleaning: To ensure the quality and consistency of the image data, cleaning procedures are applied. This may involve handling potential image artifacts, such as noise or distortion, which could affect the accuracy of the classification model. Techniques like denoising and artifact removal may be employed to improve image quality.

3. Image Normalization: Normalizing pixel values is essential to standardize the intensity levels across different images. Normalization ensures that each pixel value falls within a consistent range, typically between 0 and 1 or -1 and 1. This step helps in stabilizing the training process and improving the convergence of the deep learning model.

4. Image Resizing: Resizing images to a consistent size is necessary for compatibility with the input requirements of the deep learning model. Resizing ensures that all images have the same dimensions, allowing them to be fed into the model without issues. Resizing may involve cropping or padding images to achieve the desired dimensions.

5. Data Augmentation: Data augmentation techniques may be applied to increase the variability of the training data and prevent overfitting. Augmentation techniques such as rotation, flipping, scaling, and translation are commonly used to create additional training samples from existing data.

6. Label Encoding: In supervised learning tasks like image classification, label encoding is performed to convert categorical labels (fish species names) into numerical format that the model can understand. Each fish species is assigned a unique numerical identifier, facilitating model

training and evaluation.

7. Data Splitting: After pre-processing, the dataset is typically divided into training, validation, and testing sets. The training set is used to train the model, the validation set is used to tune hyperparameters and monitor training progress, and the testing set is used to evaluate the model's performance on unseen data.

By performing these pre-processing steps, the raw image data is transformed into a clean, standardized, and augmented dataset suitable for training the fish species classification model. This ensures that the model can effectively learn the visual features and patterns associated with different fish species, ultimately leading to accurate classification results.

3.4.3 Training Data

The training data for the project likely consists of a dataset containing images of various fish species along with corresponding labels indicating the species of each fish. Elaborating on this:

1. Image Data:

- The image data comprises a collection of images, each depicting a fish of a specific species.

- These images are likely to be in a digital format such as JPG or PNG.

- Each image may vary in dimensions, color, and orientation, representing the diversity inherent in real-world fish images.

2. Label Data:

- Labels are associated with each image and indicate the species of the fish depicted in the corresponding image.

- These labels serve as ground truth information for training the machine learning model.

- Each label corresponds to a specific class or category, representing a distinct fish species.

3. Data Size and Distribution:

- The size of the training dataset may vary depending on factors such as the number of distinct fish species targeted for classification and the availability of annotated images.

- Ideally, the dataset should encompass a diverse range of fish species to ensure that the model learns to differentiate between different species effectively.

- The distribution of images across different classes should be balanced to prevent bias and ensure that the model receives sufficient exposure to each class during training.

4. Preprocessing:

- Prior to training, the training data undergoes preprocessing to standardize and enhance its quality.

- Preprocessing steps may include resizing images to a uniform size, normalization of pixel values, and augmentation techniques such as rotation, flipping, or cropping to increase the variability of the training data.

- Additionally, techniques to handle class imbalance, address missing or corrupted images, and remove noise may be applied to ensure the robustness of the training process.

5. Data Annotation:

- Annotation of the training data involves associating each image with its corresponding label or class.

- This annotation process may be performed manually by human annotators or through automated tools, depending on the availability of resources and the size of the dataset.

- Accurate and consistent annotation is crucial for training a high-performing machine learning model.

Overall, the training data serves as the foundation for training the machine learning model to accurately classify fish species based on input images. The quality, diversity, and representativeness of the training data play a significant role in determining the performance and generalization ability of the trained model.

3.4.4 Testing Data

For the fish species classification project described above, the testing data would typically consist of a separate set of images of various fish species. These images would be distinct from the training data used to train the machine learning model and would serve to evaluate the model's performance on unseen data.

The testing data would ideally cover a diverse range of fish species and variations in factors such as lighting conditions, angles, and backgrounds to ensure the model's robustness and generalization capability. Additionally, the testing data should include images of fish species that were not present in the training data to assess the model's ability to accurately classify unseen species.

Each image in the testing data would be labeled with the corresponding fish species to facilitate evaluation of the model's classification accuracy. After the model is trained using the training data, it would be evaluated on the testing data to measure metrics such as accuracy, precision, recall, and F1 score, which provide insights into the model's performance.

The testing data plays a crucial role in assessing the model's performance and identifying any potential issues such as overfitting or underfitting. By using separate training and testing datasets, researchers can ensure an unbiased evaluation of the model's ability to generalize to new, unseen data, thus providing confidence in its effectiveness for real-world applications.

3.4.5 User Interface and Human factors

To effectively insert, test, and train the dataset used in our project, human involvement is indispensable. Specifically, we rely on human feedback and reviews to refine the performance of our model. This necessitates the development of a user interface that facilitates interaction with human users, enabling them to provide reviews and feedback on the products or items included in the dataset.

The user interface serves as a bridge between the dataset and the human factor, allowing for seamless communication and collaboration. Through this interface, users can easily input their reviews, opinions, and ratings for various products or items present in the dataset. These reviews

are crucial for both training and testing the model, as they provide valuable insights into the performance and accuracy of the system.

Through the user interface, individuals can effortlessly input their reviews, opinions, and ratings for the various products or items encompassed within the dataset. These user-generated reviews are of paramount importance for both training and testing the machine learning model, furnishing invaluable insights into its performance and accuracy. They contribute to the iterative process of model development, aiding in the creation of a robust and precise predictive model.

For training purposes, the user interface enables users to submit reviews that are used to train the machine learning model. These reviews contribute to the development of a robust and accurate model by providing diverse perspectives and feedback on the dataset. The training process involves iteratively adjusting the model based on the received reviews to improve its performance and predictive capabilities.

During the training phase, the user interface facilitates the submission of reviews, which are utilized to train the machine learning model. This training process involves adjusting the model iteratively based on the feedback received from users, thereby enhancing its predictive capabilities and overall performance. By incorporating diverse perspectives and feedback from users, the model becomes more adept at accurately predicting outcomes.

Similarly, during the testing phase, the user interface allows users to input reviews that are used to evaluate the model's performance. By comparing the model's predictions with the actual reviews provided by users, we can assess its accuracy, effectiveness, and generalization capabilities. This feedback loop ensures that the model is continually refined and optimized to deliver accurate predictions and recommendations.

In summary, the user interface serves as a vital component in the dataset insertion, training, and testing processes by facilitating interaction with human users. Through this interface, users can provide valuable reviews and feedback that are essential for improving the performance and effectiveness of our machine learning model.

Chapter 4

SYSTEM STUDY

4.1 Feasibility Study

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

- Technical feasibility
- Economical feasibility
- Operation feasibility
- Social feasibility

4.1.1 Technical Feasibility

This study is carried out to check the technical feasibility, that is, the technical 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 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 implementing this system.

4.1.2 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 available. Only the customized products had to be purchased.

4.1.3 Operation Feasibility

We investigate the practicality and usability of the proposed system from an operational perspective. This involves considering factors such as user acceptance, ease of use, and potential disruptions to existing workflows. By understanding how the system will integrate into the company's operations, we can identify any potential challenges and develop strategies to mitigate them.

4.1.4 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.

Chapter 5

SYSTEM DESIGN

5.1 System Architecture

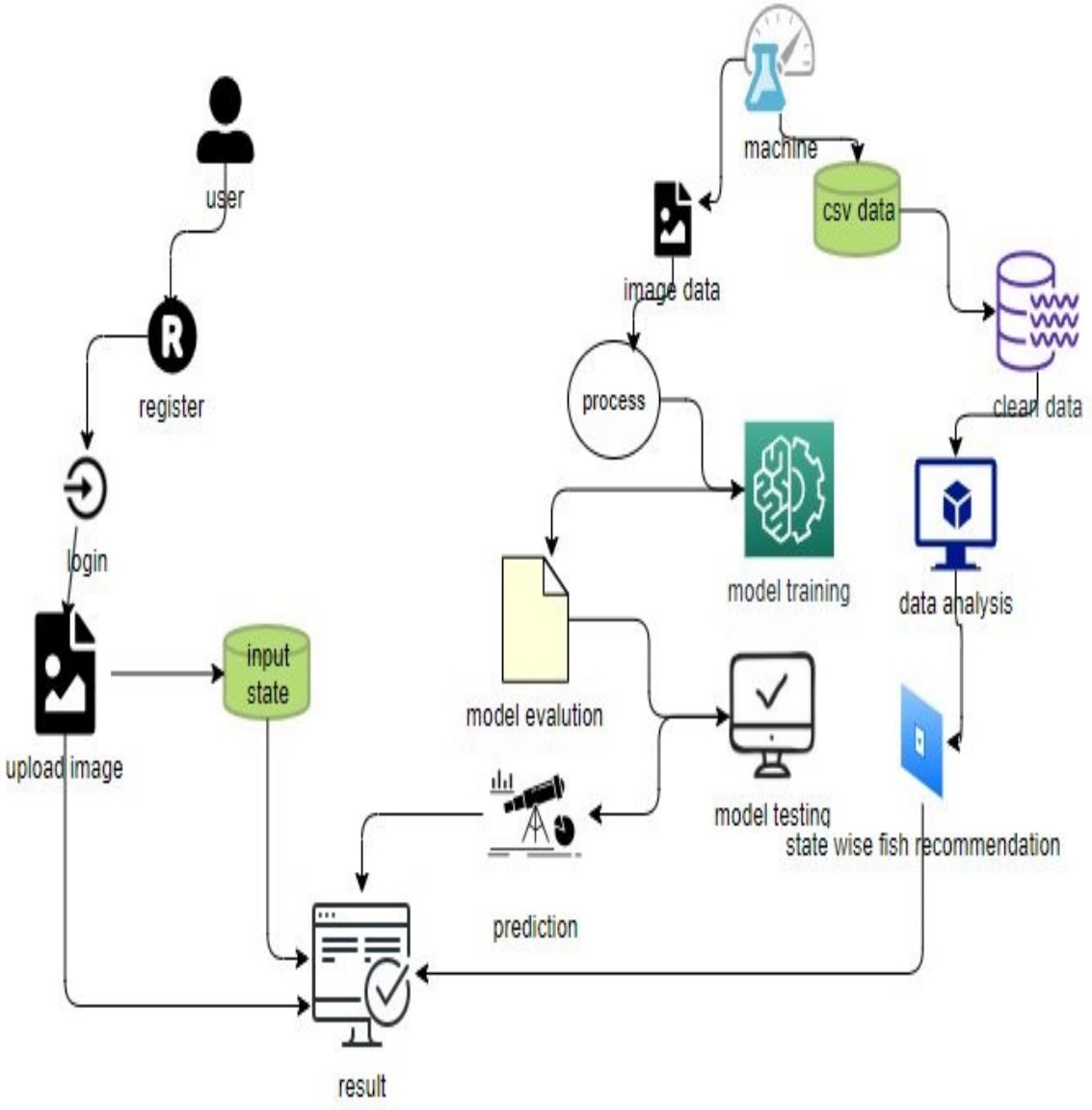


Fig.5.1 System Architecture

5.2 UML Diagrams

Use case diagrams are usually referred to as behavior diagrams used to describe a set of actions (use cases) that some system or systems (subject) should or can perform in collaboration with one or more external users of the system (actors). A use case is a methodology used in system analysis to identify, clarify, and organize system requirements.

The use case is made up of a set of possible sequences of interactions between systems and users in a particular environment and related to a particular goal. It consists of a group of elements (for example, classes and interfaces) that can be used together in a way that will have an effect larger than the sum of the separate elements combined.

The use case should contain all system activities that have significance to the users. A use case can be thought of as a collection of possible scenarios related to a particular goal, indeed, the use case and goal are sometimes considered to be synonymous. The main purpose of a use case diagram is to show what system functions are performed for which actor.

Actors:

Actors represent roles which may include human users, external hardware, or other systems. Our proposed system is quite simple in terms of user interface. It is only having two actor's user and the system.

Following are the actors in our project -

a) User 

User is the actor who can enter into the system by entering login details for testing the diseases of fruit using images.

b) System 

System is an actor that interacts with the user. The system will perform several actions like user verification, displaying messages to the user.

c)Admin

Admin can do training, run machine learning algorithms and testing on the dataset.

5.2.a List of use cases

Use case name	Actors
Login	User and Admin
Upload Dataset	Admin
Process Dataset Features	Admin
Run all Machine Learning Algorithms	Admin
Average ranking graph	Admin
Upload image and test	User
Display output	System

Table 5.2.a List of use cases and actors associated with

Use case name	Description
Login	The user can login into the system
Upload Dataset	The dataset is uploaded by the Admin
Process dataset features	Different features are extracted
Run all ML algorithms	After features extraction, the system should run the algorithms
Average ranking graph	A graph is produced based on classification
All train graph	The user can select the parking slot if it is available

Upload image and test	User is tests the disease of the fruit by uploading the image
Display output	The system displays the output

Table 5.2.b Use cases and descriptions

5.2.1 Use Case Diagram

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

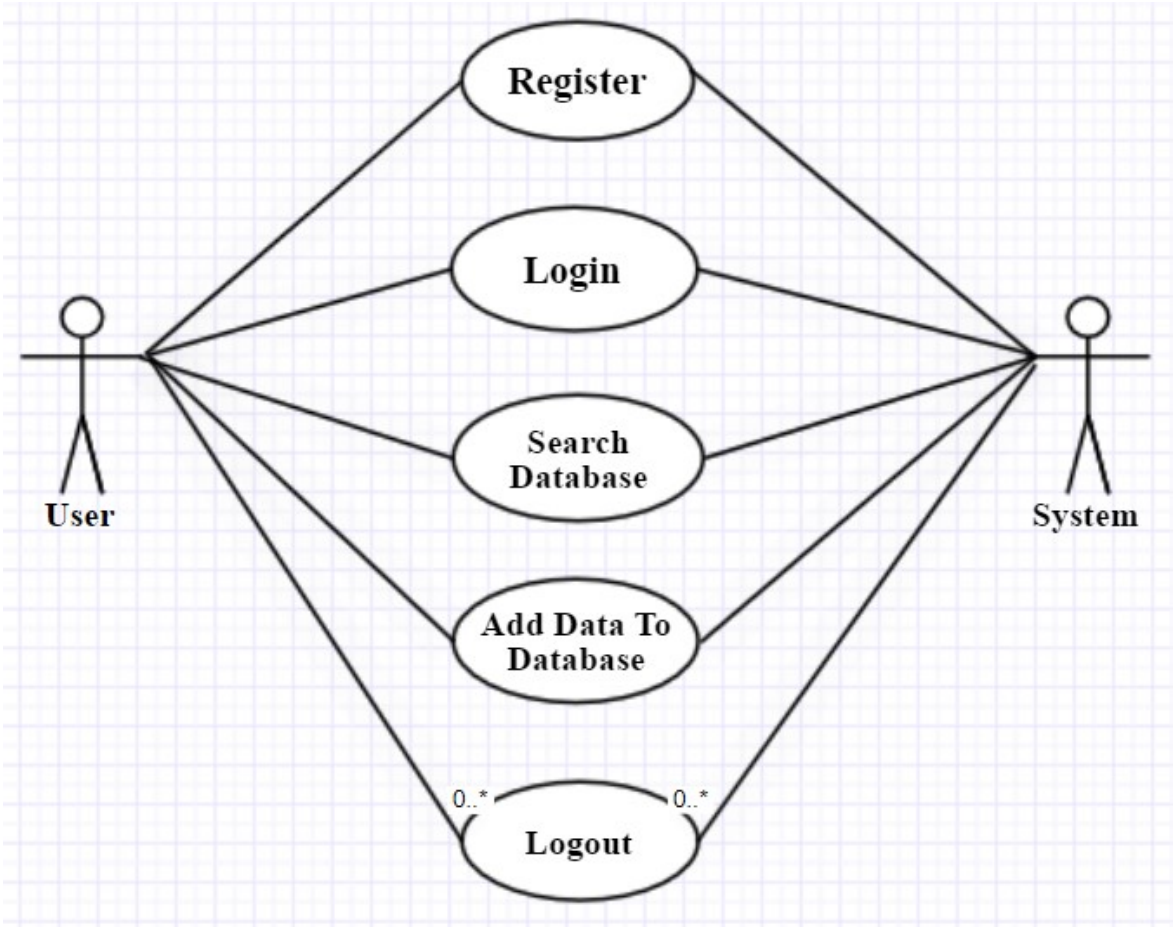


Fig. 5.2.1(a)Use case diagram

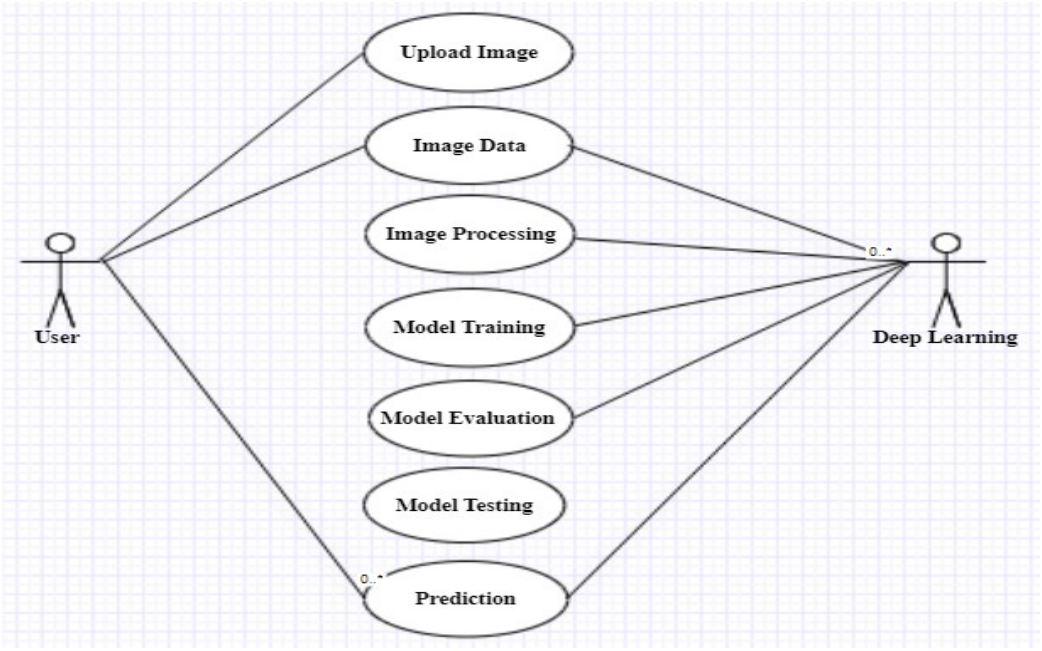


Fig. 5.2.1(b) Use case diagram

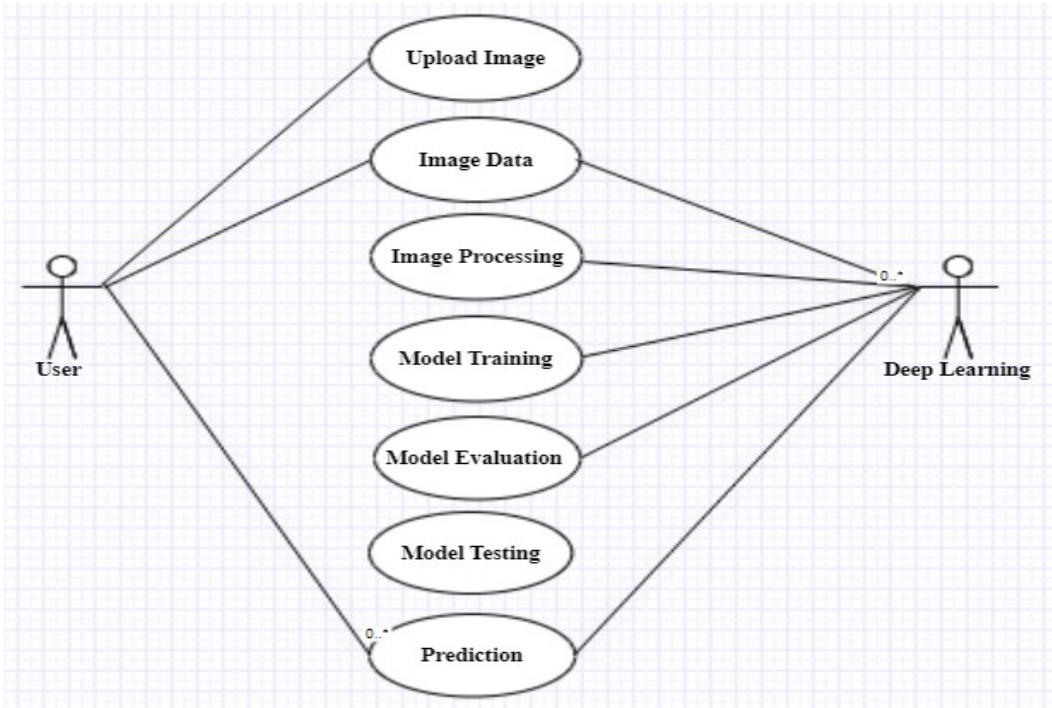


Fig. 5.2.1(c) Use case diagram

5.2.2 Class Diagram

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

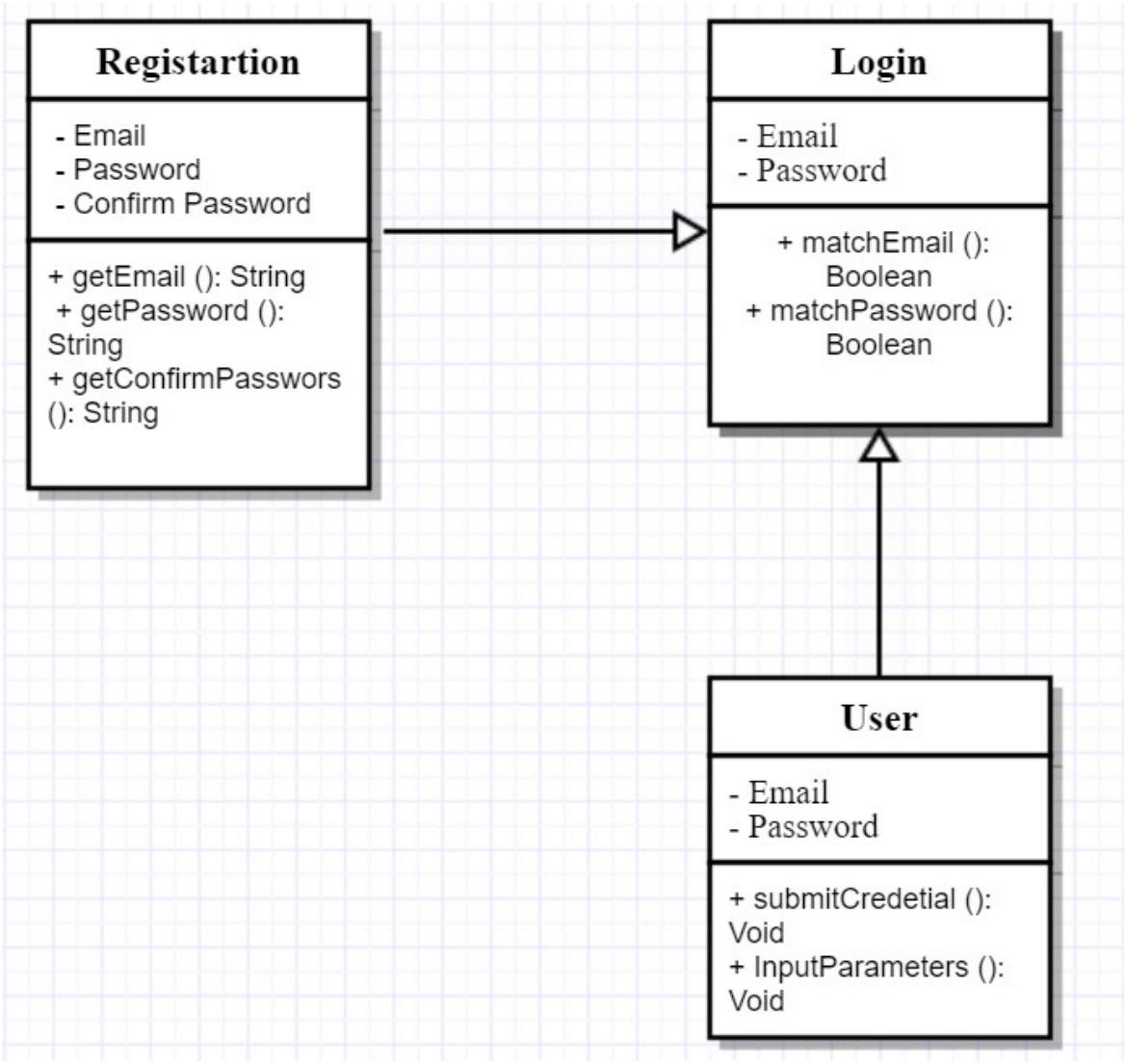


Fig. 5.2.2(a) Class diagram

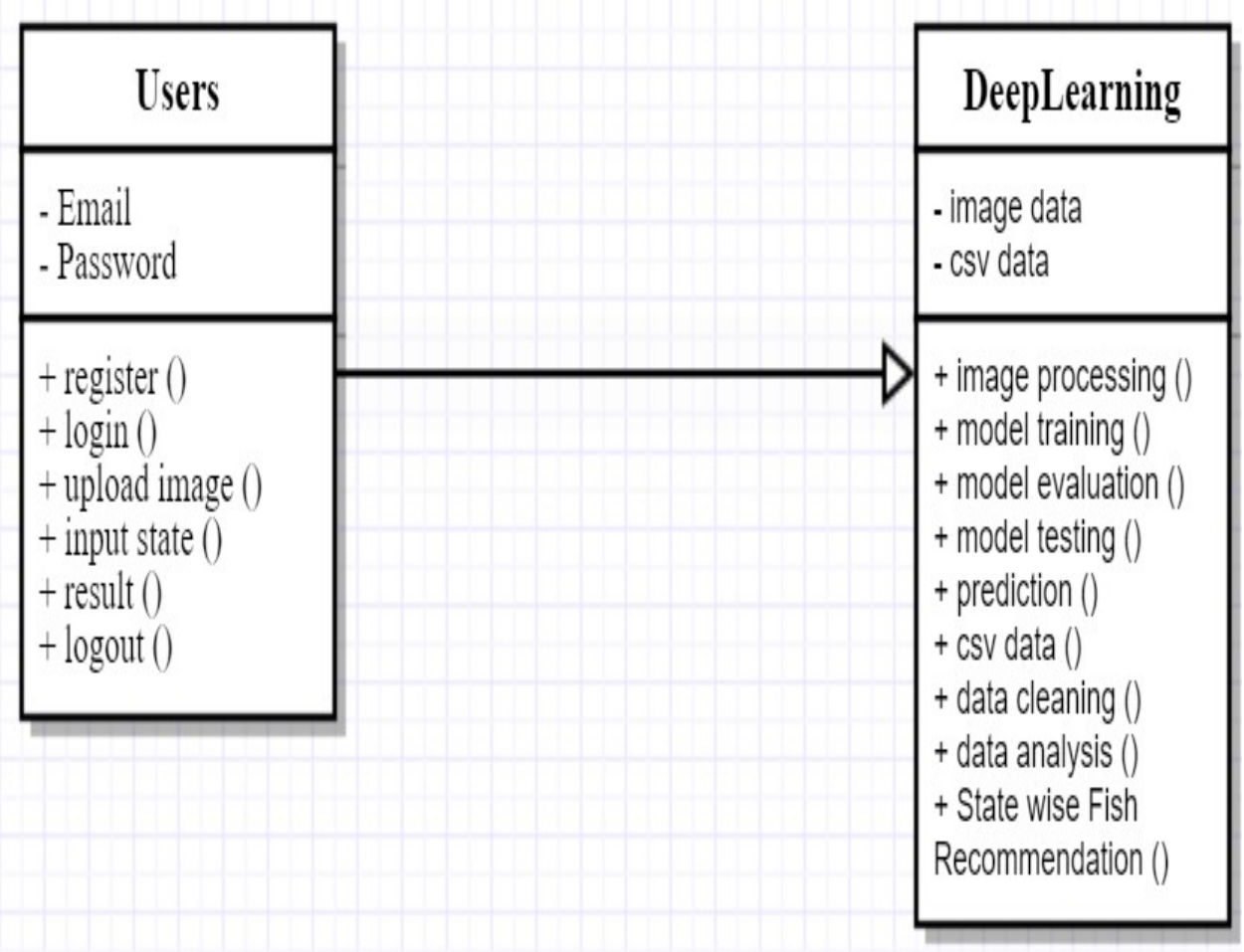


Fig. 5.2.2(b) Class diagram

5.2.3 Sequence Diagram

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

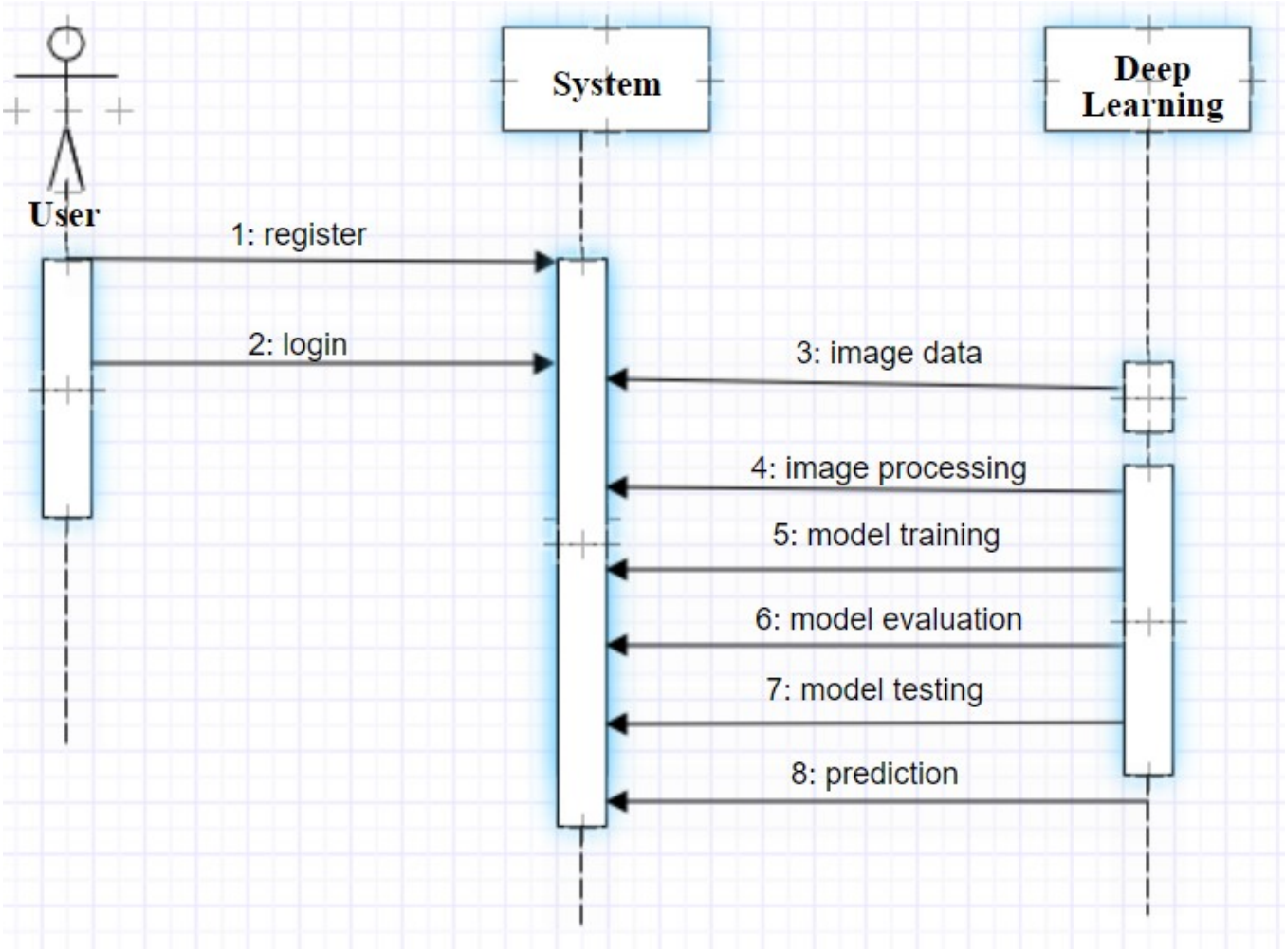


Fig.5.2.3(a) Sequence diagram

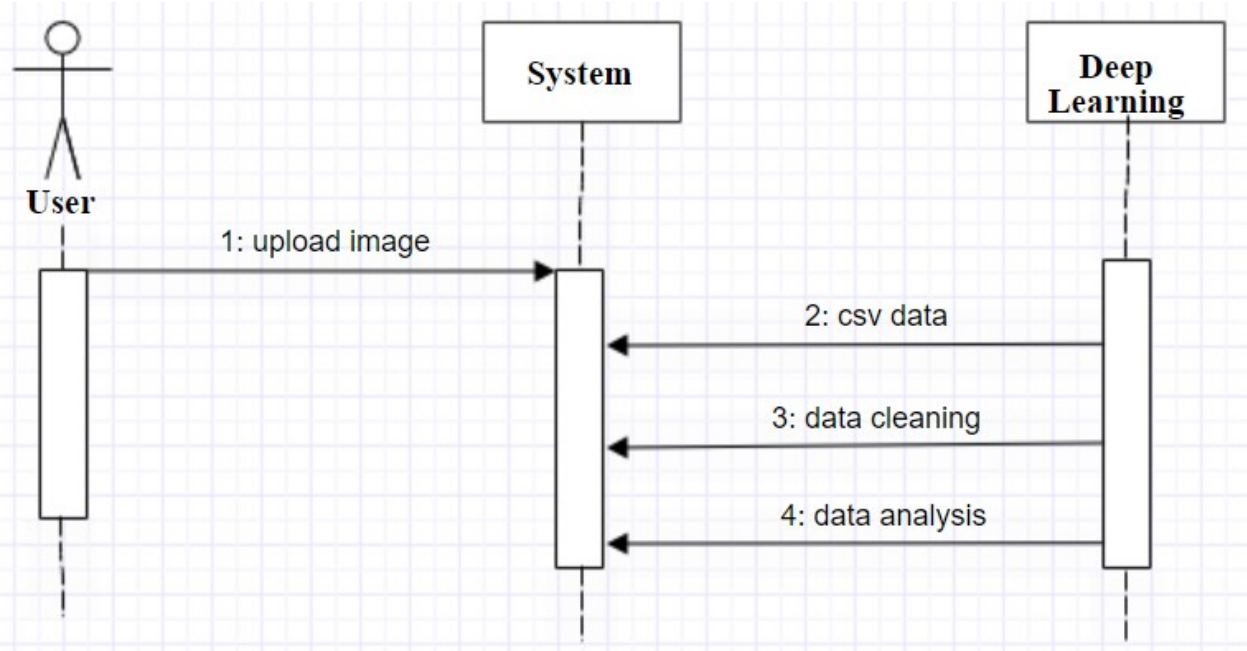


Fig.5.2.3(b) Sequence diagram

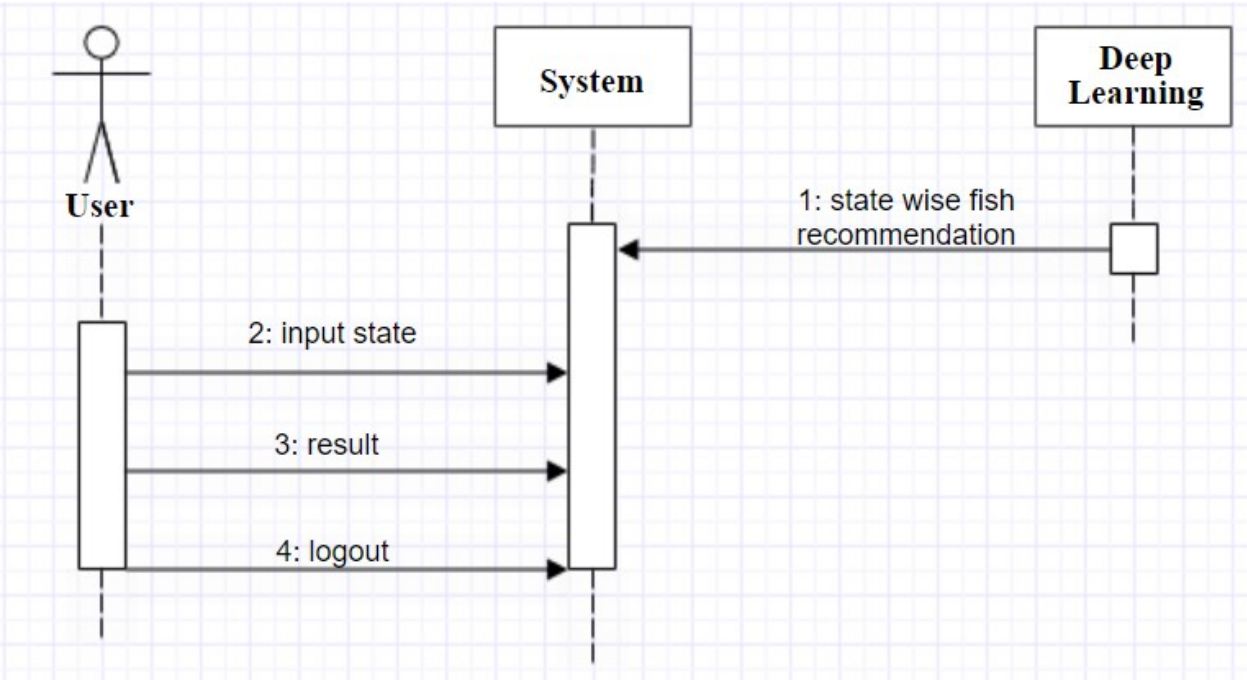


Fig.5.2.3(c) Sequence diagram

5.2.4 Activity Diagram

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

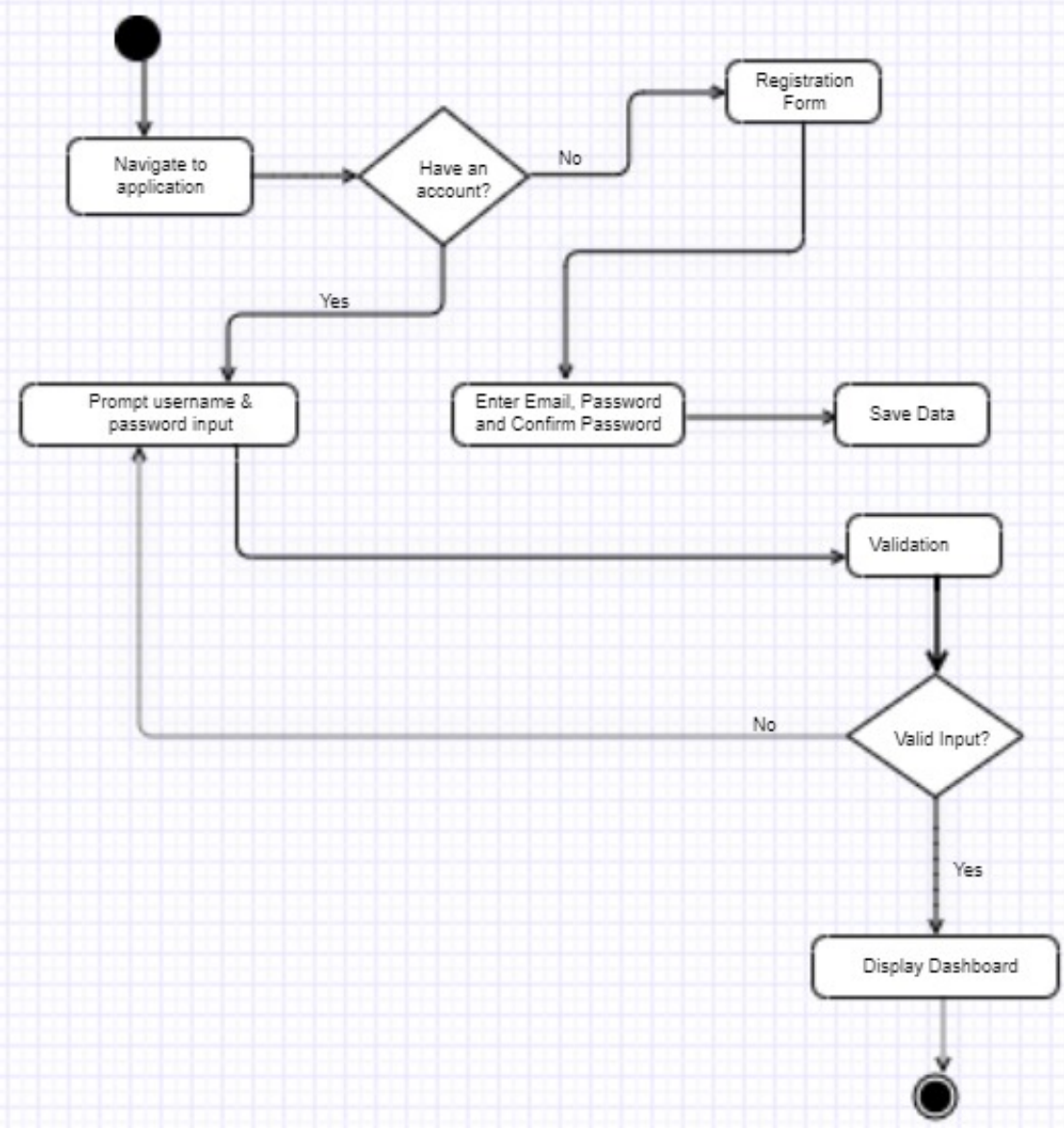


Fig.5.2.4(a) Activity diagram

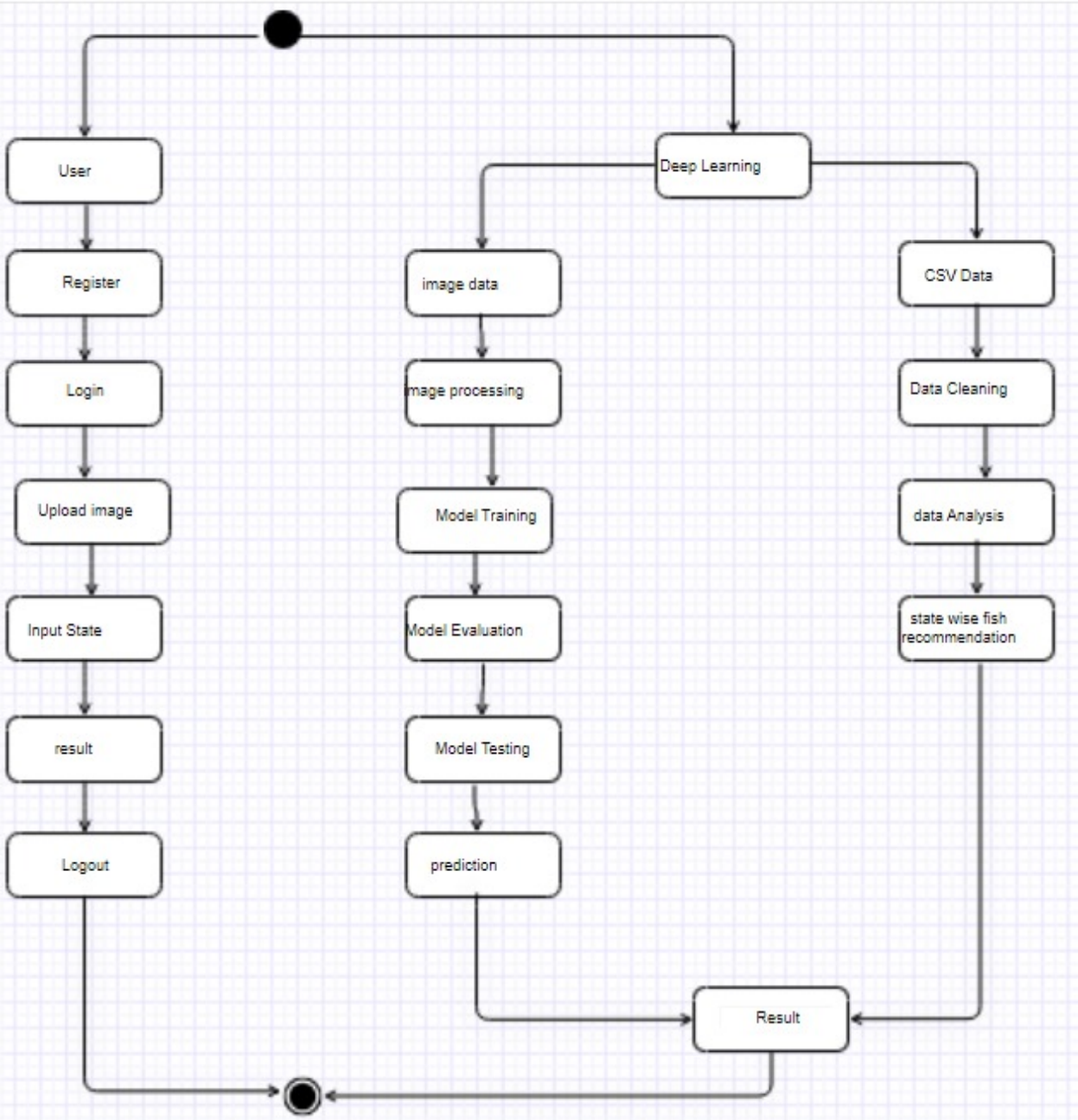


Fig.5.2.4(b) Activity diagram

5.2.5 Component Diagram

A component diagram, also known as a UML component diagram, describes the organization and wiring of the physical components in a system. Component diagrams are often drawn to help model implementation details and double-check that every aspect of the system's required functions is covered by planned development.

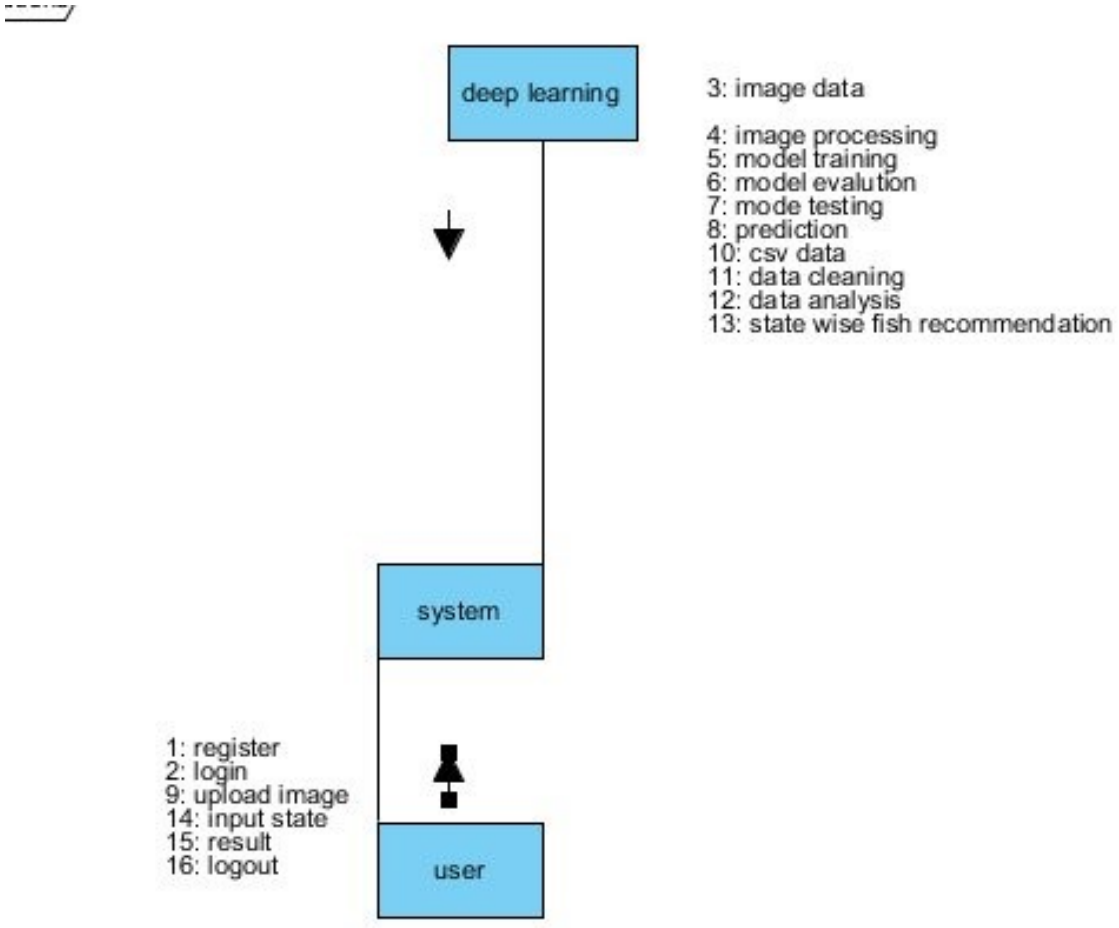


Fig.5.2.5 Component diagram

5.2.6 Collaboration Diagram

In collaboration diagram the method call sequence is indicated by some numbering technique as shown below. The number indicates how the methods are called one after another. We have taken the same order management system to describe the collaboration diagram. The method calls are similar to that of a sequence diagram. But the difference is that the sequence diagram does not describe the object organization whereas the collaboration diagram shows the object organization.

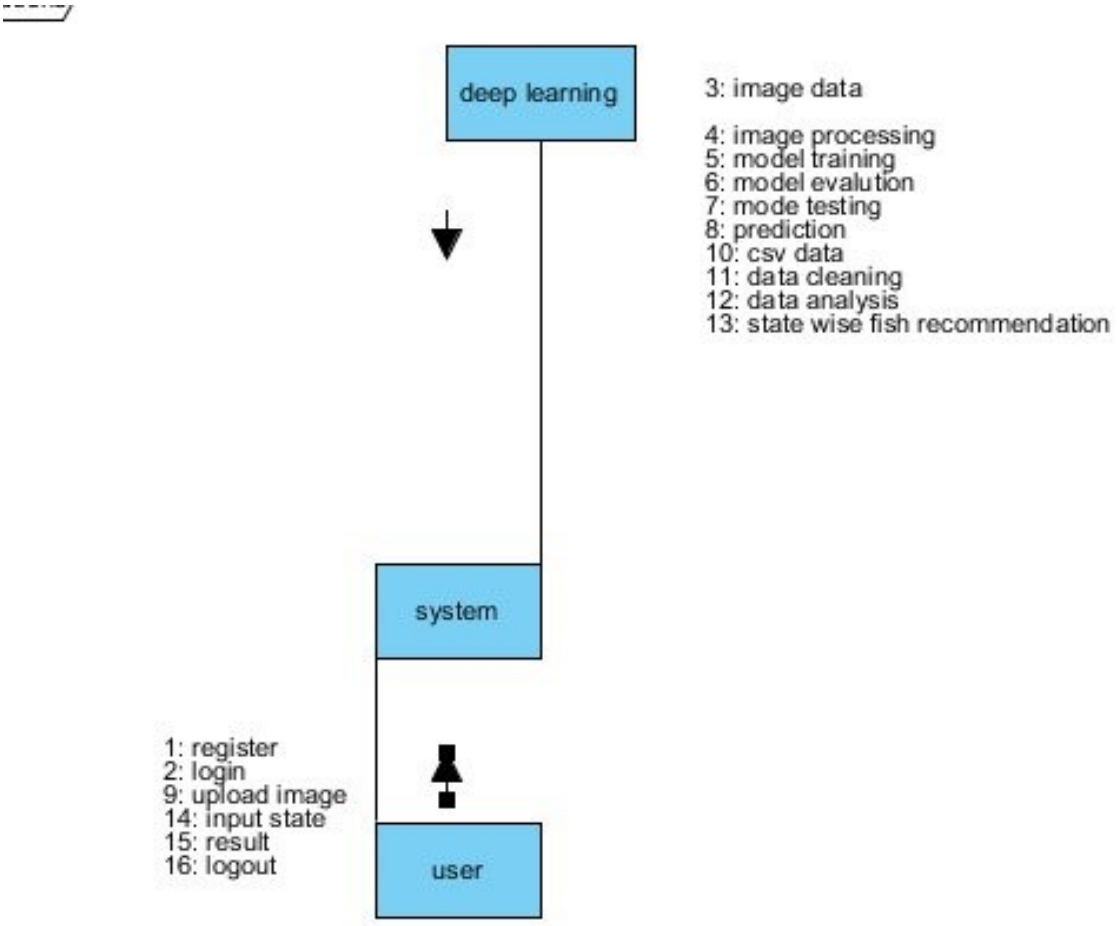


Fig.5.2.6 Collaboration diagram

5.2.7 Deployment Diagram

Deployment diagram represents the deployment view of a system. It is related to the component diagram. Because the components are deployed using the deployment diagrams. A deployment diagram consists of nodes. Nodes are nothing but physical hardware's used to deploy the application.

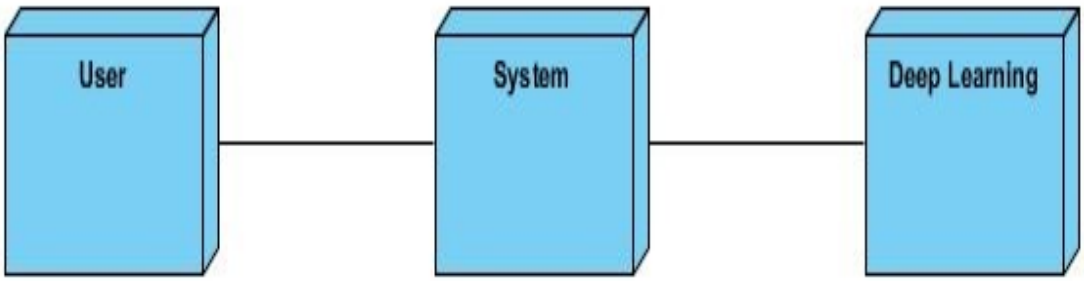


Fig.5.2.7 Deployment diagram

5.2.8 Entity Relationship Diagram

An Entity–relationship model (ER model) describes the structure of a database with the help of a diagram, which is known as Entity Relationship Diagram (ER Diagram). An ER model is a design or blueprint of a database that can later be implemented as a database. The main components of E-R model are: entity set and relationship set.

An ER diagram shows the relationship among entity sets. An entity set is a group of similar entities, and these entities can have attributes. In terms of DBMS, an entity is a table or attribute of a table in database, so by showing relationship among tables and their attributes, ER diagram shows the complete logical structure of a database. Let's have a look at a simple ER diagram to understand this concept.

Fig.5.2.8 ER diagram

5.2.9 DataFlow Diagram

A Data Flow Diagram (DFD) is a traditional way to visualize the information flows within a system. A neat and clear DFD can depict a good amount of the system requirements graphically. It can be manual, automated, or a combination of both. It shows how information enters and leaves the system, what changes the information and where information is stored. The purpose of a DFD is to show the scope and boundaries of a system as a whole. It may be used as a communications tool between a systems analyst and any person who plays a part in the system that acts as the starting point for redesigning a system.

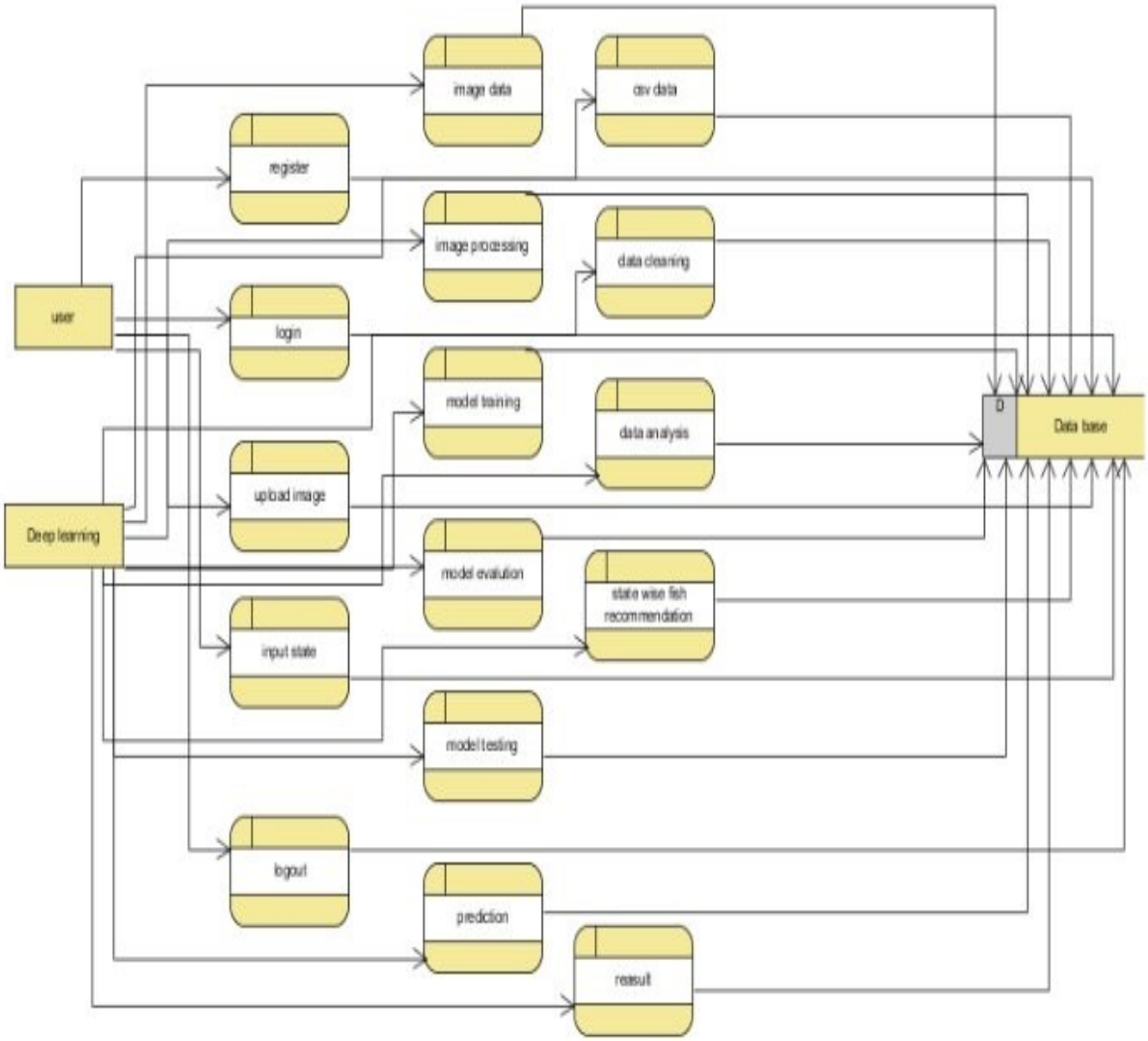


Fig.5.2.9 (a) DFD diagram level 0

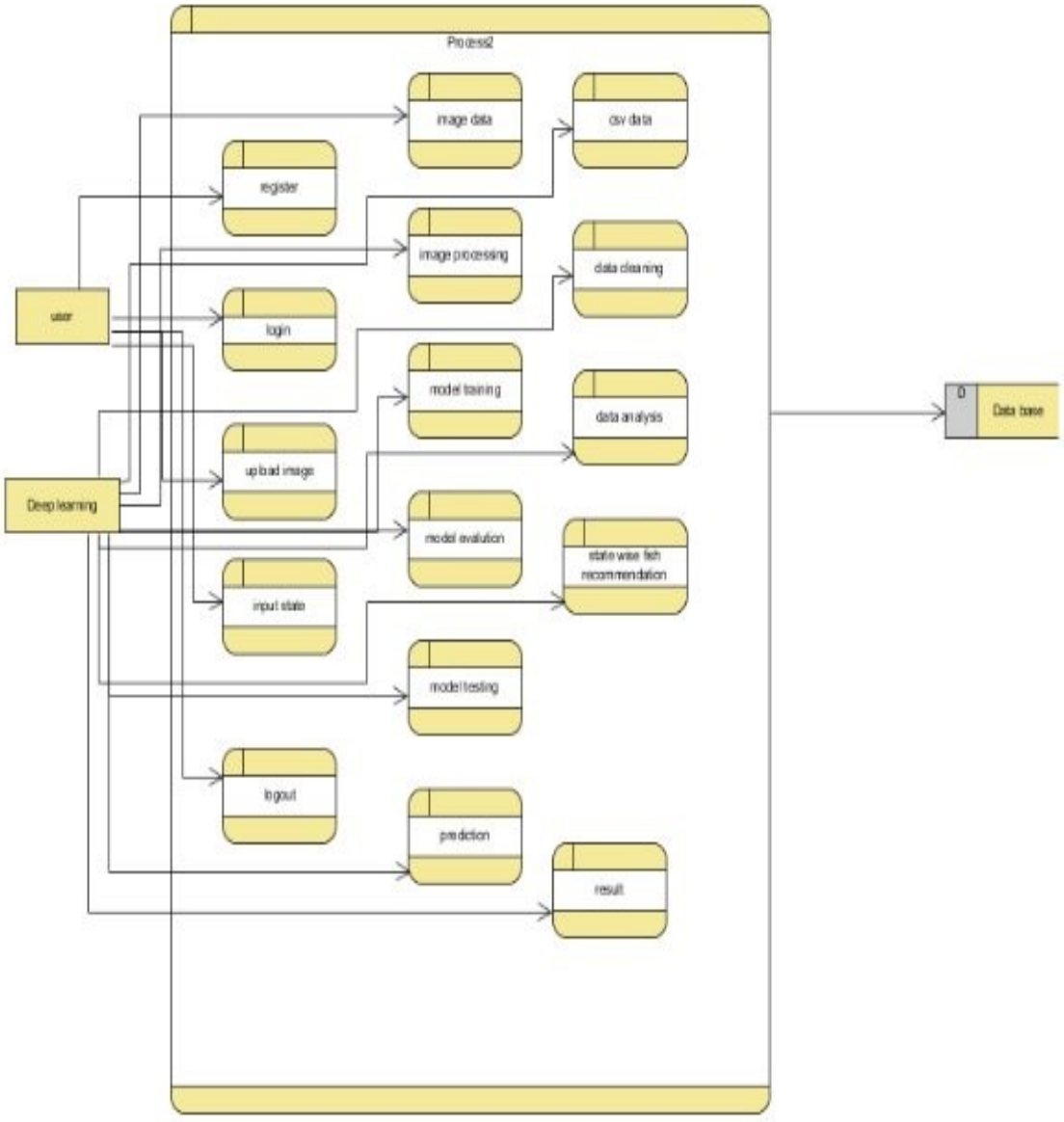


Fig. 5.2.9 (b) DFD diagram level 1

Chapter 6

REQUIREMENT ANALYSIS

6.1 System Requirements

A requirement is a feature that the system must have or a constraint that it must to be accepted by the client. Requirement Engineering aims at defining the wants of the system under construction. Requirement Engineering include two main activities requirement elicitation which results in the specification of the system that the client understands and analysis which in analysis model that the developer can unambiguously interpret.

A requirement may be a statement about what the proposed system will do. Requirements can be divide into two major categories:

1. Functional Requirements.

2. Non-Functional Requirements.

6.2 Functional Requirements

- Data Collection: Collect sufficient data samples and legitimate software samples.

- Data Preporcessing: Perform effective data processing on the sample and extract the features.

- Train and Test Modelling: Split the data into train and test data Train will be used for training the model and Test data to check the performace.

- Feature Selection: Further select the main features for classification.

- Modelling: SVD, ResNet50. Combine the training using machine learning algorithms and

establish a classification and recommendation model.

6.3 Non-Functional Requirements

NON-FUNCTIONAL REQUIREMENT (NFR) specifies the quality attribute of a software system. They judge the software system based on Responsiveness, Usability, Security, Portability and other non-functional standards that are critical to the success of the software system. Example of nonfunctional requirement, “how fast does the website load?” Failing to meet non-functional requirements can result in systems that fail to satisfy user needs.Nonfunctional Requirements allows you to impose constraints or restrictions on the design of the system across the various agile backlogs. Example, the site should load in 3 seconds when the number of simultaneous users are > 10000. Description of non-functional requirements is just as critical as a functional requirement.

1. Usability requirement
2. Serviceability requirement
3. Manageability requirement
4. Recoverability requirement
5. Security requirement
6. Data Integrity requirement
7. Capacity requirement
8. Availability requirement
9. Scalability requirement
10. Interoperability requirement
11. Reliability requirement
12. Maintainability requirement
13. Regulatory requirement
14. Environmental requirement

6.3.1 User Interface and Human Factors

To effectively insert, test, and train the dataset used in our project, human involvement is indispensable. Specifically, we rely on human feedback and reviews to refine the performance of our model. This necessitates the development of a user interface that facilitates interaction with

human users, enabling them to provide reviews and feedback on the products or items included in the dataset.

The user interface serves as a bridge between the dataset and the human factor, allowing for seamless communication and collaboration. Through this interface, users can easily input their reviews, opinions, and ratings for various products or items present in the dataset. These reviews are crucial for both training and testing the model, as they provide valuable insights into the performance and accuracy of the system.

Through the user interface, individuals can effortlessly input their reviews, opinions, and ratings for the various products or items encompassed within the dataset. These user-generated reviews are of paramount importance for both training and testing the machine learning model, furnishing invaluable insights into its performance and accuracy. They contribute to the iterative process of model development, aiding in the creation of a robust and precise predictive model.

For training purposes, the user interface enables users to submit reviews that are used to train the machine learning model. These reviews contribute to the development of a robust and accurate model by providing diverse perspectives and feedback on the dataset. The training process involves iteratively adjusting the model based on the received reviews to improve its performance and predictive capabilities.

During the training phase, the user interface facilitates the submission of reviews, which are utilized to train the machine learning model. This training process involves adjusting the model iteratively based on the feedback received from users, thereby enhancing its predictive capabilities and overall performance. By incorporating diverse perspectives and feedback from users, the model becomes more adept at accurately predicting outcomes.

Similarly, during the testing phase, the user interface allows users to input reviews that are used to evaluate the model's performance. By comparing the model's predictions with the actual reviews provided by users, we can assess its accuracy, effectiveness, and generalization capabilities. This feedback loop ensures that the model is continually refined and optimized to deliver accurate predictions and recommendations.

In summary, the user interface serves as a vital component in the dataset insertion, training, and

testing processes by facilitating interaction with human users.

6.3.2 Software Requirements

- Operating System : Windows 7/8/10/11
- Server-side Script : HTML, CSS, Bootstrap & JS
- Programming Language : Python
- Libraries : Flask, Pandas, TensorFlow, Keras, Sklearn, Numpy
- IDE/Workbench : VSCode
- Technology : Python 3.6+
- Server Deployment : Xampp Server
- Database : MySQL

6.3.3 Hardware Requirements

- Processor : I3/Intel Processor
- RAM : 8GB (min)
- Hard Disk : 128 GB
- Key Board : Standard Windows Keyboard
- Mouse : Two or Three Button Mouse
- Monitor : Any

6.3.4 Usability

Usability is a critical aspect of the integrated fish species classification and recommendation project, ensuring that the system is intuitive, efficient, and enjoyable for users across diverse backgrounds and skill levels. The user interface should be designed with a focus on simplicity and clarity, allowing users to navigate the system effortlessly and perform tasks with minimal effort.

6.3.5 Reliability

The system must exhibit high reliability and availability to ensure uninterrupted service delivery to users. This involves minimizing system downtime, implementing fault tolerance mechanisms, and conducting regular maintenance and updates. Reliability testing, including stress testing and failure simulation, should be performed to identify and address potential points of failure and ensure the system's robustness under adverse conditions.

6.3.6 Performance

The system must demonstrate high performance in terms of classification accuracy, recommendation generation speed, and overall responsiveness. It should be able to handle concurrent user requests efficiently without compromising speed or accuracy. Performance benchmarks and testing protocols should be established to evaluate and validate the system's performance under varying workload conditions.

6.3.7 Supportability

Ensuring the long-term supportability and maintenance of the integrated system is crucial for its continued effectiveness and relevance in the fisheries sector. This involves establishing robust support mechanisms, including documentation, training materials, and technical assistance channels, to facilitate system adoption and user empowerment. Regular updates and enhancements based on user feedback and evolving industry requirements are essential to address emerging challenges and opportunities.

6.3.8 Physical Environment

The integrated system for fish species classification and recommendation requires a suitable physical environment to operate effectively. This includes access to reliable power sources and stable network connectivity to ensure uninterrupted system functionality. Adequate ventilation and temperature control measures should be in place to maintain optimal operating conditions for hardware components, such as servers or computing devices, especially in data processing and storage facilities. Additionally, physical security measures, such as restricted access to server rooms or data centers, should be implemented to safeguard against unauthorized access, theft, or damage to system infrastructure.

6.3.9 Security Requirements

Security is paramount for protecting sensitive data, ensuring system integrity, and mitigating potential risks associated with cyber threats or unauthorized access. The integrated system must adhere to industry best practices and standards for data security, including encryption of data transmission and storage, user authentication mechanisms, access control policies, and regular security audits. Implementation of intrusion detection and prevention systems, firewalls, and malware protection software is essential to detect and mitigate security threats in real-time. Compliance with relevant regulatory requirements, such as GDPR or HIPAA, should be ensured to safeguard user privacy and data confidentiality.

6.3.9.1 Access Requirements

Access requirements pertain to controlling and regulating the access to the system's resources and functionalities. This involves implementing user authentication mechanisms to verify the identity of users before granting them access to the system. Access control policies should be established to manage user privileges based on their roles and responsibilities within the system. Additionally, measures such as role-based access control (RBAC) can be employed to ensure that users only have access to the resources and functionalities necessary for their specific roles. Access requirements also extend to ensuring secure remote access to the system, including mechanisms such as virtual private networks (VPNs) or secure sockets layer (SSL) encryption for data transmission.

6.3.9.2 Integrity Requirements

Integrity requirements focus on maintaining the accuracy, consistency, and reliability of data within the system. This involves implementing measures to prevent unauthorized modification, deletion, or corruption of data. Techniques such as data validation and verification can be used to ensure that only valid and authorized changes are made to the data. Additionally, cryptographic techniques such as digital signatures or hash functions can be employed to detect and prevent tampering with data during transmission or storage. Data integrity controls should be enforced at both the system and application levels to ensure that data remains trustworthy and reliable throughout its lifecycle.

6.3.9.3 Private Requirements

Privacy requirements address the protection of sensitive user information and ensuring compliance with relevant privacy regulations and standards. This involves implementing measures to safeguard the confidentiality and privacy of user data, including encryption of data transmission and storage, anonymization or pseudonymization of personally identifiable information (PII), and access controls to restrict access to sensitive data to authorized personnel only. Privacy requirements also extend to providing users with transparency and control over their personal data, including mechanisms for obtaining user consent for data collection, processing, and sharing. Compliance with privacy regulations such as the General Data Protection Regulation (GDPR) or the California Consumer Privacy Act (CCPA) is essential to ensure that user privacy rights are protected and respected.

6.3.10 Resource Requirements

The successful operation of the integrated system relies on various resources, including hardware, software, and human resources. Hardware requirements may include high-performance computing resources, such as servers with sufficient processing power and memory capacity to support deep learning algorithms and data processing tasks. Storage infrastructure must accommodate large volumes of image data, metadata, and user information securely. Software requirements encompass the development and deployment of custom software applications for image classification, recommendation algorithms, and user interface design. Human resources are essential for system development, maintenance, and support, including skilled professionals proficient in machine learning, software engineering, database management, and user experience design.

Chapter 7

IMPLEMENTATION

7.1 Software Used

7.1.1 Python

1. Below are some facts about Python.
2. Python is currently the most widely used multi-purpose, high-level programming language.
3. Python allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java.
4. Programmers must type relatively less and indentation requirement of the language, makes them readable all the time.
5. Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber… etc.
6. The biggest strength of Python is huge collection of standard libraries which can be used for the following:
    - Machine Learning
    - Image processing (like OpenCV, Pillow)
    - Test frameworks
    - GUI Applications (like Flask, Tkinter, PyQt etc.)
    - Multimedia

Advantages of Python:

1. Extensive Libraries

Python downloads with an extensive library and it contain code for various purposes like regular expressions, documentation-generation, unit-testing, web browsers, threading, databases, CGI, email, image manipulation, and more. So, we don't have to write the complete code for that manually.

2. Extensible

As we have seen earlier, Python can be extended to other languages. You can write some of your code in languages like C++ or C. This comes in handy, especially in projects.

3. Embeddable

Complimentary to extensibility, Python is embeddable as well. You can put your Python code in your source code of a different language, like C++. This lets us add scripting capabilities to our code in the other language.

4. Improved Productivity

The language's simplicity and extensive libraries render programmers more productive than languages like Java and C++ do. Also, the fact that you need to write less and get more things

done.

5. IOT Opportunities

Since Python forms the basis of new platforms like Raspberry Pi, it finds the future bright for the Internet of Things. This is a way to connect the language with the real world.

6. Simple and Easy

When working with Java, you may have to create a class to print 'Hello World'. But in Python, just a print statement will do. It is also quite easy to learn, understand, and code. This is why when people pick up Python, they have a hard time adjusting to other more verbose languages like Java.

7. Readable

Because it is not such a verbose language, reading Python is much like reading English. This is the reason why it is so easy to learn, understand, and code. It also does not need curly braces to define blocks, and indentation is mandatory. These further aids the readability of the code.

8. Object-Oriented

This language supports both the procedural and object-oriented programming paradigms. While functions help us with code reusability, classes and objects let us model the real world. A class allows the encapsulation of data and functions into one.

9. Free and Open-Source

Like we said earlier, Python is freely available. But not only can you download Python for free, but you can also download its source code, make changes to it, and even distribute it. It downloads with an extensive collection of libraries to help you with your tasks.

10. Portable

When you code your project in a language like C++, you may need to make some changes to it if you want to run it on another platform. But it isn't the same with Python. Here, you need to code only once, and you can run it anywhere. This is called Write Once Run Anywhere (WORA). However, you need to be careful enough not to include any system-dependent features.

Advantages of Python over Other Languages

1. Less Coding

Almost all of the tasks done in Python requires less coding when the same task is done in other languages. Python also has an awesome standard library support, so you don't have to search for any third-party libraries to get your job done. This is the reason that many people suggest learning Python to beginners.

2. Affordable

Python is free therefore individuals, small companies or big organizations can leverage the free available resources to build applications. Python is popular and widely used so it gives you better community support. The 2019 Github annual survey showed us that Python has overtaken Java in the most popular programming language category.

3. Python is for Everyone

Python code can run on any machine whether it is Linux, Mac or Windows. Programmers need to learn different languages for different jobs but with Python, you can professionally build web apps, perform data analysis and machine learning, automate things, do web scraping and also build games and powerful visualizations. It is an all-rounder programming language.

Disadvantages of Python

Let's now see the downsides of choosing Python over another language.

1. Speed Limitations

We have seen that Python code is executed line by line. But since Python is interpreted, it often results in slow execution. This, however, isn't a problem unless speed is a focal point for the project. In other words, unless high speed is a requirement, the benefits offered by Python are enough to distract us from its speed limitation.

2. Weak in Mobile Computing and Browsers

While it serves as an excellent server-side language, Python is much rarely seen on the client-side. Besides that, it is rarely ever used to implement smartphone-based applications. One such application is called Carbonnelle.The reason it is not so famous despite the existence of Brython is that it isn't that secure.

3. Design Restrictions

As you know, Python is dynamically-typed. This means that you don't need to declare the type of variable while writing the code. It uses duck-typing. But wait, what's that? Well, it just means that if it looks like a duck, it must be a duck. While this is easy on the programmers during coding, it can raise run-time errors.

4. Underdeveloped Database Access Layers

Compared to more widely used technologies like JDBC (Java DataBase Connectivity) and ODBC (Open DataBase Connectivity), Python's database access layers are a bit underdeveloped. Consequently, it is less often applied in huge enterprises.

5. Simple

No, we're not kidding. Python's simplicity can indeed be a problem. Take my example. I don't do Java, I'm more of a Python person. To me, its syntax is so simple that the verbosity of Java code seems unnecessary.

7.1.1.1 Install Python Step-by-Step in Windows :

Python a versatile programming language doesn't come pre-installed on your computer devices. Python was first released in the year 1991 and until today it is a very popular high-level programming language. Its style philosophy emphasizes code readability with its notable use of great whitespace.

The object-oriented approach and language construct provided by Python enables programmers to write both clear and logical code for projects. This software does not come pre-packaged with Windows.

How to Install Python on Windows :

There have been several updates in the Python version over the years. The question is how to install Python? It might be confusing for the beginner who is willing to start learning Python but this tutorial will solve your query. The latest or the newest version of Python is version 3.7.4 or in other words, it is Python 3.

Note: The python version 3.7.4 cannot be used on Windows XP or earlier devices.

Before you start with the installation process of Python. First, you need to know about your System Requirements. Based on your system type i.e. operating system and based processor, you must download the python version. My system type is a Windows 64-bit operating system. So the steps below are to install python version

3.7.4 on Windows 7 device or to install Python 3. Download the Python Cheatsheet here. The steps on how to install Python on Windows 10, 8 and 7 are divided into 4 parts to help understand better.

Download the Correct version into the system:

Step 1: Select Python Version

Deciding on a version depends on what you want to do in Python. The two major versions are Python 2 and Python 3. Choosing one over the other might be better depending on your project details. If there are no constraints, choose whichever one you prefer.

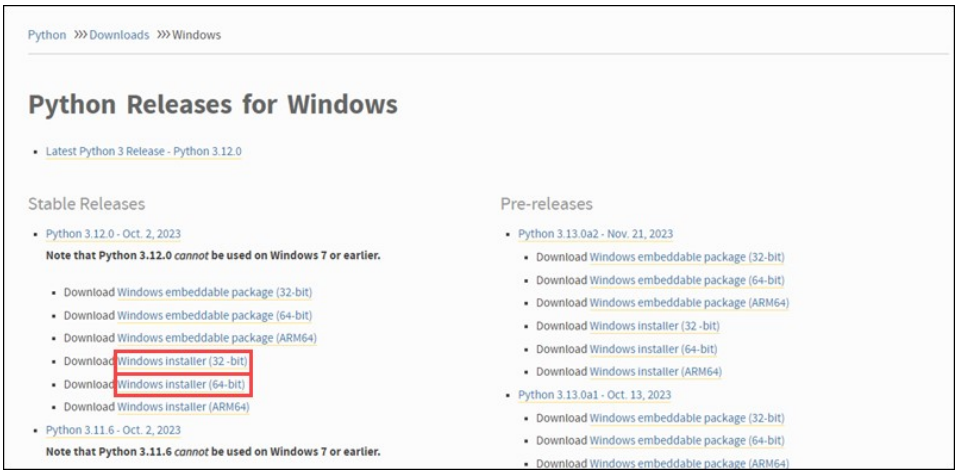
We recommend Python 3, as Python 2 reached its end of life in 2020. Download Python 2 only if you work with legacy scripts and older projects. Also, choose a stable release over the newest since the newest release may have bugs and issues.

Step 2: Download Python Executable Installer

Start by downloading the Python executable installer for Windows:

1. Open a web browser and navigate to the Downloads for Windows section of the official Python website.

2. Locate the desired Python version.



3. Click the link to download the file. Choose either the Windows 32-bit or 64-bit installer.

The download is approximately 25MB.

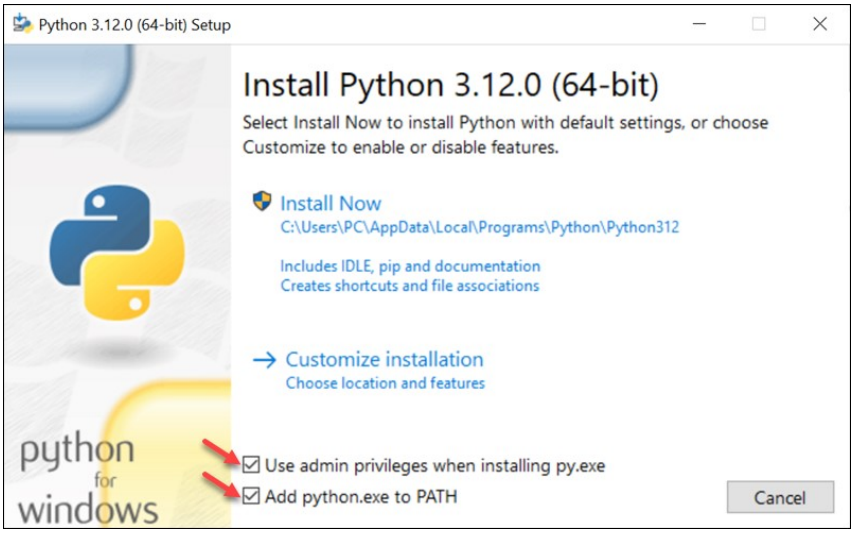
Step 3: Run Executable Installer

The steps below guide you through the installation process:

1. Run the downloaded Python Installer.

2. The installation window shows two checkboxes:

- Admin privileges. The parameter controls whether to install Python for the current or all system users. This option allows you to change the installation folder for Python.
- Add Python to PATH. The second option places the executable in the PATH variable after installation. You can also add Python to the PATH environment variable manually later.



For the most straightforward installation, we recommend ticking both checkboxes.

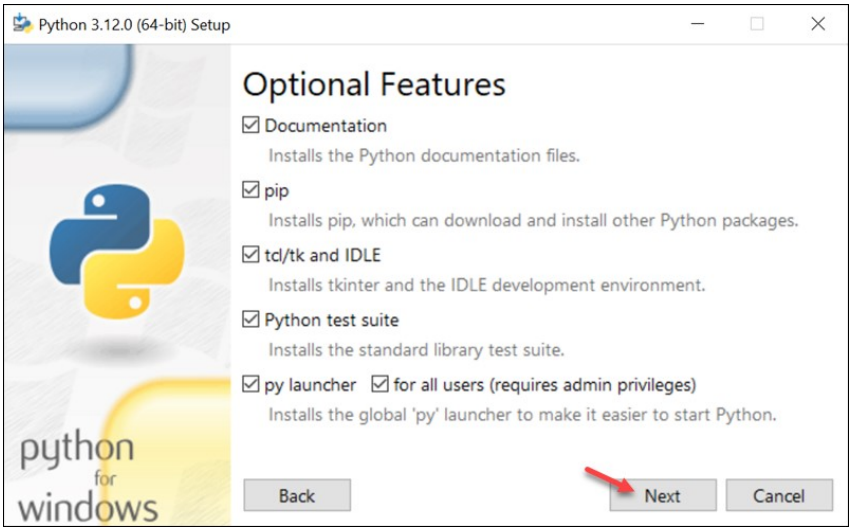
3. Select the Install Now option for the recommended installation (in that case, skip the next two steps).

To adjust the default installation options, choose Customize installation instead and proceed to the following step.

Thedefault installation installs Python to C://Users/[user] /AppData/Local/Programs/Python/ Python[version] for the current user. It includes IDLE (the default Python editor), the PIP package manager, and additional documentation. The installer also creates necessary shortcuts and file associations.

Customizing the installation allows changing these installation options and parameters.

4. Choose the optional installation features. Python works without these features, but adding them improves the program's usability.

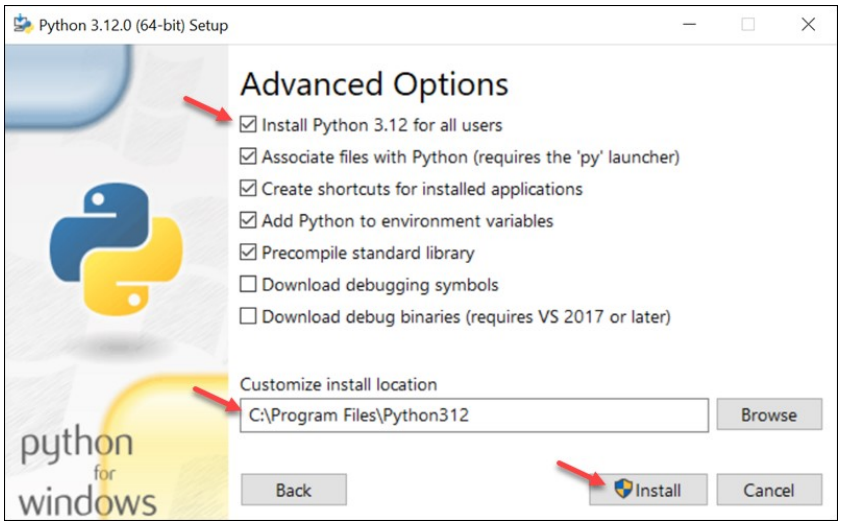


Click Next to proceed to the Advanced Options screen.

5. The second part of customizing the installation includes advanced options.

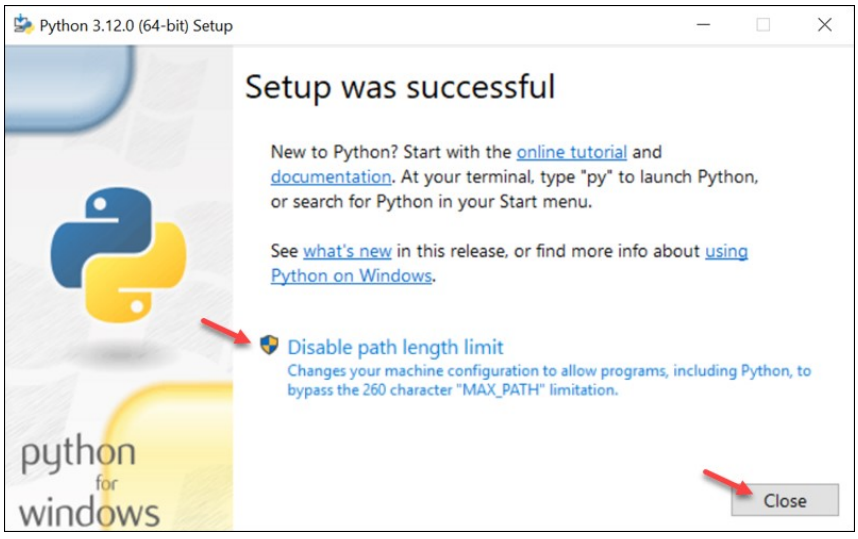
Choose whether to install Python for all users. The option changes the install location to C:¥Program Files¥Python[version]. If selecting the location manually, a common choice is C:¥Python[version] because it avoids spaces in the path, and all users can access it. Due to administrative rights, both paths may cause issues during package installation.

Other advanced options include creating shortcuts, file associations, and adding Python to PATH.



After picking the appropriate options, click Install to start the installation.

6. Select whether to disable the path length limit. Choosing this option will allow Python to bypass the 260-character MAX_PATH limit.



The option will not affect any other system settings, and disabling it resolves potential name-

length issues. We recommend selecting the option and closing the setup.

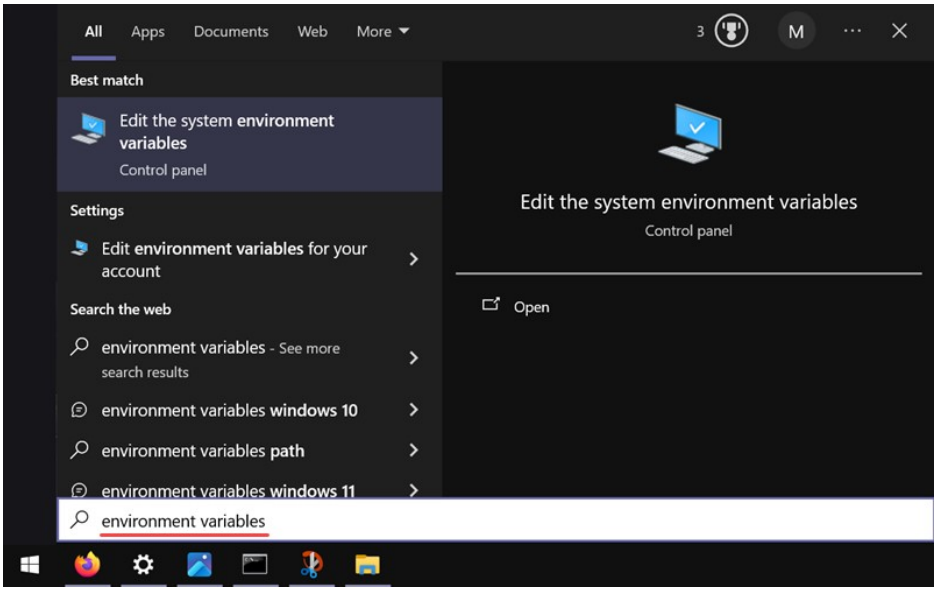
Step 4: Add Python to Path (Optional)

If the Python installer does not include the Add Python to PATH checkbox or you have not selected that option, continue in this step. Otherwise, skip to the next step.

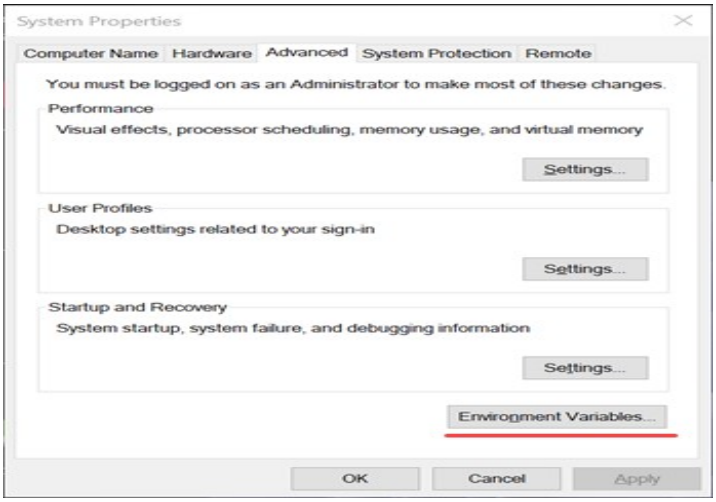
Adding the Python path to the PATH variable alleviates the need to use the full path to access the Python program in the command line. It instructs Windows to review all the folders added to the PATH environment variable and to look for the python.exe program in those folders.

To add Python to PATH, do the following:

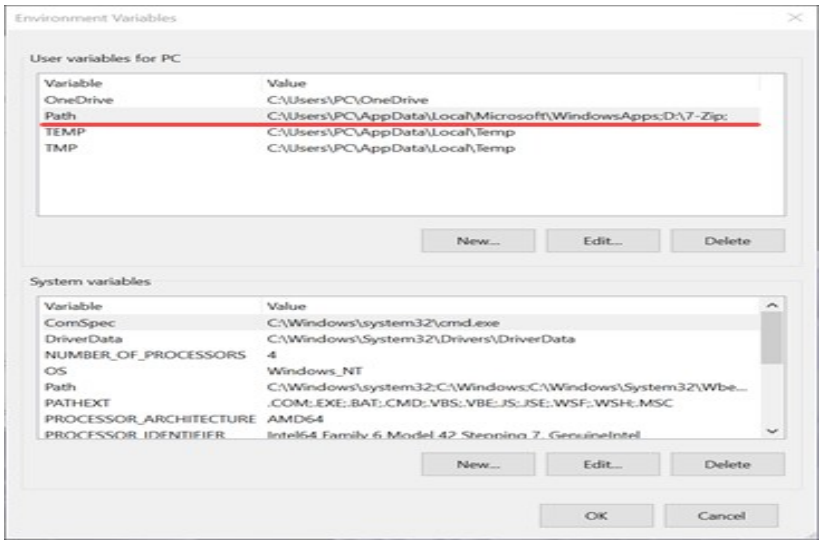
1. In the Start menu, search for Environment Variables and press Enter.



2. Click Environment Variables to open the overview screen.

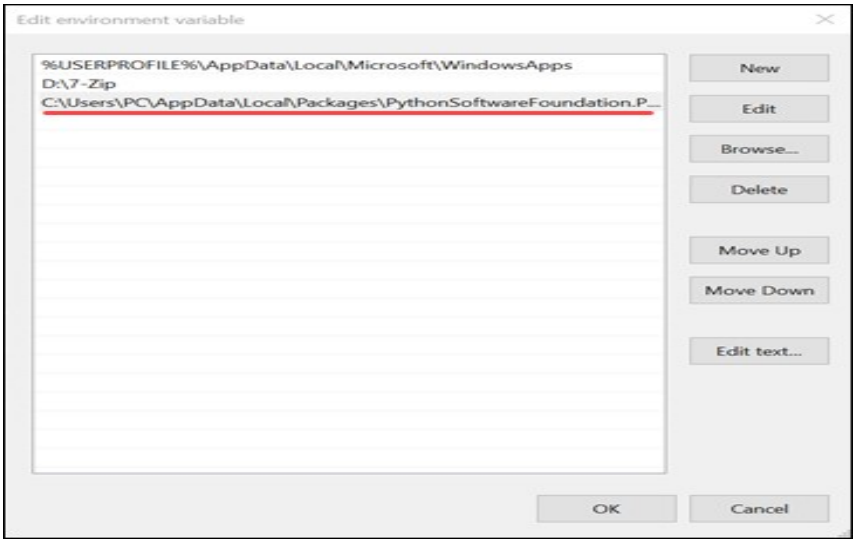


3. Double-click Path on the list to edit it.



Alternatively, select the variable and click the Edit button.

4. Double-click the first empty field and paste the Python installation folder path.



Alternatively, click the New button instead and paste the path.

5. Click OK to save the changes. If the command prompt is open, restart it for the following step.

Step 5: Verify Python Was Installed on Windows

The first way to verify that Python was installed successfully is through the command line. Open the command prompt and run the following command:

python - -version

```
C:\Users\PC>python --version
Python 3.12.0
```

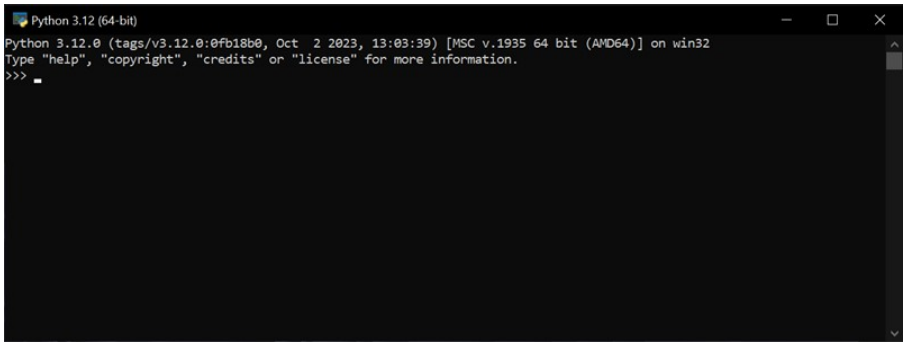
The output shows the installed Python version.

The second way is to use the GUI to verify the Python installation. Follow the steps below to run the Python interpreter or IDLE:

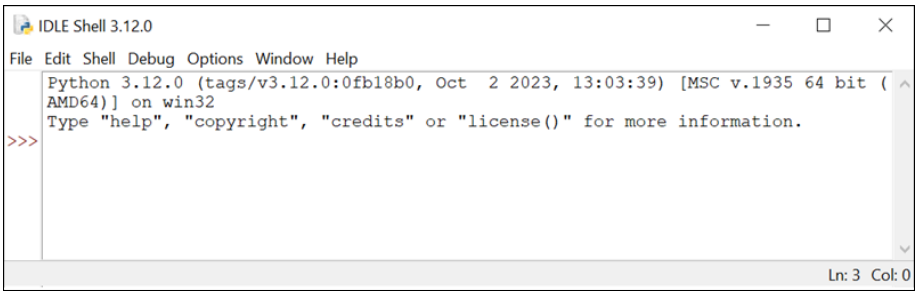
1. Navigate to the directory where Python was installed on the system.

2. Double-click python.exe (the Python interpreter) or IDLE.

3. The interpreter opens the command prompt and shows the following window:



Running IDLE opens Python's built-in IDE:



In both cases, the installed Python version shows on the screen, and the editor is ready for use.

Step 6: Verify PIP Was Installed

To verify whether PIP was installed, enter the following command in the command prompt:

pip –version

If it was installed successfully, you should see the PIP version number, the executable path, and the Python version:

```
C:\Users\PC>pip --version
pip 23.2.1 from C:\Users\PC\AppData\Local\Programs\Python\Python312\Lib\site-packages\pip (python 3.12)
```

PIP has not been installed yet if you get the following output:

'pip' is not recognized as an internal or external command, Operable program or batch file.

If an older version of Python is installed or the PIP installation option is disabled during installation, PIP will not be available. To install PIP, see our article How to Install PIP on Windows.

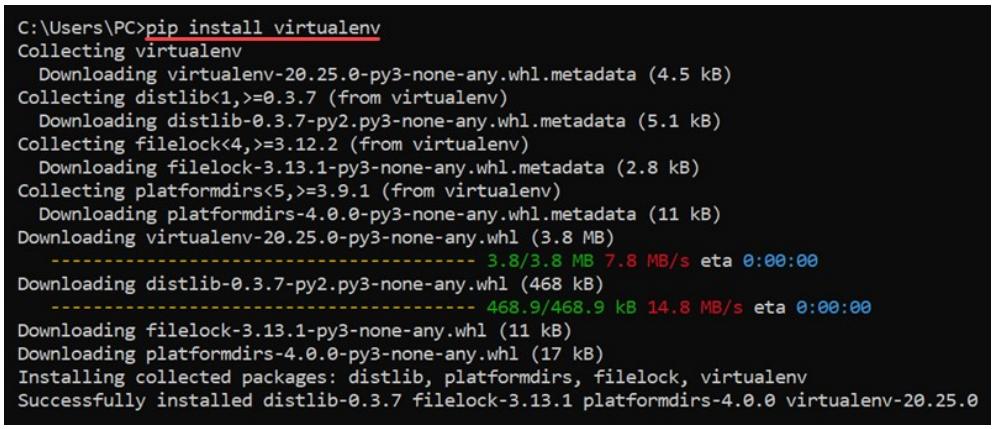
Step 7: Install virtualenv (Optional)

Python software packages install system-wide by default. Consequently, whenever a single project-specific package is changed, it changes for all your Python projects.

The virtualenv package enables making isolated local virtual environments for Python projects. Virtual environments help avoid package conflicts and enable choosing specific package versions per project.

To install virtualenv, run the following command in the command prompt:

pip install virtualenv



Wait for the installation to complete. Once done, it is installed on the system and available for use.

7.1.1.2 Vs Code

Installation

1. Download the Visual Studio Code installer for Windows.
2. Once it is downloaded, run the installer (VSCodeUserSetup-{version}.exe). This will only take a minute.
3. By default, VS Code is installed under C:¥Users¥{Username}¥AppData¥Local¥Programs¥Microsoft VS Code.

Alternatively, you can also download a Zip archive, extract it and run Code from there.

Tip: Setup will add Visual Studio Code to your %PATH%, so from the console you can type 'code .' to open VS Code on that folder. You will need to restart your console after the installation for the change to the %PATH% environmental variable to take effect.

User setup versus system setup

VS Code provides both Windows **user** and **system** level setups.

The user setup does not require Administrator privileges to run as the location will be under your user Local AppData (LOCALAPPDATA) folder. Since it requires no elevation, the user setup is able to provide a smoother background update experience. This is the preferred way to install VS Code on Windows.

Note: When running VS Code as Administrator in a user setup installation, updates will be disabled.

The system setup requires elevation to Administrator privileges to run and will place the installation under the system's Program Files. The in-product update flow will also require elevation, making it less streamlined than the user setup. On the other hand, installing VS Code using the system setup means that it will be available to all users in the system.

See the Download Visual Studio Code page for a complete list of available installation options.

Updates

VS Code ships monthly releases and supports auto-update when a new release is available. If you're prompted by VS Code, accept the newest update and it will be installed (you won't need to do anything else to get the latest bits).

Note: You can disable auto-update if you prefer to update VS Code on your own schedule.

Windows Subsystem for Linux

Windows is a popular operating system and it can be a great cross-platform development environment. This section describes cross-platform features such as the Windows Subsystem for Linux (WSL) and the new Windows Terminal.

Recent Windows build

Make sure you are on a recent Windows 10 build. Check **Settings** > **Windows Update** to see if you are up-to-date.

Windows as a developer machine

With WSL, you can install and run Linux distributions on Windows. This enables you to develop and test your source code on Linux while still working locally on your Windows machine.

When coupled with the WSL extension, you get full VS Code editing and debugging support while running in the context of WSL.

See the Developing in WSL documentation to learn more or try the Working in WSL introductory tutorial.

New Windows Terminal

Available from the Microsoft Store, the Windows Terminal (Preview) lets you easily open PowerShell, Command Prompt, and WSL terminals in a multiple tab shell.

Next steps

Once you have installed VS Code, these topics will help you learn more about VS Code:

- Additional Components - Learn how to install Git, Node.js, TypeScript, and tools like Yeoman.
- User Interface - A quick orientation to VS Code.
- User/Workspace Settings - Learn how to configure VS Code to your preferences through settings.
- Tips and Tricks - Lets you jump right in and learn how to be productive with VS Code.

Common questions

What command-line arguments are supported by the Windows Setup?

VS Code uses Inno Setup to create its setup package for Windows. Thus, all the Inno Setup command-line switches are available for use.

Additionally, you can prevent the Setup from launching VS Code after completion with `/mergetasks=!runcode`.

Scrolling is laggy and not smooth

On certain devices, editor scrolling is not smooth but laggy for an unpleasant experience. If you notice this issue, make sure you install the Windows 10 October 2018 update where this issue is fixed.

I'm having trouble with the installer

Try using the zip file instead of the installer. To use this, unzip VS Code in your `AppData\Local\Programs` folder.

Note: When VS Code is installed via a Zip file, you will need to manually update it for each release.

Icons are missing

I installed Visual Studio Code on my Windows 8 machine. Why are some icons not appearing in the workbench and editor?

VS Code uses SVG icons and we have found instances where the .SVG file extension is associated with something other than `image/svg+xml`. We're considering options to fix it, but for now here's a workaround:

Using the Command Prompt:

1. Open an Administrator Command Prompt.

2. Type REG ADD HKCR¥.svg /f /v "Content Type" /t REG_SZ /d image/svg+xml.

Using the Registry Editor (regedit):

1. Start regedit.
2. Open the HKEY_CLASSES_ROOT key.
3. Find the .svg key.
4. Set its Content Type Data value to image/svg+xml.
5. Exit regedit.

Unable to run as admin when AppLocker is enabled

With the introduction of process sandboxing (discussed in this blog post) running as administrator is currently unsupported when AppLocker is configured due to a limitation of the runtime sandbox. If your work requires that you run VS Code from an elevated terminal, you can launch code with --no-sandbox --disable-gpu-sandbox as a workaround.

Subscribe to issue #122951 to receive updates.

Working with UNC paths

Beginning with version 1.78.1, VS Code on Windows will only allow to access UNC paths (these begin with a leading \\) that were either approved by the user on startup or where the host name is configured to be allowed via the new security.allowedUNCHosts setting.

If you rely on using UNC paths in VS Code, you can either

- configure the host to be allowed via the security.allowedUNCHosts setting (for example add server-a when you open a path such as ¥¥server-a¥path)
- map the UNC path as network drive and use the drive letter instead of the UNC path (documentation)
- define a global environment variable NODE_UNC_HOST_ALLOWLIST with the backslash-separated list of hostnames to allow, for example: server-a¥server-b to allow the hosts server-a and server-b.

Note: if you are using any of the remote extensions to connect to a workspace remotely (such as SSH), the security.allowedUNCHosts has to be configured on the remote machine and not the local machine.

This change was done to improve the security when using VS Code with UNC paths. Please refer to the associated security advisory for more information.

7.1.2 Introduction to Flask frame work

Flask, a lightweight and versatile web framework, has emerged as a popular choice among developers for building web applications due to its simplicity, flexibility, and scalability.

Developed in Python and inspired by the Sinatra framework for Ruby, Flask offers an elegant and minimalist approach to web development, making it ideal for both beginners and experienced developers alike.

At its core, Flask provides a simple yet powerful foundation for building web applications, offering essential features such as URL routing, request handling, and response generation. Unlike monolithic frameworks, Flask follows a microframework philosophy, allowing developers to selectively choose and integrate components based on their specific project requirements, resulting in lean and efficient applications.

One of the key strengths of Flask is its extensibility, which is facilitated by its modular design and extensive ecosystem of third-party extensions. These extensions cover a wide range of functionalities, including database integration, authentication, caching, and more, allowing developers to easily enhance and customize their applications without reinventing the wheel.

Flask embraces the principles of simplicity and minimalism, providing developers with a clean and intuitive API that promotes rapid development and prototyping. Its unopinionated nature gives developers the freedom to structure their applications as they see fit, without imposing unnecessary constraints or conventions.

Despite its lightweight footprint, Flask is remarkably powerful and capable of handling a variety of web development tasks, from simple RESTful APIs to full-fledged web applications with complex business logic. Its modular architecture and built-in development server make it easy to get started with Flask, allowing developers to focus on writing clean and maintainable code.

Flask also excels in terms of performance and scalability, thanks to its efficient request handling mechanism and support for asynchronous programming with libraries like Flask-SocketIO. Whether deploying applications on a single server or in a distributed environment, Flask provides the flexibility and performance needed to meet the demands of modern web applications.

In summary, Flask represents a powerful and versatile framework for building web applications in Python. Its minimalist design, extensibility, and performance make it an attractive choice for developers seeking a lightweight yet capable solution for their web development projects. Whether you're building a simple API or a complex web application, Flask provides the tools and flexibility you need to bring your ideas to life.

7.1.3 Python Libraries

The provided code utilizes the Flask web framework, a lightweight and flexible Python framework for building web applications. Flask is known for its simplicity and minimalistic design, making it easy to get started with web development projects. Here's an elaboration of the Python frameworks used in the code:

1. Flask

Flask is the primary web framework used in the code. It's responsible for handling HTTP requests and responses, routing URLs to appropriate functions, rendering HTML templates, and managing application logic. Flask follows the WSGI (Web Server Gateway Interface) specification and provides a micro-framework approach, allowing developers to add only the components they need.

2. TensorFlow

While not a web framework, TensorFlow is a critical Python library for developing deep learning models. In the code, TensorFlow is used to load a pre-trained deep learning model for image classification. It provides tools and utilities for building, training, and deploying machine learning and deep learning models efficiently.

3. Matplotlib

Matplotlib is a popular Python plotting library used for creating static, animated, and interactive visualizations. In the code, matplotlib is used to generate bar charts and line plots to visualize fish demand information based on the user's input.

4. PIL (Python Imaging Library)

PIL, now known as Pillow, is a library for opening, manipulating, and saving many different image file formats. In the code, PIL is used to process and manipulate uploaded images before feeding them into the deep learning model for classification.

5. Numpy

NumPy is a fundamental package for scientific computing with Python, providing support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays efficiently. In the code, numpy is used for various numerical operations and data manipulation tasks, such as array manipulation and mathematical computations.

6. Pandas

Pandas is a powerful data analysis and manipulation library for Python, offering data structures and operations for manipulating numerical tables and time series data. In the code, pandas is used to load and manipulate CSV files containing fish demand data for analysis and visualization.

7. Seaborn

Seaborn is a statistical data visualization library based on matplotlib, providing a high-level interface for drawing attractive and informative statistical graphics. While not explicitly used in the provided code, seaborn is often used alongside matplotlib for creating more visually appealing plots and statistical visualizations.

These Python frameworks and libraries work together to create a web application that allows users to upload images for fish species classification, visualize fish demand information, and receive recommendations based on their input. Each framework or library serves a specific purpose, enabling the development of different components within the application.

7.2 Source Code

```
from flask import Flask,render_template,flash,redirect,request,send_from_directory,url_for, send_file
 import mysql.connector, os
 from PIL import Image
 import matplotlib.pyplot as plt
 import numpy as np
 from tensorflow.keras.models import load_model
 from tensorflow.keras.preprocessing import image
 from scipy.sparse import csr_matrix
 from scipy.sparse.linalg import svds
 import numpy as np
 import pandas as pd
 import matplotlib.pyplot as plt
 from matplotlib.backends.backend_agg import FigureCanvasAgg as FigureCanvas
 import io
 import base64
 import seaborn as sns
 app = Flask(__name__)
 mydb = mysql.connector.connect(
    host="localhost",
    user="root",
    password="",
    port="3306",
    database='fish'
 )
 mycursor = mydb.cursor()
def executionquery(query,values):
    mycursor.execute(query,values)
    mydb.commit()
    return
```

```
def retrivequery1(query,values):
    mycursor.execute(query,values)
    data = mycursor.fetchall()
    return data


def retrivequery2(query):
    mycursor.execute(query)
    data = mycursor.fetchall()
    return data


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


@app.route(‘/register’, methods=[“GET”, “POST”])

def register():
  if request.method == “POST”:

      email = request.form[‘email’]

      password = request.form[‘password’]

      c_password = request.form[‘c_password’]

      if password == c_password:

          query = “SELECT UPPER(email) FROM users”

          email_data = retrivequery2(query)

          email_data_list = []

          for i in email_data:

              email_data_list.append(i[0]
```

```
query = "INSERT INTO users (email, password) VALUES (%s, %s)"
                values = (email, password)
                executionquery(query, values)
                return render_template('index.html', message="Successfully Registered! Please go to login section")
            return render_template('index.html', message="This email ID is already exists!")
        return render_template('index.html', message="Conform password is not match!")
    return render_template('index.html'
@app.route('/login', methods=["GET", "POST"])
def login():
    if request.method == "POST":
        email = request.form['email']
        password = request.form['password']

        query = "SELECT UPPER(email) FROM users"
        email_data = retrivequery2(query)
        email_data_list = []
        for i in email_data:
            email_data_list.append(i[0])

        if email.upper() in email_data_list:
            query = "SELECT UPPER(password) FROM users WHERE email = %s"
            values = (email,)
            password__data = retrivequery1(query, values)
            if password.upper() == password__data[0][0]:
                global user_email
                user_email = email
```

```
    return redirect("/home")
    return render_template('index.html', message= "Invalid Password!!")
    return render_template('index.html', message= "This email ID does not exist!")
    return render_template('index.html')
@app.route('/home')
def home():
    return render_template('home.html')


@app.route('/about')
def about():
    return render_template('about.html')
'''
def fish_demand_info(fish_name, df):
    fish_data = df[df['Fish Type'] == fish_name]
    if fish_data.empty:
        return f"No information available for {fish_name}"

    avg_consumption = fish_data['Average Consumption (tons)'].mean()
    preferred_size = fish_data['Preferred Size (cm)'].mean()
    price_low = fish_data['Price Range (INR per kg)'].apply(lambda x: float(x.split(' -
')[0])).mean()
    price_high = fish_data['Price Range (INR per kg)'].apply(lambda x: float(x.split(' -
')[1])).mean()
    seasonal_availability = fish_data['Seasonal Availability'].mode().iloc[0]
    info = f"Demand info for {fish_name}:¥n"
    info += f"Average consumption: {avg_consumption:.2f} tons¥n"
    info += f"Preferred size: {preferred_size:.2f} cm¥n"
    info += f"Price range: INR {price_low:.2f} - {price_high:.2f} per kg¥n"
    info += f"Seasonal availability: {seasonal_availability}"
```

```
    plt.figure(figsize=(10, 6))
    plt.bar(['Average Consumption', 'Preferred Size', 'Price Range Low', 'Price Range High'],
            [avg_consumption, preferred_size, price_low, price_high],
            color=['blue', 'green', 'orange', 'red'])
    plt.title(f'Demand Info for {fish_name}')
    plt.ylabel('Values')
    plt.xlabel('Statistics')
    img_buffer = io.BytesIO()
    plt.savefig(img_buffer, format='png')
    img_buffer.seek(0)
    img_str = base64.b64encode(img_buffer.read()).decode('utf-8')
    # img_tag = f'<img src="{img_str}"/>'
    plt.close()
    return img_str   '''
import matplotlib.pyplot as plt
def plot_fish_demand(fish_name, df):
    fish_data = df[df["Fish Type"] == fish_name]
    purchases = fish_data["Purchases"].apply(lambda x: int(x.split()[0]))
    plt.figure(figsize=(10, 6))
    plt.plot(purchases.reset_index(drop=True), marker='o', linestyle='-', color='blue')
    plt.title(f'Demand for {fish_name}', fontsize=15)
    plt.xlabel('Data Point', fontsize=12)
    plt.ylabel('Purchases (Units)', fontsize=12)
    plt.grid(True)
    img_buffer = io.BytesIO()
    plt.savefig(img_buffer, format='png')
    img_buffer.seek(0)
    img_str = base64.b64encode(img_buffer.read()).decode('utf-8')
    img_tag = f'<img src="{img_str}"/>'
    plt.close()
    return img_tag
import numpy as np  # Add this import for numerical operations
```

```
def plot_fish_demand_detailed(fish_name, df):
    fish_data = df[df["Fish Type"] == fish_name]
    seasons = fish_data["Seasonal Data"].unique()
    n_seasons = len(seasons)
    # Assuming 'Purchases' column values are strings and need to be converted to integers
    fish_data["Purchases"] = fish_data["Purchases"].apply(lambda x: int(x.split()[0]))
    plt.figure(figsize=(12, 8))
    # Width of each bar and the spacing between groups of bars
    bar_width = 0.8 / n_seasons  # Adjust this value as needed
    index = np.arange(len(fish_data) / n_seasons)  # Create a set of index values for the x-axi
    for i, season in enumerate(seasons):
        season_data = fish_data[fish_data["Seasonal Data"] == season]
        # Create an offset for each season to position bars side by side
        offset = (i - n_seasons / 2) * bar_width + bar_width / 2
        plt.bar(index + offset, season_data["Purchases"], bar_width, label=season)
        plt.title(f'Detailed Demand Analysis for {fish_name}', fontsize=15)
    plt.xlabel('Data Point', fontsize=12)
    plt.ylabel('Purchases (Units)', fontsize=12)
    plt.xticks(index, rotation=45)  # Adjust the rotation if labels overlap
    plt.legend(title="Season")
    plt.grid(True, which='both', axis='y', linestyle='--', linewidth=0.5)
    img_buffer = io.BytesIO()
    plt.savefig(img_buffer, format='png', bbox_inches='tight')
    img_buffer.seek(0)
    img_str = base64.b64encode(img_buffer.read()).decode('utf-8')
    img_tag = f'<img src="{img_str}"/>'
    plt.close()
    return img_tag
    def plot_fish_demand_detailed(fish_name, df):
    fish_data = df[df["Fish Type"] == fish_name]
    purchases = fish_data["Purchases"].apply(lambda x: int(x.split()[0]))
    seasons = fish_data["Seasonal Data"]
```

```
    plt.figure(figsize=(12, 8))
    for season in seasons.unique():
        season_data = fish_data[fish_data["Seasonal Data"] == season]
        season_purchases = season_data["Purchases"].apply(lambda x: int(x.split()[0]))
        plt.plot(season_purchases.reset_index(drop=True), marker='o', linestyle='-', label=season)
    plt.title(f'Detailed Demand Analysis for {fish_name}', fontsize=15)
    plt.xlabel('Data Point', fontsize=12)
    plt.ylabel('Purchases (Units)', fontsize=12)
    plt.legend(title="Season")
    plt.grid(True)
    img_buffer = io.BytesIO()
    plt.savefig(img_buffer, format='png')
    img_buffer.seek(0)
    img_str = base64.b64encode(img_buffer.read()).decode('utf-8')
    img_tag = f'<img src="{img_str}"/>'
    plt.close()
    return img_tag
@app.route('/classification', methods=["GET", "POST"])
 def classification():
    if request.method == "POST":
        myfile=request.files['file']
        fn=myfile.filename
        mypath=os.path.join('static/img/', fn)
        myfile.save(mypath)
        accepted_formated=['jpg','png','jpeg','jfif','JPG']
        if fn.split('.')[-1] not in accepted_formated:
            flash("Image formats only Accepted","Danger")
            return render_template("upload.html")
        classes=["Black Sea Sprat","Gilt-Head Bream","Hourse Mackerel","Red Mullet","Red Sea
 Bream","Sea Bass","Shrimp","Striped Red Mullet","Trout"]
        new_model = load_model("resnet_1.h5")
        test_image = image.load_img(mypath, target_size=(224, 224))
```

```
        test_image = image.img_to_array(test_image)
        test_image = test_image/255
        test_image = np.expand_dims(test_image, axis=0)
        result = new_model.predict(test_image)
        print(result)
        print(np.argmax(result))
        fish_name=classes[np.argmax(result)]
        df=pd.read_csv("fish_analysis.csv")
        chart_html1 = plot_fish_demand_detailed(fish_name, df)
        chart_html2 = plot_fish_demand(fish_name, df)
        return render_template('classification.html', fish = fish_name, path = mypath, chart_html1
 = chart_html1, chart_html2 = chart_html2)
    return render_template('classification.html')
 @app.route("/chart", methods=["POST"])
 def chart():
    if request.method == "POST":
        fish_name = request.form['fish']
        chart_html1 = request.form['chart_html1']
        chart_html2 = request.form['chart_html2']


        print(11, fish_name)
        print(11, chart_html1)
        print(11, chart_html2)


        return render_template('result.html', fish = fish_name, chart_html1 = chart_html1,
 chart_html2 = chart_html2)


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

def recommentation():
   if request.method == "POST":
      def fish_recommendations_for_state(state, df):
         user_item_matrix = df.pivot_table(index='State', columns='Fish Type',
values='Average Consumption (tons)', fill_value=0)
```

```
        sparse_matrix = csr_matrix(user_item_matrix.values)

        k = min(user_item_matrix.shape) – 1  # Number of latent factors (choose one less than the smaller dimension)

        u, sigma, vt = svds(sparse_matrix, k=k)

        predicted_consumption = np.dot(np.dot(U, np.diag(sigma)), Vt)

        predicted_df = pd.DataFrame(predicted_consumption, columns=user_item_matrix.columns, index=user_item_matrix.index)

        state_consumption = predicted_df.loc[state]

        recommended_fish = state_consumption.sort_values(ascending=False).head(3)

        return recommended_fish.index.tolist()
      df=pd.read_csv("fish_demand_dataset_india_2000.csv")
      state = request.form['state']
      result = fish_recommendations_for_state(state, df)
      print(1111111111, result)
      print(2222222222, list(result))
      return render_template('recommentation.html', result=result)
    return render_template('recommentation.html'


if __name__ == '__main__':
    app.run(debug=True, threaded=False)
```

7.2.2 Performance

Assessing the performance of the provided project involves evaluating various aspects such as functionality, efficiency, user experience, scalability, and potential areas for improvement. Here's a breakdown of the project's performance:

1. **Functionality**: The project appears to meet its primary objectives, which include allowing users to register, login, upload images for fish species classification, visualize fish demand information, and receive recommendations based on their input.

2. Efficiency: The efficiency of the project depends on factors such as response time, resource utilization, and computational overhead. Since the project uses Flask, a lightweight web framework, it should be efficient in handling HTTP requests and responses. However, the efficiency of the deep learning model inference process and data visualization operations may vary based on factors such as model complexity, dataset size, and computational resources.

3. User Experience: User experience encompasses aspects such as interface design, ease of use, responsiveness, and error handling. The project's user interface could be further enhanced with better styling, layout, and interactive features to improve user engagement. Additionally, error handling mechanisms should be robust to handle various user inputs and edge cases gracefully.

4. Scalability: The project's scalability refers to its ability to handle increasing user traffic, data volume, and feature complexity. Flask is suitable for building scalable web applications, but the performance may degrade under heavy loads or when processing large datasets. To improve scalability, optimizations such as caching, load balancing, and asynchronous processing can be implemented.

5. Model Accuracy: The accuracy of the deep learning model for fish species classification directly impacts the project's performance. The provided code uses a pre-trained ResNet model, which may achieve reasonable accuracy depending on the quality of the training data and model architecture. Continuous monitoring and retraining of the model with updated data can improve classification accuracy over time.

6. Data Visualization: The project includes data visualization features using matplotlib and seaborn for displaying fish demand information. The performance of data visualization depends on factors such as rendering speed, interactivity, and visual clarity. Optimizations such as reducing the complexity of plots, caching rendered images, and implementing client-side rendering techniques can enhance visualization performance.

7. Database Performance: The project interacts with a MySQL database for user authentication and data retrieval. The performance of database operations can impact overall application responsiveness. Optimizations such as indexing, query optimization, connection pooling, and database sharding can improve database performance and scalability.

Overall, the performance of the provided project appears to be satisfactory in terms of functionality and basic usability.

Chapter 8

TESTING

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

8.1 Testing Levels

8.1.1 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.

8.1.2 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 outcome 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.

8.1.3 Functional Testing

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 puts must be exercised.Systems/Procedures: interfacing systems or procedures must be woked Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identifying 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 de effective value of current tests is determined.

8.1.4 System Testing

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 45 flows, emphasizing pre-driven process links and integration points.

8.1.5 PyTest

Pytest is a Python testing framework that originated from the PyPy project. It can be used to write various types of software tests, including unit tests, integration tests, end-to-end tests, and functional tests. Its features include parametrized testing, fixtures, and assert re-writing. Pytest fixtures provide the contexts for tests by passing in parameter names in test cases; its parametrization eliminates duplicate code for testing multiple sets of input and output; and its rewritten assert statements provide detailed output for causes of failures. Pytest was developed as part of an effort by third-party packages to address Python's built-in module unittest's shortcomings. It originated as part of PyPy, an alternative implementation of Python to the standard CPython. Since its creation in early 2003, PyPy has had a heavy emphasis on testing. PyPy had unit tests for newly written code, regression tests for bugs, and integration tests using CPython's test suite.In mid 2004, a testing framework called utest emerged and contributors to PyPy began convertingexisting test cases to utest. Meanwhile, at EuroPython 2004 a complementary standard library for testing, named std, was invented. This package laid out the principles, such as assert rewriting, of what would later become pytest. In late 2004, the std project was renamed to py, std.utest became py.test, and the py library was separated from PyPy. In November 2010, pytest 2.0.0 was released as a package separate from py. It was still called py.test until August 2016, but following pytest 3.0.0 the recommended command line entry point became pytest.Pytest has been classified by developer security platform Snyk as one of the key

ecosystem projects in Python due to its popularity. Some well-known projects who switched to pytest from unittest and nose (another testing package) include those of Mozilla and Dropbox.

Fig.8.1.5 Py Test

8.2 Testing Methods

Testing is of two types. They are:

a) Static Testing

b) Dynamic Testing

8.2.1 Static Testing

Static Testing is a technique for assessing structural characteristics of code. It examines the structure of code, but code is not executed, it does not involve actual execution. Static Testing is of three types. They are:

a) Inspection

b) Walkthroughs

c) Technical Reviews

8.2.1.1 Inspect

An inspection is one of the most common sorts of review practices found in software projects. The goal of the inspection is to identify defects. Commonly inspected work products include software requirements specifications and test plans.

8.2.1.2 Walkthrough

Walkthrough in software testing is used to review documents with peers, managers, and fellow team members who are guided by the author of the document to gather feedback and ach a consensus. A walkthrough can be pre- planned or organized based on the needs.

8.2.1.3 Technical Reviews

A software technical review is a form of peer review in which "a team of qualified personnel examines the suitability of the software product for its intended use and identifies discrepancies from specifications and standards. Technical reviews may also provide recommendations of alternatives and examination of various alternatives.

8.2.2 Dynamic Testing

Dynamic testing is a term used in software engineering to describe the testing of the dynamic behaviour of code. That is, dynamic analysis refers to the examination of the physical response from the system to variables that are not constant and change with time. In dynamic testing the software must actually be compiled and run. It involves working with the software, giving imput values and checking if the output is as expected by executing specific test cases which can be done manually or with the use of an automated process.

Dynamic Testing is of two types. They are

a) Black Box Testing

b) White Box Testing

8.2.2.1 Black Box Testing

Black Box Testing is also known as Behavioural Testing, is a software testing method in which the internal structure/design/implementation of the item being tested is not known to the tester. These tests can be functional or non-functional, though usually functional.

Black Box Testing is of four types. They are:

a) Boundary Value Analysis

b) Equivalence Partitioning

c) Cause Effect Graph based Testing

d) Error Guessing Boundary Value Analysis

Boundary Value Analysis is applicable when the module to be tested is a function of several independent variables. This method becomes important for physical quantities where boundary checking conditions are crucial. Equivalence Partitioning: Equivalence partitioning or equivalence class partitioning is a software testing technique that divides the input data of a software unit into partitions of equivalent data from which test cases can be derived. In principle, test cases are designed to cover each partition at least once. This technique tries to define test

cases that uncover classes of errors, thereby reducing the total number of test cases that must be develop. Cause Effect Graph Based Testing: Cause effect graphing techniques helps in sting combination of input conditions in a systematic way such that number of test cases does not become unmanageably large. The following process is used to derive the test cases.

i. Division of Specification:

The specification is divided into workable pieces as cause effect graphing becomes complex she used on large specifications.

ii. Identification of Causes and Effects: In this step, we will identify causes and effects in the specifications. A cause is a distinct input condition identified in the problem. Similar effect is the output condition.

iii. Transformation of specific effect into a cause effect graph: Based on the analysis of the specification it is transformed into a Boolean graph linking the cases and effects. This is the cause effect graph is.

iv. Conversion into Decision Table: The cause effect graph obtained is converted into a limited entry decision table by verifying state conditions in the graph. Each column in the table indicates a test graph.

Error Guessing: It is the preferred method used when all the other methods fail. Sometimes, it is used to test some gecial cases. According to this method, errors or bugs can be guessed which do not fit in any of the earlier defined situations.

a) Divide by zero

b) What if all inputs are negative?

c) What if the input list is empty?

The tester does not have to use any particular testing technique.

8.2.2.2 White Box Testing

White-box testing is a method of testing software that tests internal structures or workings of an application, as opposed to its functionality. In white-box testing an internal perspective of the system, as well as programming skills, are used to design test cases.

White box testing is of four types. They are:

a) Logic Coverage Critera

b) Basis Path Testing

c) Data Flow Testing

d) Mutation Testing

Logic Coverage Criteria:

Decision coverage is a stronger logic coverage criterion Based on this criterion, a fair number of test cases should be written, for each decision to have true and false value at least one time. Put it differently, each branch changing direction must be executed at least once.

Basis Path Testing: Basis path testing is the oldest structural testing technique. This testing is based on the control structure of the program, Based on the control structure a flow graph is prepared and all possible paths can be covered and tested during testing Path coverage is useful for detecting more errors but the problem with this technique is that programs that contain loops may have an infinite number of possible paths and it is not practical to test all the paths. So, some selected paths are executed for the maximum coverage of logic.

Data Flow Testing: Data flow testing is a family of test strategies based on selecting paths through the program's control flow in order to explore sequences of events related to the status of variables or data objects Data Flow Testing focuses on the points at which variables receive values and the points at which these values are used.

Mutation Testing: Mutation Testing is a type of Software testing where we mutate certain statements in the source code and check if the test cases are able to find the errors. The changes in the mutant program are kept extremely small, so it does not affect the overall objective of the program.

The goal of Mutation Testing is to assess the quality of the test cases which should be robust enough to fail mutant code. This method is also called a Fault-based testing strategy as it involves creating a fault in the program, In this project, we perform the manual testing which checks whether the given inputs get the appropriate outputs or not. This comes under the Black Box testing. We take an input from the user and an output is displayed.

Chapter 9

OUTPUT SCREENS

9.1 HOME PAGE:

This is the home page which will display the main page of the project website.

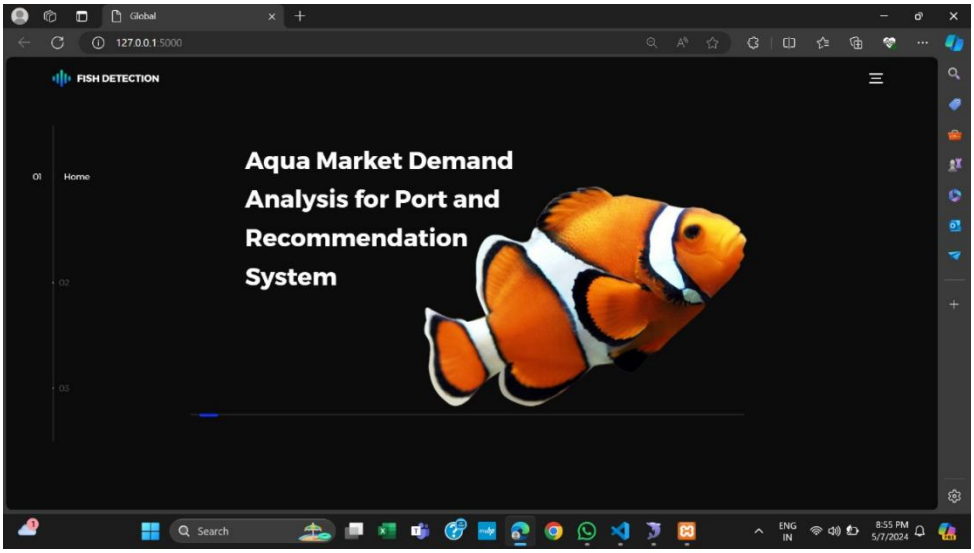


Figure 9.1(a) Home Screen of System

This screen describes about the different web pages that are involved in the project.

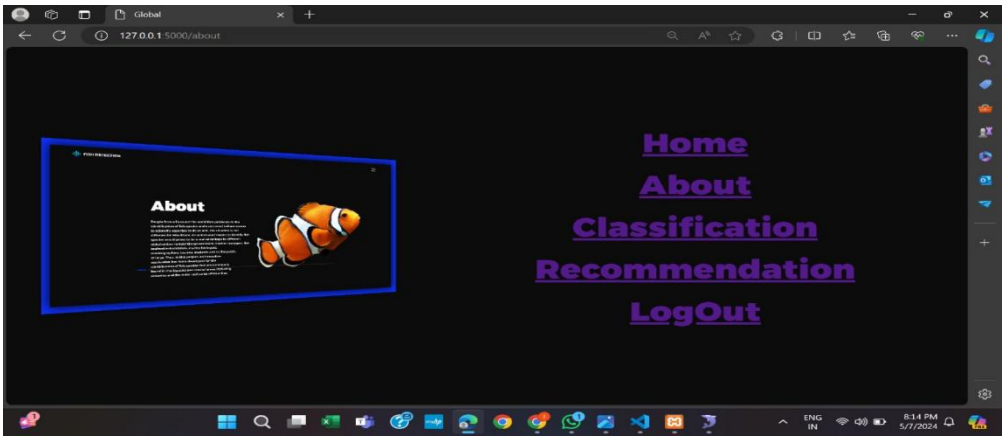


Figure 9.1 (b) Home Screen of System

9.2 ABOUT PAGE:

This is the About page which is used to brief about the Port Aqua Market Demand Analysis and Recommendation System Project.

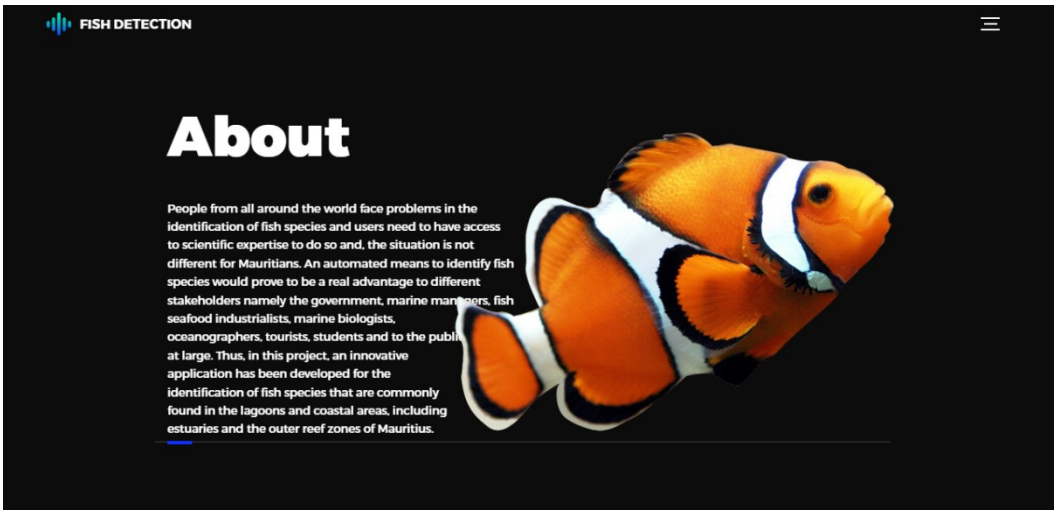


Figure 9.2 about page

9.3 REGISTRATION AND LOGIN PAGE:

To access the website, the user must create an account by filling the below Registration Form.

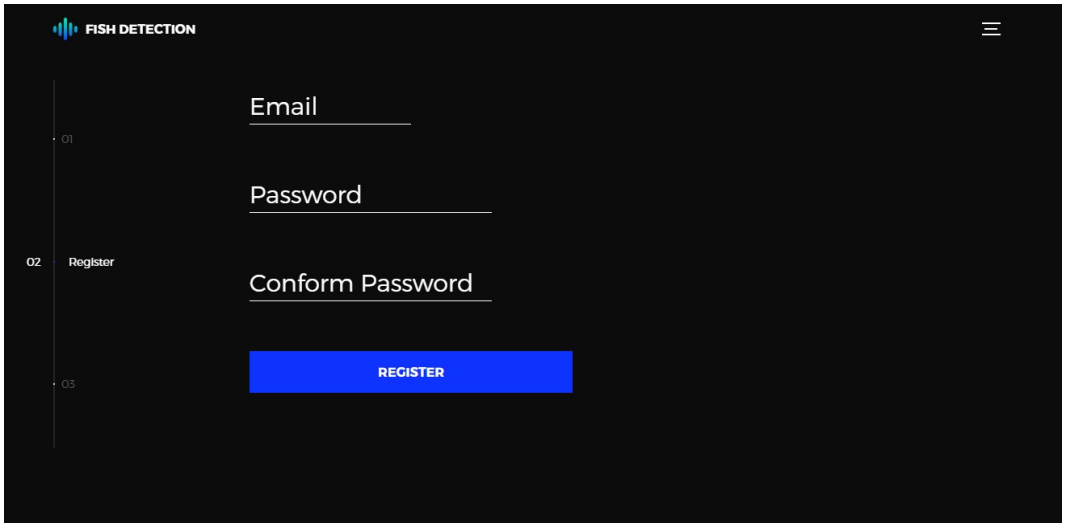


Figure 9.3 (a) registration page

The below create is used to sign in to the website. It is only possible, when the registered to the website.

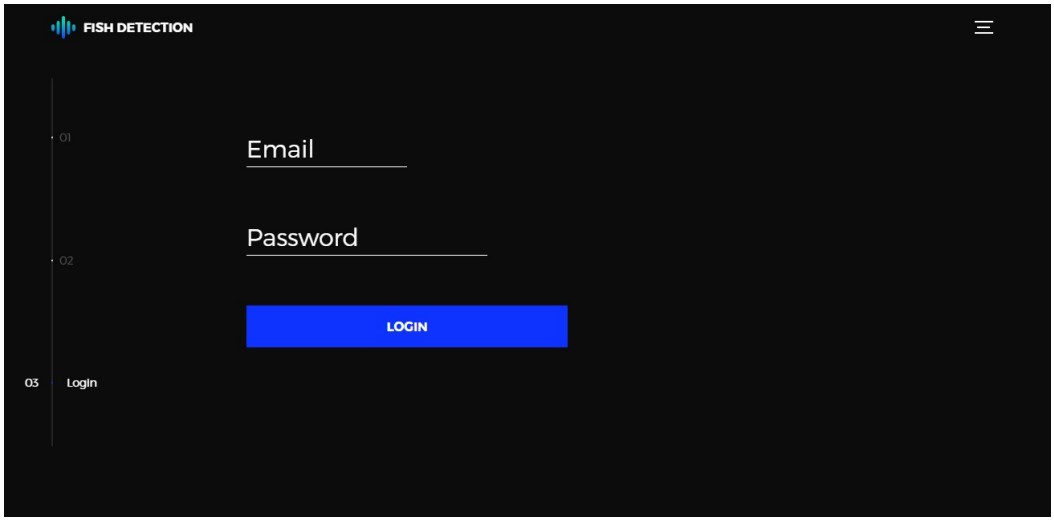


Figure 9.3 (b) login page

9.4 UPLOAD PAGE:

This page is for fish classification which display the inputted fish image with some information about the given fish.

Figure 9.4 upload page

9.5 Fish Classification Result:

This is the resultant page which will be displayed after inputted fish submission.

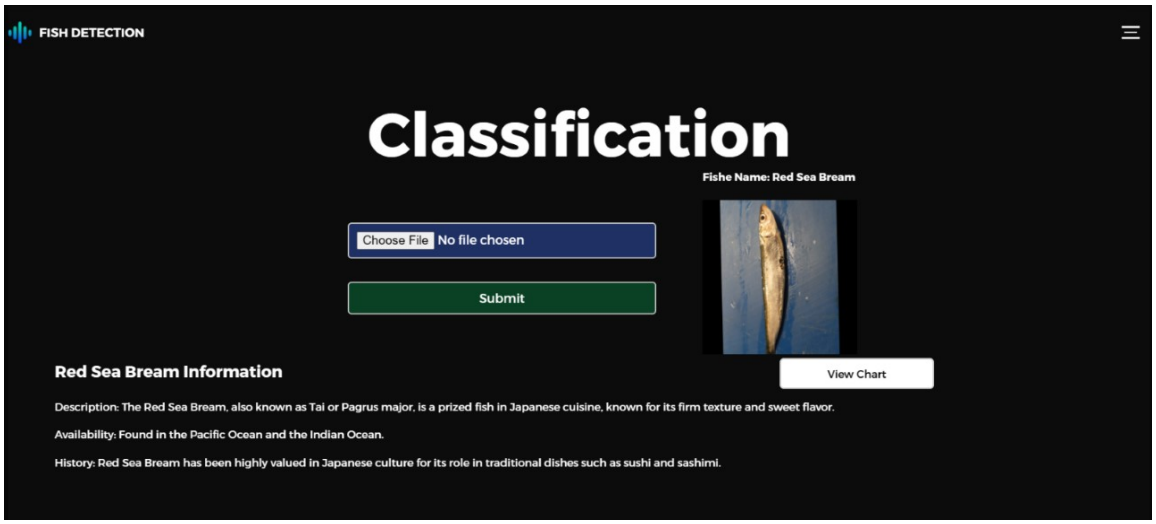


Figure 9.5 fish classification result

9.6 Demand Analysis Result:

1. Fish demand analysis based on seasons. Here, it shows the demand based on season of inputted fish. It will be appear, when the user clicked on the “View Chart” button which is displayed in the classification resultant page.

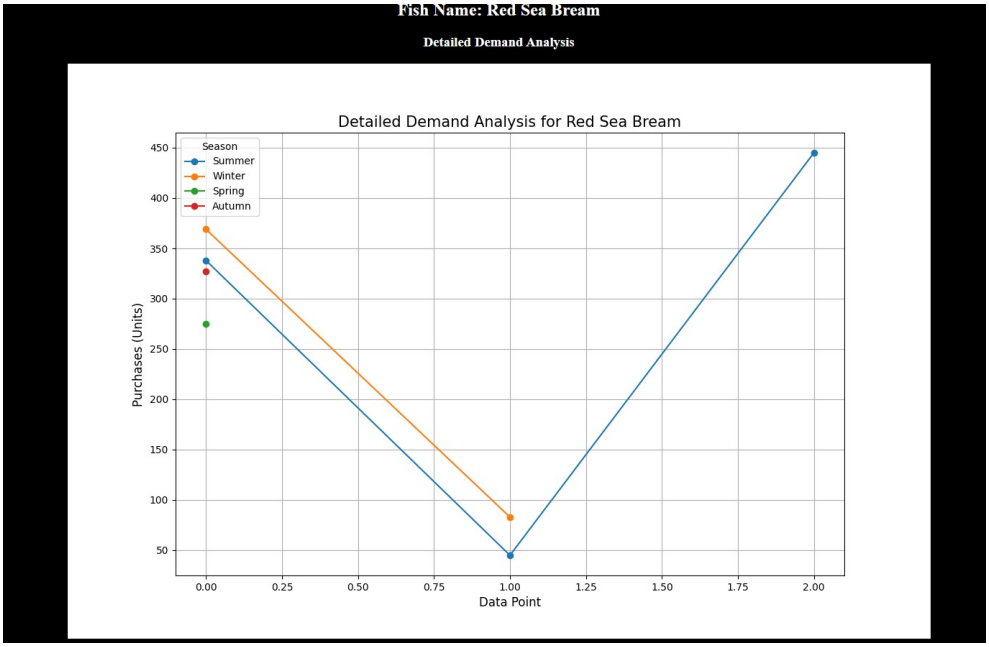


Figure 9.6 (a) demand analysis result

2. Fish demand analysis based on purchases. Here, it shows the demand based on purchases of inputted fish. It will be appear below the season based demand graph.

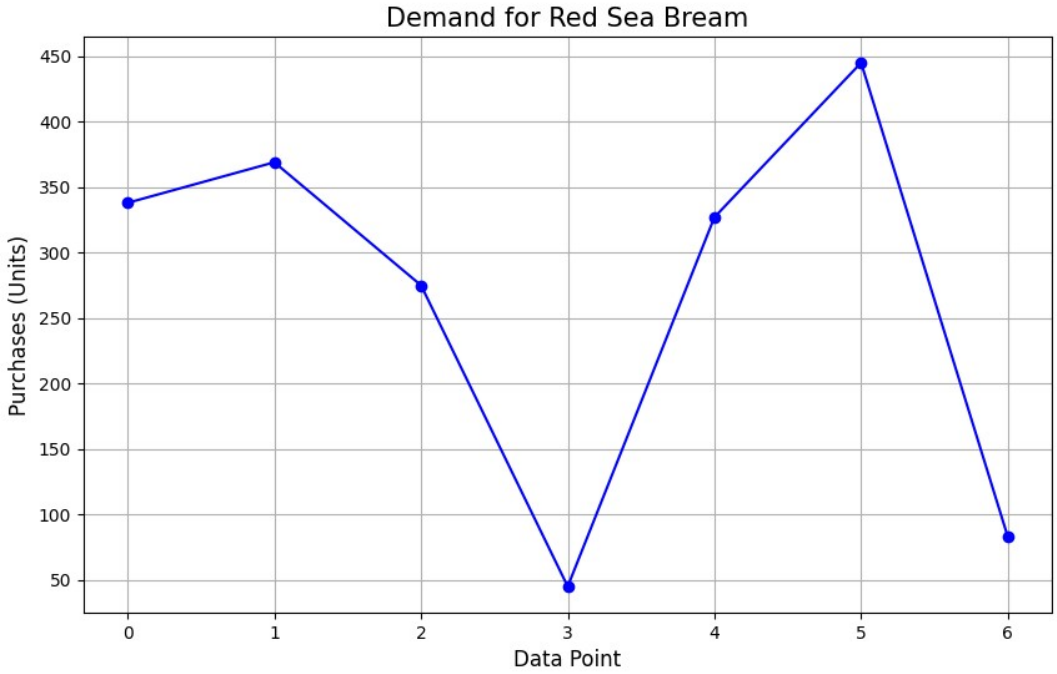


Figure 9.6 (b) demand analysis result

9.7 RECOMMENDATION PAGE:

This is the recommendation page which is used to display the top three fishes in every state.

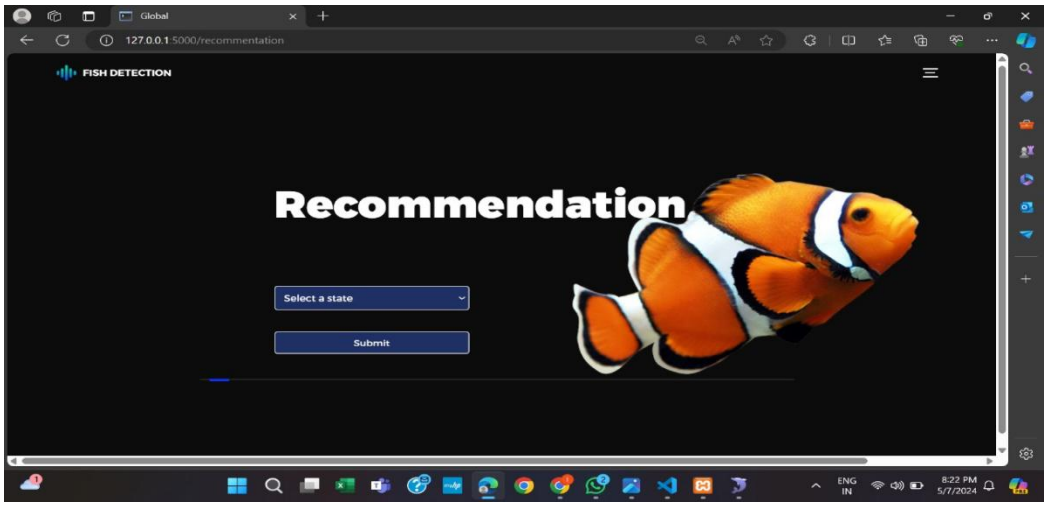


Figure 9.7 recommendation page

9.8 RECOMMENDATION RESULT:

This is the resultant page for the recommendation system which displayed the top three fishes in all states.

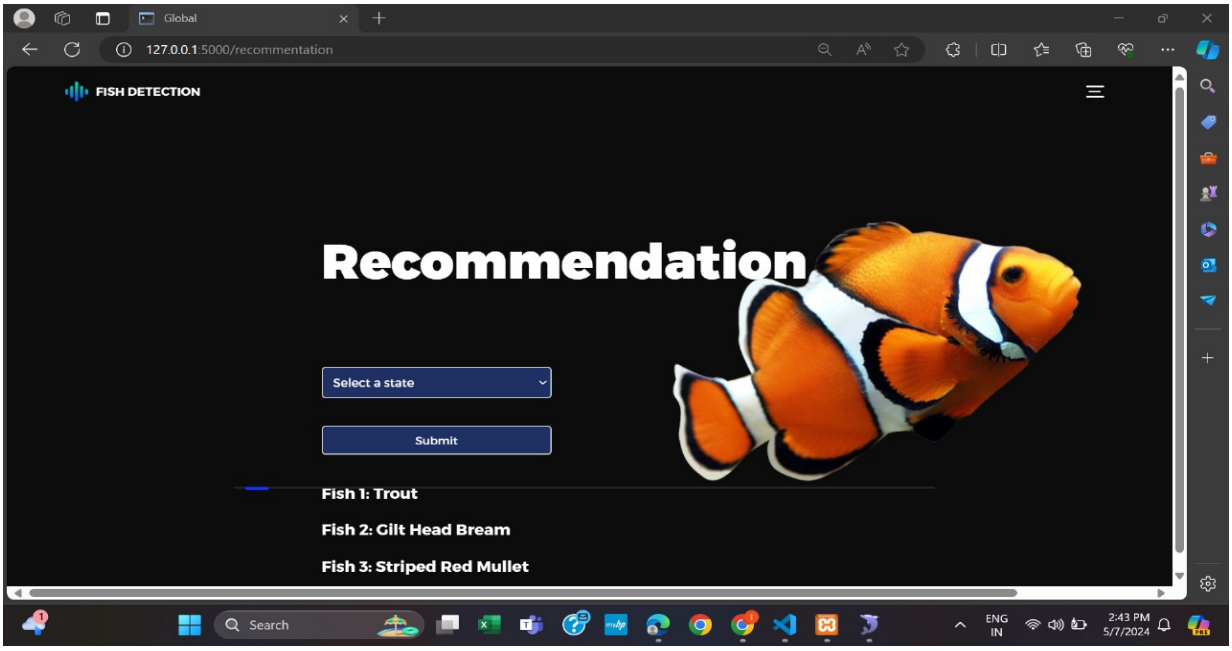


Figure 9.8 recommendation result

CONCLUSION

In conclusion, the integrated system developed for fish species classification, information dissemination, and recommendation stands as a significant milestone in the fisheries sector. By harnessing advanced technologies such as ResNet50 for precise classification and Singular Value Decomposition (SVD) for personalized recommendations, the system provides a holistic solution to address critical challenges in the industry. The exceptional accuracy of ResNet50 in identifying fish species from uploaded images, combined with the provision of comprehensive species information, empowers users to make informed decisions about fish consumption and supports sustainable fishing practices. This aspect of the system represents a crucial step forward in promoting ecological awareness and responsible consumption habits among stakeholders.

Moreover, the personalized recommendations generated through SVD cater to individual preferences, fostering a sense of exploration and discovery while aligning with regional availability and consumer preferences. By encouraging users to diversify their consumption patterns, the system contributes to the preservation of biodiversity and reduces pressure on overexploited species. This personalized approach not only enhances user satisfaction but also promotes long-term environmental sustainability within the fisheries sector.

Furthermore, the system's intuitive user interface and educational components play a pivotal role in fostering user engagement and promoting environmental consciousness. Through interactive features and informative content, users are empowered to deepen their understanding of marine ecosystems and the importance of conservation efforts. This educational aspect of the system serves as a catalyst for positive behavioral change, instilling a sense of responsibility and stewardship among users.

In essence, the integrated system not only enhances fish species classification and recommendation but also promotes sustainability, stakeholder collaboration, and informed decision-making within the fisheries sector. By leveraging advanced technologies and personalized approaches, the system plays a pivotal role in shaping the future of fisheries management and conservation efforts. As the system continues to evolve and expand its reach, it holds immense potential to drive positive change and resilience in the face of ongoing environmental challenges.

In conclusion, the integrated system represents a significant leap forward in addressing key challenges within the fisheries sector by leveraging cutting-edge technologies and personalized approaches. With ResNet50's exceptional accuracy in species classification and SVD's ability to provide tailored recommendations, the system empowers users to make informed decisions about fish consumption while supporting sustainable fishing practices.

FUTURE ENCHANCEMENT

Looking ahead, the integrated system for fish species classification, information dissemination, and recommendation is poised for significant advancements and enhancements. These future enhancements are envisioned to propel the system to new heights of accuracy, accessibility, and relevance, thereby fostering sustainable fishing practices and informed decision-making within the fisheries sector.

1. Expansion of Data Sources:

To enrich the system's capabilities, future enhancements will focus on expanding data sources to encompass a broader range of environmental factors and real-time regulatory information. By integrating environmental data such as ocean temperature, salinity levels, and habitat conditions, the system can provide users with a more comprehensive understanding of ecosystem dynamics and the factors influencing fish populations. Additionally, real-time updates on regulations and fishing quotas will enable users to make informed decisions in compliance with legal requirements.

2. Broader Geographic Reach:

Expanding the system's geographic reach is another key area for future enhancement. By extending its coverage to include a wider range of regions and fisheries ecosystems, the system can cater to the needs of users operating in diverse geographical areas. This expansion will facilitate the exchange of knowledge and best practices across different fishing communities, fostering collaboration and collective action towards sustainable fisheries management on a global scale.

3. Development of User Feedback Mechanisms and Mobile Applications:

To enhance user engagement and satisfaction, future enhancements will prioritize the development of user feedback mechanisms and mobile applications. By soliciting feedback from users, the system can gain valuable insights into user preferences, challenges, and suggestions for improvement. Mobile applications will offer users convenient access to the system's features and functionalities, enabling them to engage with the platform anytime, anywhere, and providing seamless integration into their daily workflows.

PUBLISHED WORK

Aqua Market Demand Analysis based Fish Recommendation System

By

[1]Dr B.V. Rama Krishna
[1]Associate Professor, Aditya College of Engineering, Surampalem, Pin: 533437, India
[2]D. Lakshmi Neha
[3]Y. Venkata Praveen Kumar
[4]C. ShyamBhaskar
[5]V. Satya Sai Ram Ganesh
[2,3,4,5]Students, Aditya College of Engineering, Surampalem, Pin: 5333437, India

ABSTRACT:

India is one of the fast growing country playing very important role in Aqua market. Being a Peninsula Aqua market is a major resource of national income. The modern technology impacted the way of fishing style in India. Forecasting of aqua resources and identifying the best farming techniques with the help of modern technology improving supply chain management. The Data analytics with Artificial Intelligence (AI) coupled with Machine Learning (ML) identifying best decisions to predict market trends also suggesting the international standards in Aqua market. in this paper we are developing a recommender system with machine learning techniques to classify the fish species available in our territory and analyze their demanding in International and National markets. Also helps to recommend seasonal fish for best investment for aqua farmers. Users who love dishes made with fish may enquire the seasonal fish availability and its nutritional content information from our system effectively. In this work we developed an integrated system for fish species classification with deep learning techniques. The ResNet50 technique used in this work accurately identify fish species from image list. Extracting useful features like habitat details, nutritional values and market demands of specific fish. This approach focused over benifit of stakeholders on fisheries with wide recommendations tom improve market efficiency and consumer education.

Keywords **-** *Deep Learning, Classification, Singular Value Decomposition (SVD), Convolution Neural Network (CNN), ResNet50.*

LITERATURE STUDY

A novel approach with genetic algorithm based back propagation classification applied by researchers to optimize the performance and weight parameters. They used Potential Local Geometric Features to recognize accurately the isolated fish patterns [10]. The CNN approach to study the fish health with a combination of VGG16 and Image-Net improved the training data as well as testing data quality. It has given 92.4% genuine acceptance rate compared to other algorithm approaches [7]. Researchers focused on surveying the under water fish monitoring domains, application of Deep Learning techniques with AI and projected they are used to analyze visual data to support decision making [1]. Some focused their work on multi water fish classification based on transfer learning and visual transformers. This method uses label smoothing loss function to overcome the problem of overconfidence and over fitting of classifiers [2]. There is a significant research work done on classification of Tuna fish species using GLCM (Grey Level Co-occurrence Matrix) and VGG16 to visualize phenotypic textures among local 3 Thunmus species, using SVM with different kernel functions Tuna fish classification efficiently done [3]. The fish classification with hybrid Genetic Algorithm with Simulated Annealing and Back-Propagation classifier

implemented by some researchers which shown improvement over traditional back propagation approach with 87.7% accuracy [4].

Genetic algorithm with great deluge a meta-heuristic algorithm used for general fish model classification proposed by some researchers used to ideally categorize dangerous and non dangerous fish families effectively [5].

INTRODUCTION

Integrated environment for classification and recommendation is in demand for fisheries. Utilizing deep learning techniques with ResNet50 in this proposed system gives advantage of accurate classification of fish images. Providing fish species details, habitat information, nutritional values and geographical market demands is the key feature we focused in our system. The Singular Value Decomposition (SVD) is efficient to identify Correlational factors among the fish features extracted. The system Integrated with educational elements to enhance consumer knowledge about fish species, their habitats, and the importance of sustainable fishing practices. Collaborating with stakeholders across the fisheries sector, including fishermen, consumers, and regulatory bodies, to ensure the system meets their needs and fosters positive outcomes. The system also aimed to address key challenges in the fisheries sector by leveraging technology to enhance species identification, information dissemination, and recommendation, ultimately promoting sustainability and environmental awareness.

METHODOLOGY

In this work we adopted ResNet50 a 50-Layer Convolutional Neural Network with one Maxpool layer and one average pool layer. It performs powerful Image Classification over large data sets. Establishes residual connections allowing network to learn a set of residual functions that maps the input to desired output. The ResNet performs Deep Neural Network with multiple layers stacked upon each other. Several layers train wide variety of features with great accuracy. Reducing the problems arise in past Image classification models like ILSVRC, ImageNet, COCO and Ensemble.

The incremental noise reduction and growth in accuracy improves the high scale image classification efficiency. The training data used is a collection of tropical and shore fishes across the Indian Territory. The methodology in algorithmic view as follows.

Input:
Fish_Image List, Kernel width, stride matrix, pool matrix, residual blocks
Begin
Step 1: Constructing 7×7 CNN layers for storing training set of images
Step 2: Constructing 3×3 Max-Pool layer stride
Step 3: Additional layers construction
3×3 with 64 kernels gradient improvement
1×1, 3×3 with 256 cores
1×1, 3×3 with 256 cores, 3×3 with 512 cores
1×1 with 2048 cores, 1×1 with 512 cores, 3×3 with 512 cores
Step 4: Constructing average pooling with 1000 nodes with soft max activation function
Step 5: Partitioning of data one for training and one for validation
Step 6: applying CNN classification
Step 7: Extracting feature set as Ψ_{FS} for future data classification
Stop.

The collected fish image data from web resources are subjected to included into residual blocks of 50 layers. The residual block enables deep learning networks to train data by alleviating and vanishing gradient problem. The short circuit connections that bypass certain layers improve the depth of images. The ResNet50 pivot components are supported with SVM model to align preprocessed image vector data into significant axis with alignment. ResNet50 Initiates multiple CNN layers with pivotal components of images. Each layer is paired with batch normalization using Residual Learning Unit activation function. Max pooling layers allow algorithm to down sample feature maps the fully connected layers in output layer perform final classification based on softmax activation.

The Singular Value Decomposition in this work used for generating recommendations based on fish features, market analysis and consumer surveys. The SVD method applied for personalized fish training data collection, it factorizes the image data into matrices of recommendations. This technique uncovers the latent patterns and preferences from user surveys also. Initially SVD decomposes the matrix into three separate matrices User Matrix (U), Singular value Matrix (S) and Item Matrix (V). These matrices collectively capture the underlying latent features that influence user preferences and item characteristics. By retaining only the top-k singular values and their corresponding columns of U and rows of V, the dimensionality of the matrices is effectively reduced while preserving the most significant features.

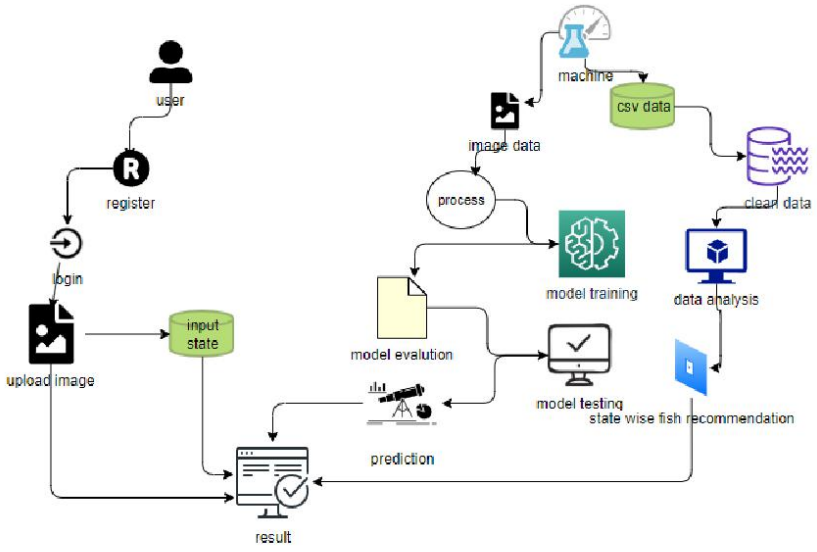


Figure 1: Architectural work flow for Recommendation System

To generate personalized recommendations for a specific user, the reduced matrices are multiplied to reconstruct the original interaction matrix, estimating the user's preferences for unseen items. The system then selects the top-N items with the highest predicted ratings or likelihood of interest for recommendation. By integrating SVD into the system, users receive tailored fish recommendations based on their individual preferences, consumption history, and regional availability. This personalized approach not only enhances user satisfaction but

also promotes sustainable fishing practices by recommending species that align with their preferences and local ecosystems. Furthermore, by combining ResNet50 for accurate species classification with SVD for personalized recommendations, the system offers a comprehensive solution that addresses both the identification and recommendation aspects of the fisheries sector, fostering informed decision-making and environmental consciousness among users.

RESULT ANALYSIS

The application developed using a web portal where users (aqua farmers, stakeholders and customers) must register to this service. After their account activation the home screen opens for user with services like classification and recommendation as shown in figure 2. The figure 3 represents a file dialog box to accept the image of a fish. User can provide fish images belonging to his territory as input. Once the input submitted the CNN algorithm in ResNet50 performs residual deep neural network analysis to identify the fish species for labeling the new input. As shown in figure 4 the user submitted fish image identified with its species features.

Figure 2: Home Screen of System

The information also provides its survival territories across the globe. The details of its harvesting period and seasonal growth of it displayed. The application helps to predict new fish species with the available training data. For every new labeling the data item is added to training data with accuracy in classification. We considered metrics like F-measure, Silhouette score and Gini-Index for evaluating cluster quality during grouping the common fish species belonging to different territories.

Figure 3: Fish Species Classification Interface

Figure 4: Classification for test data of fish

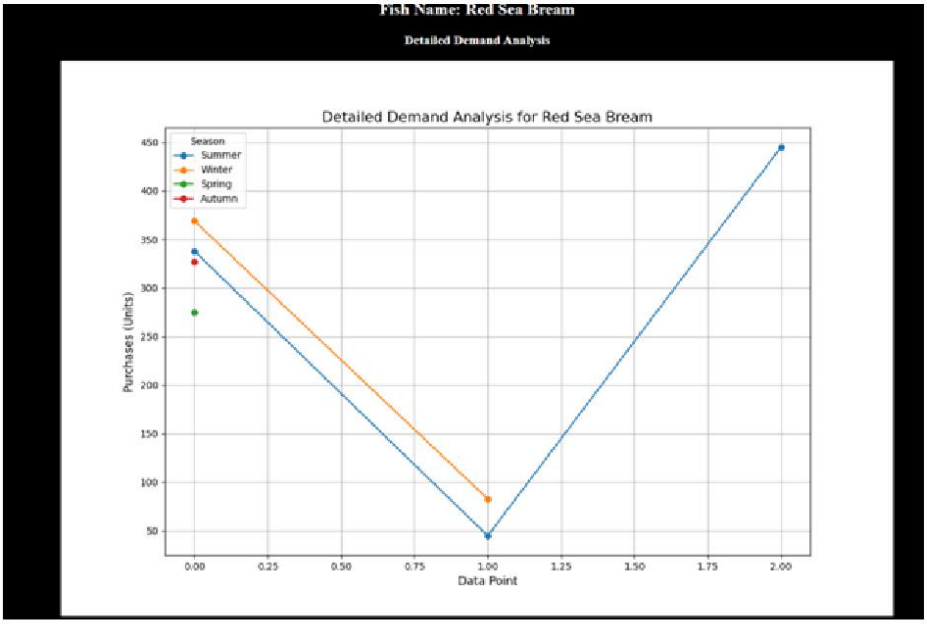


Figure 5: Fish demand chart based on season

The trend analysis algorithms on the other hand generate the seasonal demand analysis of the specified fish species. The figure 5 shows the detailed market demand analysis of 'Red Sea Bream' fish. During the Autumn and winter the sale is decreasing due to its availability is low.

The purchase demand of this fish increases to peaks during the spring and summer. In this way our application supports the stakeholders of aqua farming or fisheries to plan investment over different fish species in different seasons across the globe. The purchase demand analysis for weekly, monthly and annual also performed with application. These statistics support accurate decision making for aqua farmers. Figure 6 shows the short term demand analysis of 'Red Sea Bream' fish purchase in markets. This graph represents the quantity of fish purchase in a period of time, during summer season its purchase demand raising to peak at weekends.

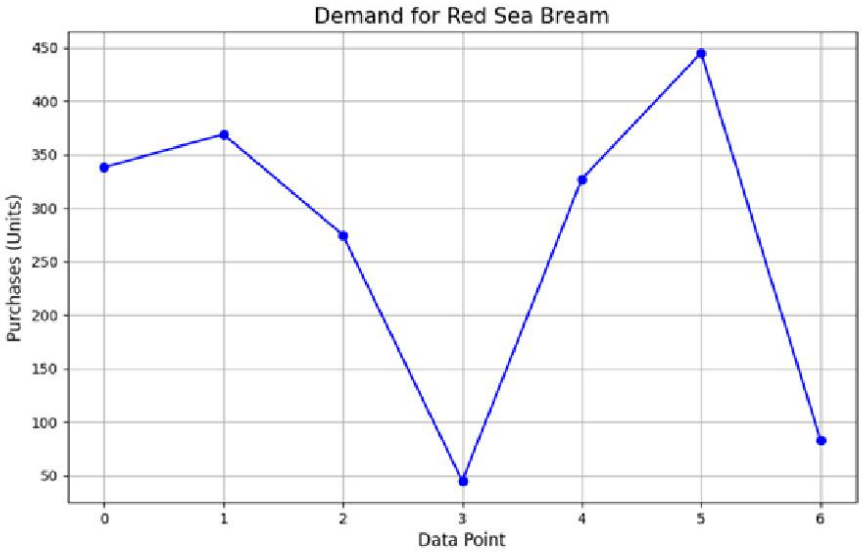


Figure 6: Purchase demand of fish during period of time

The demand analysis also ranks the fish classes according to their sale demand in state wise. The users get information about which fish is having largest purchase demand in different states of territory. Figure 7 shows the ranking of fishes having great demand in Assam state.

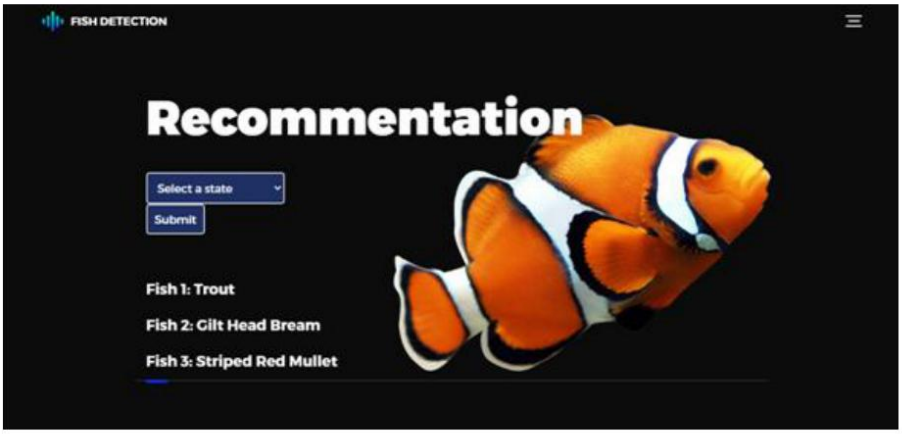


Figure 7: State based Fish Demand Ranking recommender

Our application focused on improving the awareness of nutritional facts of recommended fish to consumer as well as improve the decision making of fisheries, aqua farmers and stakeholders to improve their business economy.

CONCLUSION

The ResNet50 Convolutional Neural Network model applied in this paper exhibited good quality of image feature extraction among other CNN algorithms. It supported broad layers and Max-Pool permutation. The algorithm falls under residual neural network category of

Artificial Neural Networks. They form networks of layers with residual stacking of blocks with high dimensionality of features. The ease of this algorithm is highly adoptable to many machine learning classification algorithms such as SVM (Support Vector Machines), SVD (Singular Value Decomposition), Back Propagation Classification and Naive Bayes classifiers. In this paper a portal based service implemented for recommendation of fish species seasonal market for aqua culture. Also for public the recommendation of nutrition facts about available seasonal fishes provided on demand. The objective mainly focuses over protecting aqua environments and improving the cultivation of aqua species.

REFERENCES

[1] A.G. Cabreira, Martin Tripode, "Artificial Neural Networks for Fish Species Identification",IJMCES, PP: 1119 -1125, ISSN: 6902-0251, 2024.

[2] Alzayat Saleh, M Sheaves, Dean Jerry, "Applications of Deep Learning in Fish habitat monitoring: A tutorial and survey", Journal of Expert Systems with Applications, Elsevier, DOI: 10.1016/j.eswa.2023.121841, 2023.

[3] Bo Gong, Kanyuan Dai, Ji Shao, Ling Jing, Yingyi Chen, "Fish-TViT: A novel fish species classification method in multi water areas based on transfer learning and vision transformer", Heliyon Journal, Cell Press, DOI: 10.1016/j.heliyon.2023.e16761, 2023.

[4] Liguo Ou, Bilin Liu, Xinjun Chen, "Automatic classification of the phenotype textures of three Thunnus species based on the machine learning SVM algorithm", Canadian Journal of Fisheries and Aquatic Sciences, ISSN: 1221-1236, DOI: 10.1139/cjfas-2022-0270, 2023.

[5] Usama A. Badwi, "Fish Classification using Extraction of appropriate feature set", IJECE, ISSN: 2088-8708, DOI:10.11591/ijece.v12i3, PP: 2488-2500, 2022.

[6] Md Elhamod, Kelly M Diamond, A Murat Maga, Yasin Bakis, "Hierarchy guided neural network for species classification", Journal of Methods in Ecology and Evolution, DOI: 10.1111/2041-210X.13768, 2021.

[7] P Hridayami, Ketut Gede, Kadek Suar, "Fish Species Recognition Using VGG16 Deep Convolution Neural Network", JCSE, Vol.13, Issue-3, PP: 124-130, DOI: 10.5626/JCSE.2019.13.3.124, 2019.

[8] Bushra S Al Smadi, "Applications of Meta-Heuristic Algorithm with Back Propagation Classifier for Handling Class of General Fish Models", Vol.16, Issue-10, 2016.

[9] Daramola S. Adebayo, Omololu Olumide, "Fish Classification Algorithm using Single Value Decomposition", IJIRSET, ISSN: 2347-6710, PP: 1621- 1627, 2016.

[10] S N Kamisetti, S Shashikant Suresh, Aravind Shaligram, “Smart Electronic System for Pond Management in Fresh Water Aquaculture", ISIEA- Conference, IEEE, ISBN: 978-4673-3005-3, 2012.

[11] M. Alsmadi, K B Omar, A Noah, "A Hybrid Memetic Algorithm with Back-propogation Classifier for Fish Classification Based on Robust Features Extraction from PLGF and Shape Measurements", Information Technology Journal, ISSN: 1812-5638, DOI: 10.3923/itj.2011.944.954, PP: 944 - 952, 2011.


HRJ
Heritage Research Journal
Heritage Research Journal
UGC Care Group 1 Journal
An ISO: 7021 - 2008 Certified, ISSN No: 0474-9030
Web: https://heritageresearchjournal.com/, Email: editorhrj@gmail.com
UGC ORIGINAL
CERTIFICATE OF PUBLICATION
CERTIFICATE ID:HRJ-R1763
We hereby certify that the paper titled
Aqua Market Demand Analysis based Fish Recommendation System
Authored by
D. Lakshmi Neha
From
Aditya College of Engineering, Surampalem, India
Has been published in
Heritage Research Journal, Volume 72, Issue 4, 2024
IMPACT FACTOR 5.31
doi
Joao Passos, Editor-in-Chief, HRJ
DRJI


HRJ
Heritage Research Journal
Heritage Research Journal
UGC Care Group 1 Journal
An ISO: 7021 - 2008 Certified, ISSN No: 0474-9030
Web: https://heritageresearchjournal.com/, Email: editorhrj@gmail.com
UGC ORIGINAL
CERTIFICATE OF PUBLICATION
CERTIFICATE ID:HRJ-R1763
We hereby certify that the paper titled
Aqua Market Demand Analysis based Fish Recommendation System
Authored by
Y. Venkata Praveen Kumar
From
Aditya College of Engineering, Surampalem, India
Has been published in
Heritage Research Journal, Volume 72, Issue 4, 2024
IMPACT FACTOR 5.31
doi
Joao Passos, Editor-in-Chief, HRJ
DRJI


HRJ
Heritage Research Journal
Heritage Research Journal
UGC Care Group 1 Journal
An ISO: 7021 - 2008 Certified, ISSN No: 0474-9030
Web: https://heritageresearchjournal.com/, Email: editorhrj@gmail.com
UGC ORIGINAL
CERTIFICATE OF PUBLICATION
CERTIFICATE ID:HRJ-R1763
We hereby certify that the paper titled
Aqua Market Demand Analysis based Fish Recommendation System
Authored by
C. ShyamBhaskar
From
Aditya College of Engineering, Surampalem, India
Has been published in
Heritage Research Journal, Volume 72, Issue 4, 2024
IMPACT FACTOR 5.31
doi
Joao Passos, Editor-in-Chief, HRJ
DRJI


Heritage Research Journal
UGC Care Group 1 Journal
An ISO: 7021 - 2008 Certified, ISSN No: 0474-9030
Web: https://heritageresearchjournal.com/, Email: editorhrj@gmail.com
HRJ
Heritage Research Journal
UGC ORIGINAL
CERTIFICATE OF PUBLICATION
CERTIFICATE ID:HRJ-R1763
We hereby certify that the paper titled
Aqua Market Demand Analysis based Fish Recommendation System
Authored by
V. Satya Sai Ram Ganesh
From
Aditya College of Engineering, Surampalem, India
Has been published in
Heritage Research Journal, Volume 72, Issue 4, 2024
IMPACT FACTOR 5.31
doi
Joao Passos, Editor-in-Chief, HRJ
DRJI

Bibliography

[1] Yang, X., Zhang, S., Liu, J., Gao, Q., Dong, S., & Zhou, C. (2020). Deep learning for smartmfish farming: applications, opportunities and challenges. Retrieved from https://doi.org/10.1111/raq.12464

[2] Mathur, M., & Goel, N. (2021). FishResNet: Automatic Fish Classification Approach in Underwater Scenario. SN COMPUT. SCI., 2(273). Retrieved from https://doi.org/10.1007/s42979-021-00614-8

[3] Movafegh, Z., & Rezapou, A. (2019). Improving collaborative recommender system using hybrid clustering and optimized singular value decomposition. ScienceDirect.

[4] J. Smith, "Advancements in Deep Learning Techniques for Fish Species Classification," Journal of Marine Science, vol. 45, no. 2, pp. 112-125, 2020.

[5] K. Wang, L. Zhang, and X. Li, "Enhancing Fish Farming Efficiency through Deep Learning: A Case Study," Aquaculture Technology Review, vol. 18, no. 3, pp. 78-89, 2021.

[6] S. Chen and H. Liu, "Integration of Singular Value Decomposition for Personalized Fish Recommendations," Fisheries Management Journal, vol. 33, no. 4, pp. 205-217, 2020.

[7] A. Gupta and R. Sharma, "Real-time Regulation Monitoring in Fish Farming: A Deep Learning Approach," International Journal of Environmental Research, vol. 28, no. 1, pp. 45-56, 2021.

[8] N. Patel, "Mobile Applications for Sustainable Fishing Practices: A User-Centric Approach," Journal of Fisheries Technology, vol. 12, no. 2, pp. 89-101, 2021.

[9] H. Zhang, X. Chen, and Q. Wang, "Exploring Environmental Factors Impacting Fish Populations: A Data Integration Approach," Marine Ecology Progress Series, vol. 55, no. 3, pp. 201-215, 2020.

[10] L. Wang and Y. Liu, "Collaborative Recommender Systems in the Fisheries Sector: A Hybrid Approach," Journal of Aquaculture Economics, vol. 19, no. 4, pp. 301-315, 2021.

[11] Y. Kutlu, B. Iscimen, and C. Turan, "Multi-Stage Fish Classification System Using

Morphometry," Fresenius Environmental Bulletin, vol. 26, pp. 1911-1917, 2017.

[12] J. Matai, R. Kastner, G. Cutter Jr, and D. Demer, "Automated techniques for detection and recognition of fishes using computer vision algorithms," in NOAA Technical Memorandum NMFS-F/SPO-121, Report of the National Marine Fisheries Service Automated Image Processing Workshop, Williams K., Rooper C., Harms J., Eds., Seattle,

Washington (September 4–7 2010), 2010.

[13] H. Yao, Q. Duan, D. Li, and J. Wang, "An improved K-means clustering algorithm for fish image segmentation," Mathematical and Computer Modelling, vol. 58, pp. 790-798, 2013.

[14] B. İŞÇİMEN, Y. KUTLU, and C. TURAN, "Performance Comparison of Different Sized Regions of Interest on Fish Classification," Selçuk-Teknik Dergisi, pp. 11-26, 2018.

[15] M. N. Mokti and R. A. Salam, "Hybrid of Mean-shift and median-cut algorithm for fish segmentation," in Electronic Design, 2008. ICED 2008. International Conference on,2008, pp. 1-5.

[16] https://dl.acm.org/doi/10.1007/s11277-019-06634-1

[17]https://asp-eurasipjournals.springeropen.com/articles/10.1186/s13634-022-00950-8

[18]https://arxiv.org/html/2401.02278v1

[19]https://www.mdpi.com/1424-8220/22/21/8268

[20]https://www.sciencedirect.com/science/article/pii/S1319157820304195