

Industry Project Report

On

AI/ML Hotel Price Predictor

Developed By: -

Munish Patwa (21162121017)

Guided By:-

Prof. Sulabh Bhatt (Internal)

Rahul Bhagchandani(External)

Submitted to
Faculty of Engineering and Technology
Institute of Computer Technology
Ganpat University



Year - 2025



Ganpat University

॥ विद्यया समाजोत्कर्षः ॥

Institute of Computer Technology

DIAMOND

QS I-GAUGE

INDIAN UNIVERSITY RATINGS



GSIRF



CERTIFICATE

This is to certify that the **Industry** Project work entitled “**AI/ML Hotel Price Predictor**” by Munish Patwa (Enrolment No. 21162121017) of Ganpat University, towards the partial fulfillment of requirements of the degree of Bachelor of Technology – Computer Science and Engineering, carried out by them in the AI Department at Ascion Global Partners. The results/findings contained in this Project have not been submitted in part or full to any other University / Institute for the award of any other Degree/Diploma.

Name & Signature of Internal Guide

Name & Signature of Head

Place: ICT - GUNI

Date: 22 February 202

ACKNOWLEDGEMENT

Industry Internship project is a golden opportunity for learning and self-development. I consider myself very lucky and honored to have so many wonderful people lead me through in completion of this project. First and foremost, I would like to thank Dr. Rohit Patel, Principal, ICT, and Prof. Dharmesh Darji , Head, ICT who gave us an opportunity to undertake this project. My grateful thanks to Prof. Tejas Kadiya, Prof. Sulabh Bhatt, and Mr. Rahul Bhagchandani (Internal & External Guides) for their guidance in project work AI/ML Hotel Price Predictor, who despite being extraordinarily busy with academics, took time out to hear, guide and keep me on the correct path. I do not know where I would have been without his/her help.

MUNISH PATWA (Enrollment No:21162121017)

ABSTRACT

In the hospitality industry, pricing strategies play a crucial role in maximizing revenue and ensuring competitive advantage. This project focuses on developing an AI/ML-based hotel price prediction model to forecast hotel prices based on various factors such as location, seasonality, hotel type, and amenities. The process begins with the manual collection of hotel data, which is stored in a MySQL database. Following data acquisition, key features are extracted and selected to ensure optimal model performance. Two machine learning approaches, namely Random Forest Regressor and a Deep Neural Network (DNN), are employed to predict the prices with high accuracy. The Random Forest model, leveraging an ensemble of decision trees, provides interpretability and robustness, while the DNN model aims to capture complex patterns within the data for improved accuracy. The model's performance is evaluated using standard regression metrics, and the results indicate promising potential for practical applications in dynamic pricing, revenue management, and competitive analysis within the hospitality industry. This project showcases how advanced AI/ML techniques can be leveraged to automate price forecasting, ultimately optimizing pricing strategies and enhancing business decision-making in the hotel industry.

INDEX

1.	INTRODUCTION		1
2.	PROJECT SCOPE		3
3.	SOFTWARE AND HARDWARE REQUIREMENTS		5
4.	PROCESS MODEL		7
5.	PROJECT PLAN		9
	5.1	List of Major Activities	10
	5.2	Estimated Time Duration in Days	11
6.	IMPLEMENTATION DETAILS		12
	6.1	System Architecture	13
	6.2	Workflow Implementation	14
		6.2.1 User Input & Frontend Processing	14
		6.2.2 Query Processing via LLM	14
		6.2.3 Query Validation & Execution	14
		6.2.4 Fetching & Displaying Results	15
		6.2.5 Displaying Result	15
	6.3	Machine Learning Model Integration With Prediction Workflow	15
		6.3.1 Feature Engineering Template Design	15
		6.3.2 Machine Learning Model Interaction	16
	6.4	Backend Implementation	16
	6.5	Frontend Implementation	16
	6.6	Database Structure	16
	6.7	Conclusion	17

7.	CONCLUSION & FUTURE WORK		18
	7.1	Conclusion	19
	7.2	Future work	19
8.	REFERENCES		20

CHAPTER: 1 INTRODUCTION

CHAPTER 1 INTRODUCTION

In today's data-driven era, the ability to accurately predict pricing strategies is crucial in industries such as hospitality. With the growing competition in the hotel sector, dynamic pricing has become essential to maximize revenue and ensure competitiveness. However, predicting hotel prices involves processing large volumes of data, including factors such as location, seasonality, customer demand, amenities, and hotel type. For hotel managers and pricing strategists, manually analyzing these factors can be time-consuming and error-prone.

This project aims to develop an AI/ML-based hotel price prediction system that can automate this process and provide accurate pricing forecasts. By leveraging machine learning models such as Random Forest Regressor and Deep Neural Networks (DNN), the model processes various hotel attributes and predicts optimal pricing based on historical data. The system is designed to allow stakeholders to input hotel characteristics and receive dynamic, data-driven price recommendations, facilitating smarter decision-making.

The project combines traditional data processing techniques with advanced machine learning algorithms, providing an efficient and scalable solution for predicting hotel prices. The goal is to provide a tool that not only simplifies the pricing process but also helps businesses optimize their revenue strategies by analyzing complex patterns in hotel pricing. Through the use of feature extraction and selection techniques, the model aims to ensure high accuracy and generalization across diverse hotel data sets.

The following chapters will delve into the objectives, background, methodology, and implementation details of this project, along with its expected outcomes and future scope.

Below is the list of the tools and technologies used in this project:

- **MySQL:** Database management system used for storing hotel data and facilitating data manipulation.
- **Python:** Primary programming language for data processing, model development, and evaluation.
- **Random Forest Regressor:** Machine learning model used for predicting hotel prices based on input features.
- **Deep Neural Networks (DNN):** Advanced model used for capturing complex patterns and improving price predictions.
- **Pandas:** Python library used for data manipulation and analysis during the feature extraction and preprocessing phases.
- **Scikit-learn:** Machine learning library used for implementing and training the Random Forest model.
- **TensorFlow/Keras:** Frameworks used for building and training the Deep Neural Network model.
- **Jupyter Notebook:** Environment for data analysis, model experimentation, and testing.

CHAPTER: 2 PROJECT SCOPE

CHAPTER 2 PROJECT SCOPE

The scope of this project is to develop an AI-powered hotel price prediction system that leverages machine learning models to forecast hotel pricing based on various features, including location, hotel type, amenities, seasonality, and demand. This system is designed to assist hotel managers, pricing strategists, and other stakeholders in making informed pricing decisions by providing accurate, data-driven price recommendations.

The primary goal of this project is to build a robust model using Random Forest Regressor and Deep Neural Networks (DNN) to predict hotel prices based on historical data. The system is capable of processing large datasets of hotel records stored in MySQL, and it incorporates feature extraction and selection techniques to ensure the quality and relevance of input data. The model will be trained on diverse hotel data, allowing it to generalize well across different hotel types and geographical locations.

Key components of the system include:

- **Data Collection and Storage:** Hotel data is manually entered into MySQL, where it is stored and made available for processing.
- **Machine Learning Models:** The project uses Random Forest Regressor and Deep Neural Networks (DNN) for price prediction. These models will be evaluated for accuracy, with the goal of determining the most effective approach for forecasting hotel prices.
- **Feature Engineering:** Techniques such as feature extraction and selection will be used to enhance model performance by identifying the most important variables influencing hotel prices.
- **User Interface (UI):** While this project focuses on the model itself, the long-term scope includes the development of a user-friendly interface to allow non-technical users to input hotel attributes and receive price predictions.

This project aims to provide an automated, scalable, and data-driven solution for dynamic hotel pricing, helping stakeholders optimize revenue management strategies and gain competitive insights. It will serve as a tool to streamline pricing processes in the hospitality industry by incorporating AI/ML-driven predictions.

CHAPTER: 3 SOFTWARE AND HARDWARE REQUIREMENTS

CHAPTER 3 SOFTWARE AND HARDWARE REQUIREMENTS

Minimum Hardware Requirements

Component	Specification
Processor	Intel Core i3 / AMD Ryzen 3 (or equivalent)
RAM	4GB
HDD	30GB

Table 3.1 Minimum Hardware Requirements

Minimum Software Requirements

Component	Specification
Operating System	Windows, macOS, or Linux
Web Browser	Chrome, Firefox, Edge, or Safari (latest versions recommended)
Python	Python 3.7 or above
Machine Learning Libraries	Scikit-learn, TensorFlow/Keras, Pandas, NumPy (for model training and evaluation)
Database	MySQL (for storing hotel records and querying the data)
IDE/Notebook	Jupyter Notebook or any preferred IDE (e.g., VSCode) for model development and experimentation

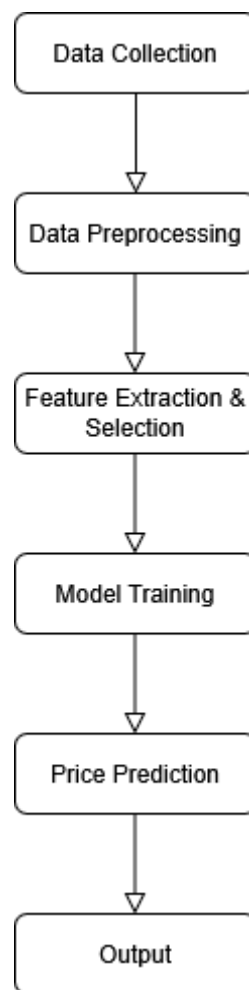
Table 3.2 Minimum Software Requirements

CHAPTER: 4 PROCESS MODE

CHAPTER 4 PROCESS MODEL

This project follows a structured workflow to ensure the accurate prediction of hotel prices based on various features. The system processes historical hotel data and user inputs, applies machine learning models, and generates price predictions. The workflow includes data collection, preprocessing, feature engineering, model training, and prediction. Below is a simplified diagram illustrating the process:

1. **Data Collection:** Hotel records are manually entered into MySQL database, including attributes like location, amenities, seasonality, and hotel type.
2. **Data Preprocessing:** The collected data is cleaned, and missing values are handled to ensure consistency.
3. **Feature Extraction & Selection:** Key features affecting hotel prices are identified and selected for training the machine learning models.
4. **Model Training:** Random Forest Regressor and Deep Neural Network models are trained using the preprocessed data to learn pricing patterns.
5. **Price Prediction:** When a user inputs hotel characteristics (such as location, amenities, and type), the trained models predict the optimal price based on the learned patterns.
6. **Output:** The predicted price is returned to the user as a result.



CHAPTER: 5 PROJECT PLAN

CHAPTER 5 PROJECT PLAN

5.1 List of Major Activities

1. **Research & Exploration** – Understanding the problem domain, exploring relevant machine learning models (Random Forest, Deep Neural Networks), and reviewing feature engineering techniques.
2. **Data Collection & Database Setup** – Manually collecting hotel data and setting up MySQL for storing hotel records.
3. **Data Preprocessing & Feature Engineering** – Cleaning the collected data, handling missing values, and selecting key features that affect hotel prices.
4. **Model Development & Training** – Training the Random Forest Regressor and Deep Neural Network models using the processed data.
5. **Model Evaluation & Tuning** – Evaluating model performance using appropriate metrics (e.g., RMSE, MAE) and fine-tuning hyperparameters.
6. **Backend Development (Optional)** – Developing the backend using FastAPI to handle user input and process predictions.
7. **Frontend Development (Optional)** – Designing a simple user interface where users can input hotel features and view the predicted price.
8. **System Integration (Optional)** – Integrating the machine learning models with the backend and connecting it to the frontend.
9. **Testing & Debugging (Optional)** – Ensuring the system's stability, testing the models' accuracy, and ensuring smooth user interactions.
10. **Final Deployment & Documentation (Optional)** – Deploying the system for use and preparing project reports and user documentation.

5.2 Estimated Time Duration in Days

Activity	Estimated Duration (Days)
Research & Exploration	10 Days
Data Collection & Database Setup	8 Days
Data Preprocessing & Feature Engineering	12 Days
Model Development & Training	14 Days
Model Evaluation & Tuning	10 Days
Backend Development (Optional)	12 Days
Frontend Development (Optional)	12 Days
System Integration (Optional)	8 Days
Testing & Debugging (Optional)	10 Days
Final Deployment & Documentation (Optional)	6 Days

Total Estimated Duration: ~2.5 Months

Table 5.1 Task Completion Estimated Time Duration in Days

CHAPTER: 6 IMPLEMENTATION DETAILS

CHAPTER 6 IMPLEMENTATION DETAIL

This chapter provides an in-depth explanation of the implementation of the AI-based hotel price prediction system. It covers the system's architecture, workflow, integration of technologies, and a step-by-step breakdown of each module involved.

6.1 System Architecture

The system follows a client-server architecture, where the frontend, backend, and database communicate seamlessly to process hotel price predictions based on user inputs. The main components of the system include:

1. **Frontend (React.js):**

The frontend of the system is built using React.js. It provides a user-friendly interface where users can input various hotel attributes such as location, hotel type, amenities, and seasonality. The interface is designed to be intuitive, allowing users to interact with the system easily. After receiving the input, the frontend sends the data to the backend for processing and displays the predicted hotel price to the user.

2. **Backend (FastAPI):**

The backend is developed using FastAPI, which is used to handle user requests. It processes the input data sent from the frontend and passes it to the machine learning models for price prediction. FastAPI communicates with the trained machine learning models (Random Forest Regressor and Deep Neural Network) and retrieves the predicted prices. The backend also manages the interaction with the MySQL database, where the hotel data is stored.

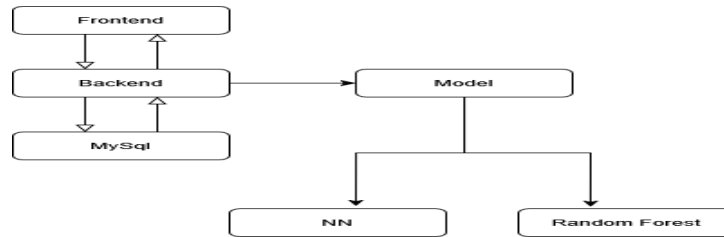
3. **Machine Learning Models (Random Forest Regressor & Deep Neural Network):**

The system uses two models for price prediction:

- **Random Forest Regressor:** An ensemble learning method that uses decision trees to predict the price based on historical data and selected features.
- **Deep Neural Network (DNN):** A neural network model designed to learn complex patterns in the hotel pricing data and make predictions. The model was trained using a deep learning framework (e.g., TensorFlow or Keras).

4. **Database (MySQL):**

MySQL is used as the database to store hotel records, including attributes like location, hotel type, amenities, and pricing data. The database is pre-populated with hotel data that is used for training the models and making price predictions. The backend queries the database to retrieve relevant hotel data for input into the machine learning models.



6.2 Workflow Implementation

This section outlines the detailed workflow of the hotel price prediction system, covering each stage from user input to the final price prediction. It includes interactions between the frontend, backend, machine learning models, and database.

6.2.1 User Input & Frontend Processing (Planned/Optional)

- The user interacts with the **React-based frontend** by entering hotel-related features, such as:
 - **Location** (e.g., city or region)
 - **Hotel Type** (e.g., budget, luxury)
 - **Amenities** (e.g., free Wi-Fi, pool)
 - **Seasonality** (e.g., peak season or off-season)
- Once the user submits their input, the frontend sends this data to the **FastAPI backend** in the form of a structured API request.

6.2.2 Feature Extraction & Preprocessing (Data Processing in Backend)

- The backend receives the hotel feature data from the frontend and performs **data preprocessing** to ensure it is in a usable format for the machine learning models.
- The backend may handle tasks like:
 - **Handling missing values:** Ensuring that all necessary features are present and valid.
 - **Normalizing or encoding:** Converting categorical data (e.g., hotel type, location) into numerical representations if necessary.
- Preprocessed data is then passed to the **machine learning models** for price prediction.

6.2.3 Price Prediction via Machine Learning Models

- The **Random Forest Regressor** and **Deep Neural Network (DNN)** models, trained on historical hotel data, are used to predict the price.
 - **Random Forest Regressor:** An ensemble model that predicts hotel prices based on various features by aggregating predictions from decision trees.
 - **Deep Neural Network (DNN):** A neural network model that can capture complex, non-linear relationships between hotel features and price.
- The backend sends the preprocessed input to the models, which generate the **predicted hotel price** based on the given features.

6.2.4 Query Execution & Result Fetching

- After receiving the predicted price, the backend prepares the data to be sent back to the frontend.
- The system queries the **MySQL database** (if needed) to fetch any related records or historical data that may help in validating the prediction.
- The predicted price and any supplementary data (if available) are packaged in a **structured format** (e.g., a JSON response) and sent back to the frontend.

6.2.5 Displaying Results to the User (Yet to be Integrated with Frontend)

- The **React frontend** receives the predicted price (and any relevant supplementary information) from the backend.
- The results are displayed to the user in a clear and readable format, such as:
 - A **predicted hotel price** displayed on the screen.
 - A **summary of the key features** (location, hotel type, etc.) influencing the price.
- If necessary, additional features (like graphs or visualizations) may be added to help the user understand the factors affecting the price.

6.3 Machine Learning Model Integration and Prediction Workflow

This section outlines how the machine learning models (specifically the Random Forest Regressor and Deep Neural Network) interact with the system to predict hotel prices. The models are trained to process the input features (such as location, hotel type, amenities) and generate price predictions. This section also explains the design of the prompt template, which ensures the models receive the correct input format for accurate predictions.

6.3.1 Feature Engineering Template Design

The feature engineering template plays a critical role in preparing the input data for the machine learning models. It defines the structure and format of the input data to ensure that the features are extracted and preprocessed correctly before being sent to the models. The template focuses on providing context about the data required for accurate predictions. It also ensures that:

- Categorical features (e.g., hotel type, amenities) are encoded appropriately.
- Numerical features (e.g., number of rooms, hotel rating) are normalized or scaled.
- Missing data is handled appropriately, either by imputation or exclusion.

The feature engineering template helps streamline the process of transforming raw user input into a format that the models can understand, allowing for more accurate price predictions.

6.3.2 Machine Learning Model Interaction

The Random Forest Regressor and Deep Neural Network (DNN) are responsible for generating hotel price predictions based on the user's input. The interaction between the system and the models is as follows:

- **User Input:** The user provides input on hotel-related features such as location, hotel type, and amenities.
- **Data Preprocessing:** The backend processes this input data based on the feature engineering template. The input is transformed into a numerical format that can be fed to the models.
- **Model Processing:** The preprocessed input is passed to the Random Forest Regressor or DNN for price prediction. The models generate a predicted hotel price based on the trained data.
- **Response Evaluation:** The predicted price is returned to the backend. The backend evaluates whether the prediction is valid, and any necessary adjustments or refinements are made before sending it to the frontend.

Once the models generate the price prediction, the backend sends the results back to the frontend, which displays them to the user in an understandable format.

6.4 Backend Implementation - FastAPI (Planned/Optional)

6.5 Frontend Implementation (React.js) (Planned/Optional)

6.6 Database Structure

Booking_ID	Unique Varchar
Room Type	Varchar
Availability	Integer
Date	Date
Holiday	Boolean
Price	Integer
Guest_Name	Varchar
Email	Varchar
Phone Number	Varchar

6.7 Conclusion

This chapter provides an overview of the current status of the hotel price prediction system and highlights the next steps required to complete the project. While some core components of the system, such as machine learning models (Random Forest Regressor and DNN), have been defined, the backend and frontend are still under development.

Key aspects of the project that remain to be developed include:

- **Backend Development:** The FastAPI-based backend, which will process user inputs, interact with the machine learning models, and manage database queries, is yet to be implemented.
- **Frontend Development:** The React-based user interface for user interaction and displaying price predictions is still in progress.
- **Error Handling:** The error handling module to validate predictions and ensure the system functions reliably needs to be developed.
- **Security Mechanisms:** Implementing role-based access control and securing the system from potential vulnerabilities will be done in the coming stages.
- **Cloud Deployment:** The system is currently running locally, and cloud deployment to ensure scalability and wider access will be a future task.

As the project progresses, more detailed documentation will be added, including UI screenshots, workflow diagrams, and backend response examples. This will help provide a clearer understanding of how each component functions and integrates with the system.

CHAPTER: 7 CONCLUSION AND FUTURE WORK

7.1 Conclusion

This project aims to leverage machine learning models to predict hotel prices based on user inputs, simplifying the pricing prediction process for users in the hospitality industry. By integrating machine learning models (Random Forest Regressor and Deep Neural Network) that can generate dynamic hotel price predictions based on a range of features, such as location, hotel type, and amenities.

So far, we have successfully established the machine learning models, set up the database for storing relevant hotel data, and created the initial framework for the backend and frontend. Although the backend and frontend development are ongoing, the key components of data preprocessing, feature engineering, and model training have been completed, demonstrating the feasibility of the approach.

The system, once fully developed, will provide a user-friendly interface where users can input hotel characteristics, and the system will predict hotel prices based on trained models. This will significantly streamline the pricing decision process for hotels, enabling users to make more informed, data-driven decisions.

7.2 Future Work

Moving forward, several critical steps will be undertaken to enhance and expand the system's capabilities:

- **Full-Stack Integration:** Connecting the **backend (FastAPI)** with the **frontend (React.js)** to enable real-time hotel price prediction. This will allow users to interact with the system and receive predictions based on their input features dynamically.
- **Dynamic Feature Handling:** Implementing a more dynamic system for handling input features and ensuring that only the relevant features (e.g., location, hotel type) are used for each prediction. This will optimize the prediction process and improve system flexibility.
- **Enhanced Error Handling and Security:** Improving the robustness of the system by adding **error handling** for invalid or incomplete inputs and implementing **role-based access control (RBAC)** to secure the application and restrict unauthorized access.
- **Model Optimization and Testing:** Exploring alternative machine learning models, such as **Gradient Boosting Machines (GBM)** or **XGBoost**, to compare their performance and accuracy with the current models. Fine-tuning the current models for improved prediction accuracy and efficiency is also planned.
- **User Testing and Interface Improvements:** Conducting thorough **user testing** to refine the frontend interface, ensuring that it is intuitive and user-friendly. Based on feedback, we will make adjustments to improve the overall user experience.
- **Documentation and Deployment:** Finalizing the system's **documentation** and preparing for **cloud deployment** to ensure scalability and easy access for a wider audience. This will involve transitioning the system from a local environment to a cloud platform for production use.

In conclusion, while significant progress has been made in developing the foundational components of the system, ongoing work will focus on full-stack integration, optimizing the machine learning models, and preparing the system for deployment and broader use. The successful completion of these steps will result in a robust, user-friendly platform capable of delivering accurate hotel price predictions.




CHAPTER: 8 REFERENCES

CHAPTER 8 REFERENCES

1. Random Forest Regressor - Scikit-learn Documentation
<https://scikit-learn.org/stable/modules/ensemble.html#random-forest>
2. Deep Learning for Regression Models - Towards Data Science
<https://towardsdatascience.com/deep-learning-for-regression-5e0d17b15657>
3. Pandas Documentation (read_sql method)
https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.read_sql.html
4. FastAPI Documentation
<https://fastapi.tiangolo.com/>
5. React.js Documentation
<https://reactjs.org/docs/getting-started.html>
6. Hotel Price Prediction with Machine Learning
<https://www.analyticsvidhya.com/blog/2021/10/a-complete-guide-to-hotel-price-prediction-with-machine-learning/>
7. Hotel Data Analysis with Python and Pandas
<https://www.kaggle.com/atharvaingle/hotel-price-prediction-using-machine-learning>

Munish Patwa

AI_ML_INDUSTRY PROJECT REPORT_NoCerti.pdf

-  Assignment 3
-  Research_1
-  Ganpat University

Document Details

Submission ID

trn:oid::1:3238847955

Submission Date

May 4, 2025, 10:25 PM GMT+5:30

Download Date

May 4, 2025, 10:36 PM GMT+5:30

File Name

AI_ML_INDUSTRY_PROJECT_REPORT_NoCerti.pdf

File Size

396.3 KB

23 Pages





3,393 Words

19,855 Characters




8% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Match Groups

-  **31 Not Cited or Quoted 8%**
Matches with neither in-text citation nor quotation marks
-  **3 Missing Quotations 1%**
Matches that are still very similar to source material
-  **0 Missing Citation 0%**
Matches that have quotation marks, but no in-text citation
-  **0 Cited and Quoted 0%**
Matches with in-text citation present, but no quotation marks

Top Sources

- 5%  Internet sources
- 5%  Publications
- 0%  Submitted works (Student Papers)

Match Groups

- 31 Not Cited or Quoted 8%**
Matches with neither in-text citation nor quotation marks
- 3 Missing Quotations 1%**
Matches that are still very similar to source material
- 0 Missing Citation 0%**
Matches that have quotation marks, but no in-text citation
- 0 Cited and Quoted 0%**
Matches with in-text citation present, but no quotation marks

Top Sources

- 5% Internet sources
- 5% Publications
- 0% Submitted works (Student Papers)

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	Publication	R. N. V. Jagan Mohan, B. H. V. S. Rama Krishnam Raju, V. Chandra Sekhar, T. V. K. P...	3%
2	Internet	www.coursehero.com	<1%
3	Internet	fenilsonani.com	<1%
4	Internet	github.com	<1%
5	Internet	www.mdpi.com	<1%
6	Publication	Howard Anderson, Sharon Yull, Bruce Hellingsworth. "Higher National Computin...	<1%
7	Internet	scholarworks.sjsu.edu	<1%
8	Internet	syllabus.africacode.net	<1%
9	Publication	Di Wu. "Data Mining with Python - Theory, Application, and Case Studies", CRC Pr...	<1%
10	Internet	core.ac.uk	<1%

11	Internet	store.steampowered.com	<1%
12	Internet	epon.org	<1%
13	Internet	mediate-project.eu	<1%
14	Internet	www.nerc.com	<1%
15	Publication	Iacopo Carnacina, Mawada Abdellatif, Manolia Andredaki, James Cooper, Darren ...	<1%
16	Internet	archive.org	<1%
17	Internet	digital.library.unt.edu	<1%
18	Internet	matheo.uliege.be	<1%

INDEX

1.	INTRODUCTION		1
2.	PROJECT SCOPE		3
3.	SOFTWARE AND HARDWARE REQUIREMENTS		5
4.	PROCESS MODEL		7
5.	PROJECT PLAN		9
	5.1	List of Major Activities	10
	5.2	Estimated Time Duration in Days	11
6.	IMPLEMENTATION DETAILS		12
	6.1	System Architecture	13
	6.2	Workflow Implementation	14
		6.2.1 User Input & Frontend Processing	14
		6.2.2 Query Processing via LLM	14
		6.2.3 Query Validation & Execution	14
		6.2.4 Fetching & Displaying Results	15
		6.2.5 Displaying Result	15
	6.3	Machine Learning Model Integration With Prediction Workflow	15
		6.3.1 Feature Engineering Template Design	15
		6.3.2 Machine Learning Model Interaction	16
	6.4	Backend Implementation	16
	6.5	Frontend Implementation	16

	6.6	Database Structure	16
	6.7	Conclusion	17
7.	CONCLUSION & FUTURE WORK		18
	7.1	Conclusion	19
	7.2	Future work	19
8.	REFERENCES		20



CHAPTER: 1 INTRODUCTION

CHAPTER 1 INTRODUCTION

In today's data-driven era, the ability to accurately predict pricing strategies is crucial in industries such as hospitality. With the growing competition in the hotel sector, dynamic pricing has become essential to maximize revenue and ensure competitiveness. However, predicting hotel prices involves processing large volumes of data, including factors such as location, seasonality, customer demand, amenities, and hotel type. For hotel managers and pricing strategists, manually analyzing these factors can be time-consuming and error-prone.

This project aims to develop an AI/ML-based hotel price prediction system that can automate this process and provide accurate pricing forecasts. By leveraging machine learning models such as Random Forest Regressor and Deep Neural Networks (DNN), the model processes various hotel attributes and predicts optimal pricing based on historical data. The system is designed to allow stakeholders to input hotel characteristics and receive dynamic, data-driven price recommendations, facilitating smarter decision-making.

The project combines traditional data processing techniques with advanced machine learning algorithms, providing an efficient and scalable solution for predicting hotel prices. The goal is to provide a tool that not only simplifies the pricing process but also helps businesses optimize their revenue strategies by analyzing complex patterns in hotel pricing. Through the use of feature extraction and selection techniques, the model aims to ensure high accuracy and generalization across diverse hotel data sets.

The following chapters will delve into the objectives, background, methodology, and implementation details of this project, along with its expected outcomes and future scope.

Below is the list of the tools and technologies used in this project:

- **MySQL:** Database management system used for storing hotel data and facilitating data manipulation.
- **Python:** Primary programming language for data processing, model development, and evaluation.
- **Random Forest Regressor:** Machine learning model used for predicting hotel prices based on input features.
- **Deep Neural Networks (DNN):** Advanced model used for capturing complex patterns and improving price predictions.
- **Pandas:** Python library used for data manipulation and analysis during the feature extraction and preprocessing phases.
- **Scikit-learn:** Machine learning library used for implementing and training the Random Forest model.
- **TensorFlow/Keras:** Frameworks used for building and training the Deep Neural Network model.
- **Jupyter Notebook:** Environment for data analysis, model experimentation, and testing.



CHAPTER: 2 PROJECT SCOPE

CHAPTER 2 PROJECT SCOPE

The scope of this project is to develop an AI-powered hotel price prediction system that leverages machine learning models to forecast hotel pricing based on various features, including location, hotel type, amenities, seasonality, and demand. This system is designed to assist hotel managers, pricing strategists, and other stakeholders in making informed pricing decisions by providing accurate, data-driven price recommendations.

The primary goal of this project is to build a robust model using Random Forest Regressor and Deep Neural Networks (DNN) to predict hotel prices based on historical data. The system is capable of processing large datasets of hotel records stored in MySQL, and it incorporates feature extraction and selection techniques to ensure the quality and relevance of input data. The model will be trained on diverse hotel data, allowing it to generalize well across different hotel types and geographical locations.

Key components of the system include:

- Data Collection and Storage: Hotel data is manually entered into MySQL, where it is stored and made available for processing.
- Machine Learning Models: The project uses Random Forest Regressor and Deep Neural Networks (DNN) for price prediction. These models will be evaluated for accuracy, with the goal of determining the most effective approach for forecasting hotel prices.
- Feature Engineering: Techniques such as feature extraction and selection will be used to enhance model performance by identifying the most important variables influencing hotel prices.
- User Interface (UI): While this project focuses on the model itself, the long-term scope includes the development of a user-friendly interface to allow non-technical users to input hotel attributes and receive price predictions.

This project aims to provide an automated, scalable, and data-driven solution for dynamic hotel pricing, helping stakeholders optimize revenue management strategies and gain competitive insights. It will serve as a tool to streamline pricing processes in the hospitality industry by incorporating AI/ML-driven predictions.



CHAPTER: 3 SOFTWARE AND HARDWARE REQUIREMENTS

CHAPTER 3 SOFTWARE AND HARDWARE REQUIREMENTS

Minimum Hardware Requirements

Component	Specification
Processor	Intel Core i3 / AMD Ryzen 3 (or equivalent)
RAM	4GB
HDD	30GB

Table 3.1 Minimum Hardware Requirements

Minimum Software Requirements

Component	Specification
Operating System	Windows, macOS, or Linux
Web Browser	Chrome, Firefox, Edge, or Safari (latest versions recommended)
Python	Python 3.7 or above
Machine Learning Libraries	Scikit-learn, TensorFlow/Keras, Pandas, NumPy (for model training and evaluation)
Database	MySQL (for storing hotel records and querying the data)
IDE/Notebook	Jupyter Notebook or any preferred IDE (e.g., VSCode) for model development and experimentation

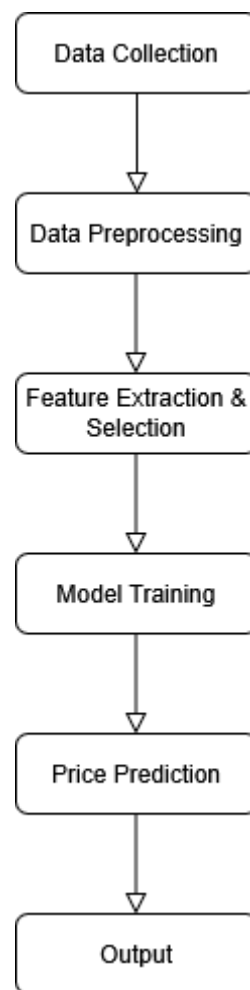
Table 3.2 Minimum Software Requirements

CHAPTER: 4 PROCESS MODE

CHAPTER 4 PROCESS MODEL

This project follows a structured workflow to ensure the accurate prediction of hotel prices based on various features. The system processes historical hotel data and user inputs, applies machine learning models, and generates price predictions. The workflow includes data collection, preprocessing, feature engineering, model training, and prediction. Below is a simplified diagram illustrating the process:

1. Data Collection: Hotel records are manually entered into MySQL database, including attributes like location, amenities, seasonality, and hotel type.
2. Data Preprocessing: The collected data is cleaned, and missing values are handled to ensure consistency.
3. Feature Extraction & Selection: Key features affecting hotel prices are identified and selected for training the machine learning models.
4. Model Training: Random Forest Regressor and Deep Neural Network models are trained using the preprocessed data to learn pricing patterns.
5. Price Prediction: When a user inputs hotel characteristics (such as location, amenities, and type), the trained models predict the optimal price based on the learned patterns.
6. Output: The predicted price is returned to the user as a result.



CHAPTER: 5 PROJECT PLAN

CHAPTER 5 PROJECT PLAN

5.1 List of Major Activities

1. **Research & Exploration** – Understanding the problem domain, exploring relevant machine learning models (Random Forest, Deep Neural Networks), and reviewing feature engineering techniques.
2. **Data Collection & Database Setup** – Manually collecting hotel data and setting up MySQL for storing hotel records.
3. **Data Preprocessing & Feature Engineering** – Cleaning the collected data, handling missing values, and selecting key features that affect hotel prices.
4. **Model Development & Training** – Training the Random Forest Regressor and Deep Neural Network models using the processed data.
5. **Model Evaluation & Tuning** – Evaluating model performance using appropriate metrics (e.g., RMSE, MAE) and fine-tuning hyperparameters.
6. **Backend Development (Optional)** – Developing the backend using FastAPI to handle user input and process predictions.
7. **Frontend Development (Optional)** – Designing a simple user interface where users can input hotel features and view the predicted price.
8. **System Integration (Optional)** – Integrating the machine learning models with the backend and connecting it to the frontend.
9. **Testing & Debugging (Optional)** – Ensuring the system's stability, testing the models' accuracy, and ensuring smooth user interactions.
10. **Final Deployment & Documentation (Optional)** – Deploying the system for use and preparing project reports and user documentation.

5.2 Estimated Time Duration in Days

Activity	Estimated Duration (Days)
Research & Exploration	10 Days
Data Collection & Database Setup	8 Days
Data Preprocessing & Feature Engineering	12 Days
Model Development & Training	14 Days
Model Evaluation & Tuning	10 Days
Backend Development (Optional)	12 Days
Frontend Development (Optional)	12 Days
System Integration (Optional)	8 Days
Testing & Debugging (Optional)	10 Days
Final Deployment & Documentation (Optional)	6 Days

Total Estimated Duration: ~2.5 Months

Table 5.1 Task Completion Estimated Time Duration in Days

CHAPTER: 6 IMPLEMENTATION DETAILS

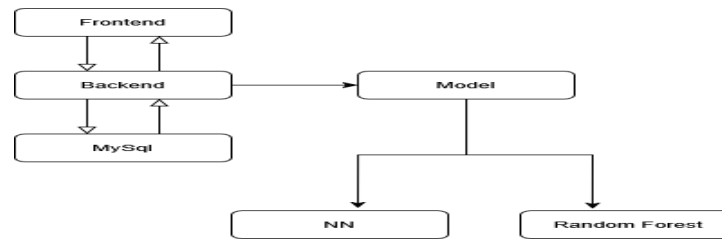
CHAPTER 6 IMPLEMENTATION DETAIL

This chapter provides an in-depth explanation of the implementation of the AI-based hotel price prediction system. It covers the system's architecture, workflow, integration of technologies, and a step-by-step breakdown of each module involved.

6.1 System Architecture

The system follows a client-server architecture, where the frontend, backend, and database communicate seamlessly to process hotel price predictions based on user inputs. The main components of the system include:

1. **Frontend (React.js):**
The frontend of the system is built using React.js. It provides a user-friendly interface where users can input various hotel attributes such as location, hotel type, amenities, and seasonality. The interface is designed to be intuitive, allowing users to interact with the system easily. After receiving the input, the frontend sends the data to the backend for processing and displays the predicted hotel price to the user.
2. **Backend (FastAPI):**
The backend is developed using FastAPI, which is used to handle user requests. It processes the input data sent from the frontend and passes it to the machine learning models for price prediction. FastAPI communicates with the trained machine learning models (Random Forest Regressor and Deep Neural Network) and retrieves the predicted prices. The backend also manages the interaction with the MySQL database, where the hotel data is stored.
3. **Machine Learning Models (Random Forest Regressor & Deep Neural Network):**
The system uses two models for price prediction:
 - **Random Forest Regressor:** An ensemble learning method that uses decision trees to predict the price based on historical data and selected features.
 - **Deep Neural Network (DNN):** A neural network model designed to learn complex patterns in the hotel pricing data and make predictions. The model was trained using a deep learning framework (e.g., TensorFlow or Keras).
4. **Database (MySQL):**
MySQL is used as the database to store hotel records, including attributes like location, hotel type, amenities, and pricing data. The database is pre-populated with hotel data that is used for training the models and making price predictions. The backend queries the database to retrieve relevant hotel data for input into the machine learning models.



6.2 Workflow Implementation

This section outlines the detailed workflow of the hotel price prediction system, covering each stage from user input to the final price prediction. It includes interactions between the frontend, backend, machine learning models, and database.

6.2.1 User Input & Frontend Processing (Planned/Optional)

- The user interacts with the **React-based frontend** by entering hotel-related features, such as:
 - **Location** (e.g., city or region)
 - **Hotel Type** (e.g., budget, luxury)
 - **Amenities** (e.g., free Wi-Fi, pool)
 - **Seasonality** (e.g., peak season or off-season)
- Once the user submits their input, the frontend sends this data to the **FastAPI backend** in the form of a structured API request.

6.2.2 Feature Extraction & Preprocessing (Data Processing in Backend)

- The backend receives the hotel feature data from the frontend and performs **data preprocessing** to ensure it is in a usable format for the machine learning models.
- The backend may handle tasks like:
 - **Handling missing values**: Ensuring that all necessary features are present and valid.
 - **Normalizing or encoding**: Converting categorical data (e.g., hotel type, location) into numerical representations if necessary.
- Preprocessed data is then passed to the **machine learning models** for price prediction.

6.2.3 Price Prediction via Machine Learning Models

- The **Random Forest Regressor** and **Deep Neural Network (DNN)** models, trained on historical hotel data, are used to predict the price.
 - **Random Forest Regressor**: An ensemble model that predicts hotel prices based on various features by aggregating predictions from decision trees.
 - **Deep Neural Network (DNN)**: A neural network model that can capture complex, non-linear relationships between hotel features and price.
- The backend sends the preprocessed input to the models, which generate the **predicted hotel price** based on the given features.

6.2.4 Query Execution & Result Fetching

- After receiving the predicted price, the backend prepares the data to be sent back to the frontend.
- The system queries the **MySQL database** (if needed) to fetch any related records or historical data that may help in validating the prediction.
- The predicted price and any supplementary data (if available) are packaged in a **structured format** (e.g., a JSON response) and sent back to the frontend.

6.2.5 Displaying Results to the User (Yet to be Integrated with Frontend)

- The **React frontend** receives the predicted price (and any relevant supplementary information) from the backend.
- The results are displayed to the user in a clear and readable format, such as:
 - A **predicted hotel price** displayed on the screen.
 - A **summary of the key features** (location, hotel type, etc.) influencing the price.
- If necessary, additional features (like graphs or visualizations) may be added to help the user understand the factors affecting the price.

6.3 Machine Learning Model Integration and Prediction Workflow

This section outlines how the machine learning models (specifically the **Random Forest Regressor** and **Deep Neural Network**) interact with **the** system to predict hotel prices. The models are trained to process the input features (such as location, hotel type, amenities) and generate price predictions. This section also explains the design of the prompt template, which ensures the models receive the correct input format for accurate predictions.

6.3.1 Feature Engineering Template Design

The feature engineering template plays a critical role in preparing the input data for the machine learning models. It defines the structure and format of the input data to ensure that the features are extracted and preprocessed correctly before being sent to the models. The template focuses on providing context about the data required for accurate predictions. It also ensures that:

- Categorical features (e.g., hotel type, amenities) are encoded appropriately.
- Numerical features (e.g., number of rooms, hotel rating) are normalized or scaled.
- Missing data is handled appropriately, either by imputation or exclusion.

The feature engineering template helps streamline **the process of transforming raw** user input **into a** format **that** the models can understand, allowing for more accurate price predictions.

6.3.2 Machine Learning Model Interaction

The **Random Forest Regressor** and **Deep Neural Network (DNN)** are responsible for generating hotel price predictions **based on the user's input**. The interaction between the system and the models is as follows:

- **User Input:** The user provides input on hotel-related features such as location, hotel type, and amenities.
- **Data Preprocessing:** The backend processes this input data based on the feature engineering template. The input is transformed into a numerical format that can be fed to the models.
- **Model Processing:** The preprocessed input is passed to the Random Forest Regressor or DNN for price prediction. The models generate a predicted hotel price based on the trained data.
- **Response Evaluation:** The predicted price is returned to the backend. The backend evaluates whether the prediction is valid, and any necessary adjustments or refinements are made before sending it to the frontend.

Once the models generate the price prediction, the backend sends the results back to the frontend, which displays them to the user in an understandable format.

6.4 Backend Implementation - FastAPI (Planned/Optional)

6.5 Frontend Implementation (React.js) (Planned/Optional)

6.6 Database Structure

Booking_ID	Unique Varchar
Room Type	Varchar
Availability	Integer
Date	Date
Holiday	Boolean
Price	Integer
Guest_Name	Varchar
Email	Varchar
Phone Number	Varchar

6.7 Conclusion

This chapter provides an overview of the current status of the hotel price prediction system and highlights the next steps required to complete the project. While some core components of the system, such as machine learning models (Random Forest Regressor and DNN), have been defined, the backend and frontend are still under development.

Key aspects of the project that remain to be developed include:

- **Backend Development:** The FastAPI-based backend, which will process user inputs, interact with the machine learning models, and manage database queries, is yet to be implemented.
- **Frontend Development:** The React-based user interface for user interaction and displaying price predictions is still in progress.
- **Error Handling:** The error handling module to validate predictions and ensure the system functions reliably needs to be developed.
- **Security Mechanisms:** Implementing role-based access control and securing the system from potential vulnerabilities will be done in the coming stages.
- **Cloud Deployment:** The system is currently running locally, and cloud deployment to ensure scalability and wider access will be a future task.

As the project progresses, more detailed documentation will be added, including UI screenshots, workflow diagrams, and backend response examples. This will help provide a clearer understanding of how each component functions and integrates with the system.



CHAPTER: 7 CONCLUSION AND FUTURE WORK

7.1 Conclusion

This project aims to leverage machine learning models to predict hotel prices based on user inputs, simplifying the pricing prediction process for users in the hospitality industry. By integrating machine learning models (Random Forest Regressor and Deep Neural Network) that can generate dynamic hotel price predictions based on a range of features, such as location, hotel type, and amenities.

So far, we have successfully established the machine learning models, set up the database for storing relevant hotel data, and created the initial framework for the backend and frontend. Although the backend and frontend development are ongoing, the key components of data preprocessing, feature engineering, and model training have been completed, demonstrating the feasibility of the approach.

The system, once fully developed, will provide a user-friendly interface where users can input hotel characteristics, and the system will predict hotel prices based on trained models. This will significantly streamline the pricing decision process for hotels, enabling users to make more informed, data-driven decisions.

7.2 Future Work

Moving forward, several critical steps will be undertaken to enhance and expand the system's capabilities:

- **Full-Stack Integration:** Connecting the **backend (FastAPI)** with the **frontend (React.js)** to enable real-time hotel price prediction. This will allow users to interact with the system and receive predictions based on their input features dynamically.
- **Dynamic Feature Handling:** Implementing a more dynamic system for handling input features and ensuring that only the relevant features (e.g., location, hotel type) are used for each prediction. This will optimize the prediction process and improve system flexibility.
- **Enhanced Error Handling and Security:** Improving the robustness of the system by adding **error handling** for invalid or incomplete inputs and implementing **role-based access control (RBAC)** to secure the application and restrict unauthorized access.
- **Model Optimization and Testing:** Exploring alternative machine learning models, such as **Gradient Boosting Machines (GBM)** or **XGBoost**, to compare their performance and accuracy with the current models. Fine-tuning the current models for improved prediction accuracy and efficiency is also planned.
- **User Testing and Interface Improvements:** Conducting thorough **user testing** to refine the frontend interface, ensuring that it is intuitive and user-friendly. Based on feedback, we will make adjustments to improve the overall user experience.
- **Documentation and Deployment:** Finalizing the system's **documentation** and preparing for **cloud deployment** to ensure scalability and easy access for a wider audience. This will involve transitioning the system from a local environment to a cloud platform for production use.

In conclusion, while significant progress has been made in developing the foundational components of the system, ongoing work will focus on full-stack integration, optimizing the machine learning models, and preparing the system for deployment and broader use. The successful completion of these steps will result in a robust, user-friendly platform capable of delivering accurate hotel price predictions.

CHAPTER: 8 REFERENCES

CHAPTER 8 REFERENCES

1. Random Forest Regressor - Scikit-learn Documentation
<https://scikit-learn.org/stable/modules/ensemble.html#random-forest>
2. Deep Learning for Regression Models - Towards Data Science
<https://towardsdatascience.com/deep-learning-for-regression-5e0d17b15657>
3. Pandas Documentation (read_sql method)
https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.read_sql.html
4. FastAPI Documentation
<https://fastapi.tiangolo.com/>
5. React.js Documentation
<https://reactjs.org/docs/getting-started.html>
6. Hotel Price Prediction with Machine Learning
<https://www.analyticsvidhya.com/blog/2021/10/a-complete-guide-to-hotel-price-prediction-with-machine-learning/>
7. Hotel Data Analysis with Python and Pandas
<https://www.kaggle.com/atharvaingle/hotel-price-prediction-using-machine-learning>

Industry Project Report

On

Sentiment Analysis

Developed By: -

Munish Patwa (21162121017)

Guided By:-

Prof. Sulabh Bhatt (Internal)

Rahul Bhagchandani(External)

Submitted to
Faculty of Engineering and Technology
Institute of Computer Technology
Ganpat University



Year - 2025



CERTIFICATE

This is to certify that the **Industry** Project work entitled “**Sentiment Analysis**” by Munish Patwa (Enrolment No. 21162121017) of Ganpat University, towards the partial fulfillment of requirements of the degree of Bachelor of Technology – Computer Science and Engineering, carried out by them in the AI Department at Ascion Global Partners. The results/findings contained in this Project have not been submitted in part or full to any other University / Institute for the award of any other Degree/Diploma.

Name & Signature of Internal Guide

Name & Signature of Head

Place: ICT - GUNI

Date: 22 February 2025

ACKNOWLEDGEMENT

Industry Internship project is a golden opportunity for learning and self-development. I consider myself very lucky and honored to have so many wonderful people lead me through in completion of this project. First and foremost, I would like to thank Dr. Rohit Patel, Principal, ICT, and Prof. Dharmesh Darji , Head, ICT who gave us an opportunity to undertake this project. My grateful thanks to Prof. Tejas Kadiya, Prof. Sulabh Bhatt, and Mr. Rahul Bhagchandani (Internal & External Guides) for their guidance in project work AI/ML Hotel Price Predictor, who despite being extraordinarily busy with academics, took time out to hear, guide and keep me on the correct path.

I do not know where I would have been without his/her help.

**MUNISH PATWA (Enrollment
No:21162121017)**

ABSTRACT

Sentiment analysis is a Natural Language Processing (NLP) technique used to determine the emotional tone of textual data. It plays a crucial role in understanding public opinion, customer feedback, and social media interactions. This project focuses on developing a sentiment analysis demo application that allows users to input review text and receive insights regarding sentiment polarity (positive, negative, or neutral) and key extracted keywords.

The system is built using a machine learning model trained on a Kaggle dataset. The trained model is integrated into a Flask-based web application, enabling real-time sentiment analysis. The application processes user input, analyzes the sentiment, and presents results in an intuitive and user-friendly manner. This project demonstrates the practical application of sentiment analysis in various domains, such as business intelligence, customer feedback evaluation, and brand reputation monitoring.

Through this project, we aim to showcase how sentiment analysis can be effectively implemented using machine learning techniques and deployed in a web-based environment for real-world applications

INDEX

1.	INTRODUCTION		7
2.	PROJECT SCOPE		9
3.	SOFTWARE AND HARDWARE REQUIREMENTS		11
4.	PROCESS MODEL		13
5.	PROJECT PLAN		17
	5.1	List of Major Activities	18
	5.2	Estimated Time Duration in Days	20
6.	IMPLEMENTATION DETAILS		21
	6.1	System Architecture	22
	6.2	Workflow Implementation	23
		6.2.1 User Input & Frontend Processing	23
		6.2.2 Text Preprocessing (Backend Processing)	23
		6.2.3 Sentiment Prediction Using Machine Learning Model	23
		6.2.4 Query Execution & Result Fetching	24
		6.2.5 Displaying Result	24
	6.3	Machine Learning Model Integration With Prediction Workflow	24
		6.3.1 Feature Engineering & Preprocessing	24
		6.3.2 Model Processing & Sentiment Prediction	25

		6.3.3	Response Evaluation & Data Processing	25
	6.4		Backend Implementation	25
	6.5		Frontend Implementation	25
	6.6		Conclusion	26
7.			CONCLUSION & FUTURE WORK	27
	7.1		Conclusion	28
	7.2		Future work	28
8.			REFERENCES	30

CHAPTER: 1 INTRODUCTION

CHAPTER 1 INTRODUCTION

In today's digital age, the vast amount of textual data generated through social media, product reviews, customer feedback, and online discussions has created an urgent need for effective sentiment analysis. Businesses, organizations, and researchers rely on sentiment analysis to extract meaningful insights from user-generated content, allowing them to understand public opinion, customer satisfaction, and market trends. Manually analyzing this data is time-consuming, inefficient, and prone to human bias. Therefore, leveraging artificial intelligence (AI) and machine learning (ML) for sentiment analysis provides a scalable and automated approach to processing and interpreting text data.

This project aims to develop a Sentiment Analysis Demo Application that enables users to input text reviews and obtain sentiment classification, keyword extraction, and relevant insights. The system utilizes a machine learning model trained on a Kaggle dataset to predict sentiment polarity (positive, negative, or neutral). The trained model is integrated into a Flask-based web application, providing an interactive platform where users can analyze textual data effortlessly.

The core of this project revolves around Natural Language Processing (NLP), which involves techniques for text preprocessing, feature extraction, and sentiment classification. The model is designed to capture sentiment-related features in textual data and make predictions based on learned patterns. The application can be useful for businesses looking to analyze customer feedback, brands monitoring social media sentiment, and researchers studying opinion trends.

Tools & Technologies Used in This Project:

- **Flask:** A lightweight web framework used to build the application's backend.
- **Python:** The primary programming language for model training, text processing, and API development.
- **Natural Language Toolkit (NLTK) & SpaCy:** NLP libraries used for text preprocessing, tokenization, stopwords removal, and lemmatization.
- **Scikit-learn:** Machine learning library used for training and evaluating sentiment classification models.
- **TensorFlow/Keras:** Frameworks used for implementing deep learning models to enhance sentiment analysis accuracy.
- **Pandas & NumPy:** Libraries used for data manipulation, analysis, and feature extraction.

- **Kaggle Dataset:** A real-world dataset used to train the sentiment analysis model.
- **Jupyter Notebook:** Environment for experimenting with data, testing models, and fine-tuning parameters.

CHAPTER: 2 PROJECT SCOPE

CHAPTER 2 PROJECT SCOPE

The scope of this project is to develop an AI-powered Sentiment Analysis Demo Application that utilizes machine learning and natural language processing (NLP) techniques to analyze user-generated text and classify it into sentiment categories. The system is designed to assist businesses, researchers, and analysts in extracting meaningful insights from textual data, such as customer reviews, social media comments, and feedback. By automating sentiment classification, the application eliminates the need for manual analysis and ensures accurate and consistent sentiment evaluation.

The primary goal of this project is to build a robust sentiment analysis model trained on a Kaggle dataset that can efficiently process user reviews and classify them as positive, negative, or neutral. The system is integrated into a Flask-based web application, enabling users to input text and receive instant sentiment analysis results. The project also focuses on enhancing sentiment detection through keyword extraction and NLP-based feature engineering to provide deeper insights into the text.

Key Components of the System:

- **Data Collection and Preprocessing:** The sentiment analysis model is trained using a dataset obtained from Kaggle. Preprocessing techniques such as tokenization, stopwords removal, stemming, and lemmatization are applied to clean and normalize the text data.
- **Machine Learning Model:** The project employs traditional machine learning models like Naïve Bayes, Logistic Regression, and Random Forest, as well as deep learning techniques using LSTMs or Transformers, to improve sentiment classification accuracy.
- **Feature Engineering:** NLP-based techniques, such as TF-IDF (Term Frequency-Inverse Document Frequency) and word embeddings, are used to extract meaningful features from the text, improving the model's ability to understand context.
- **Flask API for Deployment:** The trained model is deployed using Flask, allowing users to send text input via a web interface and receive real-time sentiment predictions.
- **User Interface (UI):** A simple and intuitive web-based UI enables users to input text, view sentiment classification results, and analyze extracted keywords.

This project aims to provide an automated, scalable, and user-friendly sentiment analysis solution, helping stakeholders gain actionable insights from textual data. The system is designed

to streamline sentiment evaluation, making it applicable in areas such as customer feedback analysis, social media monitoring, and product review sentiment tracking.

CHAPTER: 3 SOFTWARE AND HARDWARE REQUIREMENTS

CHAPTER 3 SOFTWARE AND HARDWARE REQUIREMENTS

Minimum Hardware Requirements

Component	Specification
Processor	Intel Core i3 / AMD Ryzen 3 (or equivalent)
RAM	4GB
HDD	30GB

Table 3.1 Minimum Hardware Requirements

Minimum Software Requirements

Component	Specification
Operating System	Windows, macOS, or Linux
Web Browser	Chrome, Firefox, Edge, or Safari (latest versions recommended)
Python	Python 3.7 or above
Machine Learning Libraries	Scikit-learn, TensorFlow/Keras, Pandas, NumPy (for model training and evaluation)
Database	MySQL (for storing hotel records and querying the data)
IDE/Notebook	Jupyter Notebook or any preferred IDE (e.g., VSCode) for model development and experimentation

Table 3.2 Minimum Software Requirements

CHAPTER: 4 PROCESS MODEL

CHAPTER 4 PROCESS MODEL

This project follows a structured workflow to ensure accurate sentiment analysis of textual data. The system processes user-provided text, applies Natural Language Processing (NLP) techniques, and utilizes a trained machine learning model to classify sentiment and extract key insights. The workflow includes data collection, preprocessing, feature extraction, model training, and sentiment prediction. Below is a step-by-step breakdown of the process:

1. Data Collection

The dataset used for training the sentiment analysis model is obtained from Kaggle, containing user reviews and their corresponding sentiment labels (positive, negative, or neutral). The dataset consists of a diverse set of text samples from different domains such as product reviews, customer feedback, and social media comments.

2. Data Preprocessing

Before training the model, the collected text data is cleaned and processed to enhance accuracy. The following preprocessing techniques are applied:

- Text Cleaning: Removing special characters, numbers, and unnecessary symbols.
- Tokenization: Splitting text into individual words or phrases.
- Stopword Removal: Eliminating commonly used words that do not add value to sentiment analysis (e.g., "is," "the," "and").
- Stemming & Lemmatization: Converting words to their root form to maintain consistency (e.g., "running" → "run").

3. Feature Extraction & Selection

To improve sentiment classification accuracy, key features are extracted using Natural Language Processing (NLP) techniques such as:

- TF-IDF (Term Frequency-Inverse Document Frequency): Determines the importance of words in a document relative to the entire dataset.
- Word Embeddings (Word2Vec, GloVe): Captures contextual meaning and relationships between words.
- N-grams: Identifies patterns in sequences of words to improve sentiment detection.

4. Model Training

The processed data is used to train machine learning models for sentiment classification. Different models are evaluated, including:

- Logistic Regression & Naïve Bayes: Traditional models for text classification.
- Random Forest & Support Vector Machines (SVM): More advanced models improving accuracy.
- Deep Learning (LSTMs, Transformer-based models like BERT): Enhances sentiment detection using contextual learning.

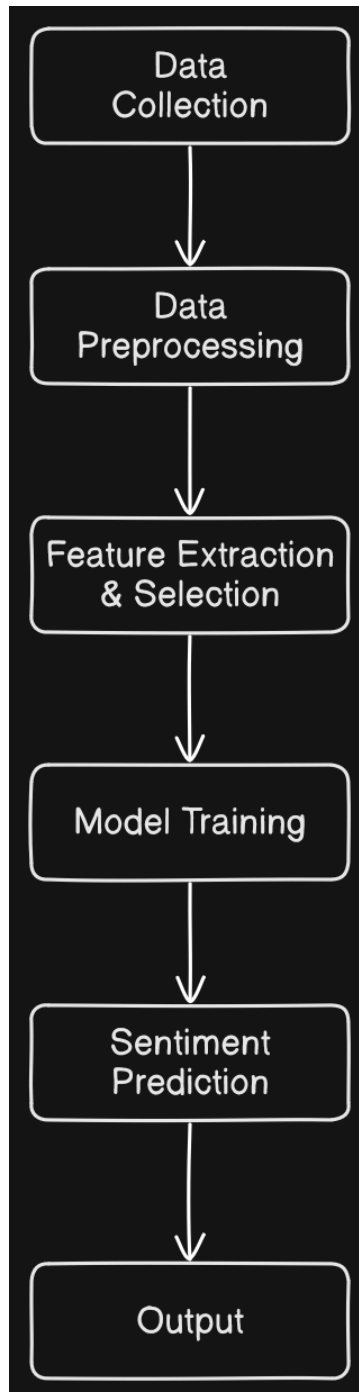
5. Sentiment Prediction

Once the model is trained, users can input new text data into the Flask-based web application. The model processes the text and predicts its sentiment category (positive, negative, or neutral), along with keyword extraction for deeper insights.

6. Output

The predicted sentiment and extracted keywords are displayed to the user, providing valuable insights into the emotional tone and key aspects of the text.

This structured workflow ensures an efficient and accurate sentiment analysis system, making it applicable for various industries, including customer feedback evaluation, brand sentiment monitoring, and social media trend analysis.



CHAPTER: 5 PROJECT PLAN

CHAPTER 5 PROJECT PLAN

5.1 List of Major Activities

The development of the **Sentiment Analysis Demo Application** follows a structured workflow consisting of several key phases. Each phase ensures systematic progress, from research and data collection to model development, testing, and deployment. Below is the list of major activities involved in the project:

1. Research & Exploration

Understanding the problem domain of sentiment analysis and its real-world applications. Exploring different machine learning and deep learning models for sentiment classification (e.g., Naïve Bayes, Random Forest, LSTMs, BERT). Reviewing **Natural Language Processing (NLP) techniques** for text processing and feature extraction.

2. Data Collection & Preparation

Obtaining a sentiment-labeled dataset from **Kaggle** or other sources. Cleaning and normalizing text data by applying **preprocessing techniques** (removing stopwords, stemming, lemmatization, etc.). Splitting the dataset into **training, validation, and test sets** for model evaluation.

3. Feature Engineering & Selection

Extracting relevant text features using **TF-IDF, word embeddings (Word2Vec, GloVe), and N-grams**. Identifying and selecting key linguistic patterns that influence sentiment classification.

4. Model Development & Training

Implementing different **machine learning models** (Logistic Regression, Naïve Bayes, Random Forest, SVM). Developing and training **deep learning models** (LSTMs, Transformer-based

models like BERT). Comparing models to select the most **accurate and efficient** approach.

5. Model Evaluation & Optimization

Evaluating model performance using metrics such as **Accuracy, Precision, Recall, and F1-score**. Fine-tuning hyperparameters to **improve accuracy and reduce overfitting**. Testing model generalization across **different datasets**.

6. Backend Development *(Optional, if required)*

Implementing a **Flask-based API** to handle text input and return sentiment predictions. Ensuring seamless integration with the trained **machine learning model**.

7. Frontend Development *(Optional, if required)*

Designing a **simple user interface** where users can input text and view sentiment analysis results. Displaying **extracted keywords** and **sentiment classification** in an intuitive manner.

8. System Integration & Testing *(Optional, if required)*

Connecting the **frontend, backend, and machine learning model** for a seamless user experience. Conducting extensive testing to ensure stability, accuracy, and performance.

9. Final Deployment & Documentation

Deploying the system on a suitable platform (e.g., **Heroku, AWS, or a local server**). Preparing project reports, **technical documentation, and user guides**.

5.2 Estimated Time Duration in Days

Activity	Estimated Duration (Days)
Research & Exploration	10 Days
Data Collection & Database Setup	8 Days
Data Preprocessing & Feature Engineering	12 Days
Model Development & Training	14 Days
Model Evaluation & Tuning	10 Days
Backend Development (Optional)	12 Days
Frontend Development (Optional)	12 Days
System Integration (Optional)	8 Days
Testing & Debugging (Optional)	10 Days
Final Deployment & Documentation (Optional)	6 Days

Total Estimated Duration: ~2.5 Months

Table 5.1 Task Completion Estimated Time Duration in Days

CHAPTER: 6 IMPLEMENTATION DETAILS

CHAPTER 6 IMPLEMENTATION DETAIL

This chapter provides an in-depth explanation of the implementation of the Sentiment Analysis Demo Application. It covers the system architecture, workflow, integration of technologies, and a step-by-step breakdown of each module involved.

6.1 System Architecture

The Sentiment Analysis system follows a client-server architecture, where the frontend, backend, and machine learning model work together to analyze user reviews and determine their sentiment. The main components of the system include:

Frontend (React.js)

- The frontend is developed using React.js, providing a simple and interactive user interface.
- Users enter text-based reviews, which are then sent to the backend for sentiment analysis.
- The results (sentiment score and keywords) are displayed to the user.

Backend (Flask API)

- The backend is implemented using Flask, which handles user requests and model predictions.
- It receives input from the frontend and preprocesses the text before passing it to the trained machine learning model.

- The backend returns sentiment analysis results to the frontend.

Machine Learning Model (Trained Sentiment Model)

- A pretrained sentiment analysis model (such as a Naïve Bayes Classifier, LSTM, or Transformer-based model) is used.
- It classifies the input review into positive, negative, or neutral sentiment and extracts important keywords.

6.2 Workflow Implementation

The sentiment analysis workflow involves multiple stages, from user input to model prediction and result display.

6.2.1 User Input & Frontend Processing

- The user enters a text review into the React-based frontend.
- The input is structured into a JSON request and sent to the Flask backend through an API call.

6.2.2 Text Preprocessing (Backend Processing)

- The backend receives the text and performs preprocessing, which includes:
 - Removing stopwords, special characters, and punctuation.
 - Converting the text to lowercase.
 - Tokenization and lemmatization.

- The cleaned text is then passed to the sentiment analysis model.

6.2.3 Sentiment Prediction Using Machine Learning Model

- The trained sentiment model analyzes the input text and predicts whether the sentiment is positive, negative, or neutral.
- The model also extracts key terms that influenced the sentiment classification.

6.2.4 Query Execution & Result Fetching

- The backend processes the model's output and formats it into a structured JSON response.
- If necessary, the backend may fetch additional reference data (e.g., predefined sentiment scores for certain words).

6.2.5 Displaying Results to the User

- The frontend receives the sentiment prediction and displays:
 - The sentiment category (Positive/Negative/Neutral).
 - Confidence score of the prediction.
 - Important keywords that influenced the sentiment.

6.3 Machine Learning Model Integration and Prediction Workflow

This section describes how the machine learning model interacts with the backend and frontend to provide sentiment predictions.

6.3.1 Feature Engineering & Preprocessing

- The raw text input undergoes preprocessing to ensure better accuracy.

- Text cleaning techniques (tokenization, lemmatization, and stopwords removal) are applied.
- TF-IDF, word embeddings (Word2Vec, GloVe), or transformer embeddings may be used to convert text into a numerical format.

6.3.2 Model Processing & Sentiment Prediction

- The preprocessed text is passed into the trained sentiment analysis model.
- The model predicts:
 - Sentiment label (Positive/Negative/Neutral).
 - Confidence score (e.g., 85% positive sentiment).
 - Top keywords contributing to the sentiment.

6.3.3 Response Evaluation & Data Processing

- The backend formats the prediction output into a structured response.
- The final result is returned to the frontend.

6.4 Backend Implementation - Flask API

- Handles API requests from the frontend.
- Processes text input and forwards it to the model.
- Returns predictions and extracted keywords.

6.5 Frontend Implementation - React.js

- Captures user input (text reviews).
- Displays sentiment results in a user-friendly format.

6.6 Conclusion

This chapter provides a comprehensive overview of the implementation details of the Sentiment Analysis Demo Application, highlighting the current progress and outlining the next steps required to complete the project. While key components, such as the sentiment analysis model and preprocessing pipeline, have been implemented, several aspects of the system are still under development.

Key Areas of Development

The following components require further work to achieve full functionality:

- **Backend Development:** The Flask-based backend, responsible for handling user inputs, processing text, and interacting with the machine learning model, is still being refined. Additional API endpoints for improved data handling may be introduced.
- **Frontend Development:** The React-based user interface, which enables users to input text and view sentiment analysis results, is in progress and will be further optimized for better user experience.
- **Error Handling & Robustness:** Implementing comprehensive error handling to manage invalid inputs, API failures, and unexpected outputs from the model is a key next step.
- **Performance Optimization:** Enhancing the speed of model inference, API response times, and frontend rendering will be addressed to ensure seamless user interactions.

- Security Enhancements: Implementing data validation, input sanitization, and API security measures to prevent vulnerabilities.
- Cloud Deployment: Currently, the system runs locally, and future deployment to a cloud platform (e.g., AWS, GCP, or Heroku) will be considered for scalability and accessibility.

CHAPTER: 7 CONCLUSION AND FUTURE WORK

7.1 Conclusion

This project aims to develop a Sentiment Analysis Demo Application that leverages machine learning techniques to analyze textual data and determine sentiment polarity. The system integrates Natural Language Processing (NLP) models with a user-friendly web interface, allowing users to input text and receive sentiment predictions along with relevant insights.

So far, we have successfully:

- Developed and trained a sentiment analysis model using a labeled dataset from Kaggle.
- Implemented data preprocessing techniques, including text cleaning, tokenization, and vectorization.
- Designed an initial Flask backend to process user input, pass it through the model, and return predictions.
- Created the basic frontend framework using React for user interaction.

Although full-stack integration is still in progress, the key machine learning components—model training, evaluation, and prediction workflow—have been successfully implemented, proving the feasibility of the approach. Once fully developed, the system will provide a seamless and efficient tool for real-time sentiment analysis, benefiting users in various domains such as customer feedback analysis, social media monitoring, and business insights.

7.2 Future Work

To enhance the Sentiment Analysis Application, several key improvements and expansions are planned:

1. Full-Stack Integration:
 - Establish seamless communication between the React frontend and the Flask backend using REST APIs.
 - Enable real-time sentiment analysis, allowing users to input text dynamically and receive instant feedback.
2. Advanced Feature Extraction & Model Enhancement:
 - Implement more advanced NLP techniques, such as word embeddings (Word2Vec, GloVe, or BERT), to improve model accuracy.
 - Experiment with alternative models, including transformer-based models (BERT, RoBERTa), to compare their performance with the current approach.
3. Improved Error Handling & Security Measures:
 - Add input validation to handle missing or inappropriate text inputs.
 - Implement role-based access control (RBAC) to restrict unauthorized access.
4. User Interface Optimization & Testing:
 - Conduct user testing to gather feedback and refine the UI for a more intuitive experience.
 - Implement visual elements (e.g., graphs, confidence scores) to improve data interpretation.
5. System Deployment & Scalability:
 - Deploy the application to a cloud platform (AWS, Heroku, or GCP) for accessibility and scalability.

- Optimize backend performance to handle high volumes of user requests efficiently.

CHAPTER: 8 REFERENCES

1. Pandas Documentation (read_sql method)
https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.read_sql.html
2. FastAPI Documentation
<https://fastapi.tiangolo.com/>
3. React.js Documentation
<https://reactjs.org/docs/getting-started.html>
4. Dataset
<https://www.kaggle.com/datasets/andrewmvd/trip-advisor-hotel-reviews>
5. Flask Documentation
<https://flask.palletsprojects.com/en/stable/>

Munish Patwa

SENTIMENT_Industry_Project_Report_(Review_2)-1.pdf



Assignment 7



Research_1



Ganpat University

Document Details

Submission ID

trn:oid:::1:3238895406

Submission Date

May 4, 2025, 11:31 PM GMT+5:30

Download Date

May 4, 2025, 11:38 PM GMT+5:30

File Name

SENTIMENT_Industry_Project_Report_Review_2_-1.pdf

File Size

447.4 KB

27 Pages





3,008 Words

18,598 Characters




10% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Match Groups

-  **31 Not Cited or Quoted 10%**
Matches with neither in-text citation nor quotation marks
-  **0 Missing Quotations 0%**
Matches that are still very similar to source material
-  **0 Missing Citation 0%**
Matches that have quotation marks, but no in-text citation
-  **0 Cited and Quoted 0%**
Matches with in-text citation present, but no quotation marks

Top Sources

- 7%  Internet sources
- 5%  Publications
- 0%  Submitted works (Student Papers)

Match Groups

- 31 Not Cited or Quoted 10%**
Matches with neither in-text citation nor quotation marks
- 0 Missing Quotations 0%**
Matches that are still very similar to source material
- 0 Missing Citation 0%**
Matches that have quotation marks, but no in-text citation
- 0 Cited and Quoted 0%**
Matches with in-text citation present, but no quotation marks

Top Sources

- 7% Internet sources
- 5% Publications
- 0% Submitted works (Student Papers)

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	Publication	Swati Raj, Jhalak Dutta, Saikat Bandopadhyay, Rajesh Prasad. "Chapter 18 Featur...	1%
2	Publication	V. Sharmila, S. Kannadhasan, A. Rajiv Kannan, P. Sivakumar, V. Vennila. "Challeng...	1%
3	Internet	www.jetir.org	<1%
4	Internet	fastercapital.com	<1%
5	Publication	Narasimha Rao Vajjhala, Kenneth David Strang. "Cybersecurity in Knowledge Ma...	<1%
6	Internet	github.com	<1%
7	Internet	www.readkong.com	<1%
8	Internet	interoncof.com	<1%
9	Publication	Howard Anderson, Sharon Yull, Bruce Hellingsworth. "Higher National Computin...	<1%
10	Internet	repository.tudelft.nl	<1%

11	Internet	www.coursehero.com	<1%
12	Internet	core.ac.uk	<1%
13	Internet	simeononsecurity.ch	<1%
14	Internet	store.steampowered.com	<1%
15	Publication	Beksultan Sagyndyk, Dilyara Baymurzina, Mikhail Burtsev. "Chapter 39 DeepPavl...	<1%
16	Internet	epon.org	<1%
17	Internet	medium.com	<1%
18	Internet	www.ijeat.org	<1%
19	Internet	www.nerc.com	<1%
20	Publication	"Evolutionary Artificial Intelligence", Springer Science and Business Media LLC, 20...	<1%
21	Internet	platforce.io	<1%
22	Publication	H.L. Gururaj, Francesco Flammini, J. Shreyas. "Data Science & Exploration in Artifi...	<1%

INDEX

1.	INTRODUCTION		7
2.	PROJECT SCOPE		9
3.	SOFTWARE AND HARDWARE REQUIREMENTS		11
4.	PROCESS MODEL		13
5.	PROJECT PLAN		17
	5.1	List of Major Activities	18
	5.2	Estimated Time Duration in Days	20
6.	IMPLEMENTATION DETAILS		21
	6.1	System Architecture	22
	6.2	Workflow Implementation	23
		6.2.1 User Input & Frontend Processing	23
		6.2.2 Text Preprocessing (Backend Processing)	23
		6.2.3 Sentiment Prediction Using Machine Learning Model	23
		6.2.4 Query Execution & Result Fetching	24
		6.2.5 Displaying Result	24
	6.3	Machine Learning Model Integration With Prediction Workflow	24
		6.3.1 Feature Engineering & Preprocessing	24
		6.3.2 Model Processing & Sentiment Prediction	25
		6.3.3 Response Evaluation & Data Processing	25
	6.4	Backend Implementation	25

	6.5	Frontend Implementation	25
	6.6	Conclusion	26
7.	CONCLUSION & FUTURE WORK		27
	7.1	Conclusion	28
	7.2	Future work	28
8.	REFERENCES		30



CHAPTER: 1 INTRODUCTION

CHAPTER 1 INTRODUCTION

In today's digital age, the vast amount of textual data generated through social media, product reviews, customer feedback, and online discussions has created an urgent need for effective sentiment analysis. Businesses, organizations, and researchers rely on sentiment analysis to extract meaningful insights from user-generated content, allowing them to understand public opinion, customer satisfaction, and market trends. Manually analyzing this data is time-consuming, inefficient, and prone to human bias. Therefore, leveraging artificial intelligence (AI) and machine learning (ML) for sentiment analysis provides a scalable & automated approach to processing and interpreting text data.

This project's goal is to develop a Sentiment Analysis Demo Application that enables users to input text reviews and obtain sentiment classification, keyword extraction, and relevant insights. The system utilizes a ML model trained on a Kaggle dataset to predict sentiment polarity (positive, negative, or neutral). The trained model is unified into a Flask-based web application, providing an interactive platform where users can analyze textual data effortlessly.

The core of this project revolves around Natural Language Processing (NLP), which involves techniques for text preprocessing, sentiment classification and feature extraction. The model is designed to capture sentiment-related features in textual data and make predictions according to the learned patterns. The application can be useful for businesses looking to analyze customer feedback, brands monitoring social media sentiment, and researchers studying opinion trends.

Tools & Technologies Used in This Project:

- **Flask:** A lightweight web framework used to build the application's backend.
- **Python:** The primary programming language for model training, text processing, and API development.
- **Natural Language Toolkit (NLTK) & SpaCy:** NLP libraries used for text preprocessing, tokenization, stopword removal, and lemmatization.
- **Scikit-learn:** Machine learning library used for training and evaluating sentiment classification models.
- **TensorFlow/Keras:** Frameworks used for implementing deep learning models to enhance sentiment analysis accuracy.
- **Pandas & NumPy:** Libraries used for data manipulation, analysis, and feature extraction.
- **Kaggle Dataset:** A real-world dataset used to train the sentiment analysis model.
- **Jupyter Notebook:** Environment for experimenting with data, testing models, and fine-tuning parameters.

CHAPTER: 2 PROJECT SCOPE

CHAPTER 2 PROJECT SCOPE

The scope of this project is to create an AI-powered Sentiment Analysis Demo Application that utilizes ML and (NLP) techniques to analyze user-generated text and classify it into sentiment categories. The system is designed to assist businesses, researchers, and analysts in extracting meaningful information from textual data, such as customer reviews, social media comments, and feedback. By automating sentiment classification, the application eliminates the need for manual analysis and ensures accurate and consistent sentiment evaluation.

The ultimate goal of this model is to construct a robust sentiment analysis model trained on a Kaggle dataset that can efficiently process user reviews and classify them as negative, positive or neutral. The system is integrated into a Flask-based web application, enabling users to input text and receive instantaneous results. The project also focuses on enhancing sentiment detection through keyword extraction and NLP-based feature engineering to provide deeper insights into the text.

Key Components of the System:

- **Data Collection and Preprocessing:** The sentiment analysis model is trained using a dataset obtained from Kaggle. Preprocessing techniques such as tokenization, stopwords removal, stemming, and lemmatization are applied to clean and normalize the text data.
- **Machine Learning Model:** The project employs traditional ML models like Random Forest, Logistic Regression and Naïve Bayes as well as deep learning techniques using LSTMs or Transformers, to improve sentiment classification accuracy.
- **Feature Engineering:** NLP-based techniques, such as TF-IDF (Term Frequency-Inverse Document Frequency) and word embeddings, are utilized to extract valuable features out of the text, improving the model's ability to understand context.
- **Flask API for Deployment:** The trained model is deployed using Flask, allowing users to send text input via a web interface and receive real-time sentiment predictions.
- **User Interface (UI):** A simple and intuitive web-based UI enables users to enter text, view sentiment classification results, and analyze extracted keywords.

This project aims to provide an automated, scalable, and user-friendly sentiment analysis solution, helping stakeholders gain actionable insights from textual data. The system is designed to streamline sentiment evaluation, making it applicable in areas such as product review sentiment tracking, social media monitoring and customer feedback analysis.



CHAPTER: 3 SOFTWARE AND HARDWARE REQUIREMENTS



CHAPTER 3 SOFTWARE AND HARDWARE REQUIREMENTS

Minimum Hardware Requirements

Component	Specification
Processor	Intel Core i3 / AMD Ryzen 3 (or equivalent)
RAM	4GB
HDD	30GB

Table 3.1 Minimum Hardware Requirements

Minimum Software Requirements

Component	Specification
Operating System	Windows, macOS, or Linux
Web Browser	Chrome, Firefox, Edge, or Safari (latest versions recommended)
Python	Python 3.7 or above
Machine Learning Libraries	Scikit-learn, TensorFlow/Keras, Pandas, NumPy (for model training and evaluation)
Database	MySQL (for storing hotel records and querying the data)
IDE/Notebook	Jupyter Notebook or any preferred IDE (e.g., VSCode) for model development and experimentation

Table 3.2 Minimum Software Requirements

CHAPTER: 4 PROCESS MODEL

CHAPTER 4 PROCESS MODEL

This project follows a structured workflow to ensure accurate sentiment analysis of textual data. The system processes user-provided text, applies NLP techniques, and utilizes a trained machine learning model to classify sentiment and extract key insights. The workflow includes data collection, preprocessing, feature extraction, model training, and sentiment prediction. Below is a step-by-step breakdown of the process:

1. Data Collection

The dataset used for training the sentiment analysis model is obtained from Kaggle, containing user reviews and their corresponding sentiment labels (negative, positive or neutral). The dataset consists of a diverse set of text samples from different domains such as product reviews, social media comments and customer feedback

2. Data Preprocessing

Before training the model, the collected text data is cleaned and processed to enhance accuracy. The following preprocessing techniques are applied:

- Text Cleaning: Removing special characters, numbers, and unnecessary symbols.
- Tokenization: Dividing text into individual words or phrases.
- Stopword Removal: Eliminating commonly used words that do not add innate value to sentiment analysis (e.g., "is," "the," "and").
- Stemming & Lemmatization: Transforming words to their original form to maintain consistency (e.g., "running" → "run").

3. Feature Extraction & Selection

To improve sentiment classification accuracy, key features are extracted using (NLP) techniques such as:

- TF-IDF (Term Frequency-Inverse Document Frequency): Decides the importance/value of words in a document relative to the entire dataset.
- Word Embeddings (Word2Vec, GloVe): Captures contextual meaning and relationships between words.

- N-grams: Identifies patterns in sequences of words to improve sentiment detection.

4. Model Training

The processed data is used to train ML models for sentiment classification. Different models are evaluated, including:

- Logistic Regression & Naïve Bayes: Traditional models for text classification.
- Random Forest & Support Vector Machines (SVM): More advanced models improving accuracy.
- Deep Learning (LSTMs, Transformer-based models like BERT): Enhances sentiment detection using contextual learning.

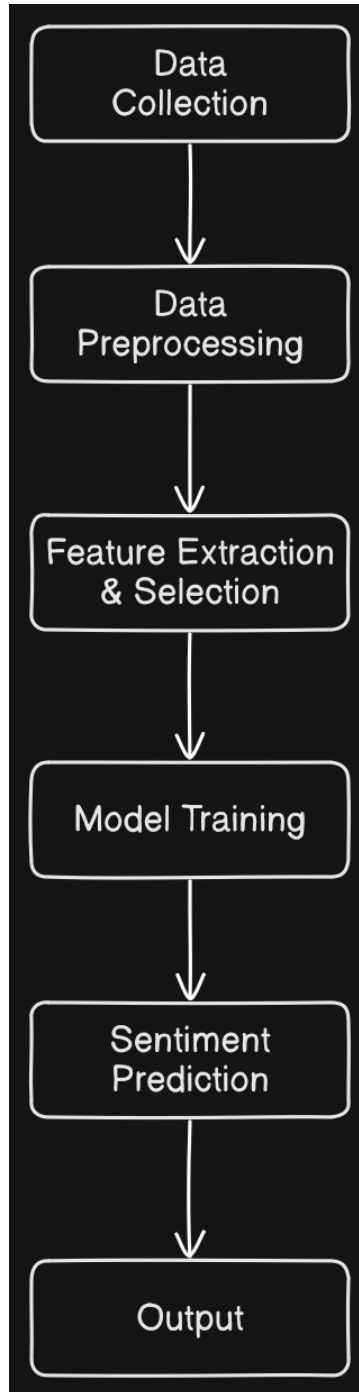
5. Sentiment Prediction

Once the model is trained, users can enter text data into the Flask-based web application. The model processes the text and predicts its sentiment category (positive, negative, or neutral), along with keyword extraction for deeper insights.

6. Output

The predicted sentiment and extracted keywords are displayed to the user, providing valuable insights into the emotional tone and key aspects of the text.

This structured workflow ensures an efficient and accurate sentiment analysis system, making it applicable for various industries, including customer feedback evaluation, brand sentiment monitoring, and social media trend analysis.



CHAPTER: 5 PROJECT PLAN

CHAPTER 5 PROJECT PLAN

5.1 List of Major Activities

The development of the **Sentiment Analysis Demo Application** follows a structured workflow consisting of several key phases. Each phase ensures systematic progress, from research and data collection to model development, testing, and deployment. Below is the list of major activities involved in the project:

1. Research & Exploration

Understanding the problem domain of sentiment analysis and its real-world applications. Exploring different deep learning and machine learning models for sentiment classification (e.g., Naïve Bayes, Random Forest, LSTMs, BERT). Reviewing **NLP techniques** for text processing and feature extraction.

2. Data Collection & Preparation

Obtaining a sentiment-labeled dataset from **Kaggle** or other sources. Cleaning and normalizing text data by applying **preprocessing techniques** (removing stopwords, stemming, lemmatization, etc.). Splitting **the dataset into validation, training and test sets** for **model** evaluation.

3. Feature Engineering & Selection

Extracting relevant text features using **TF-IDF, word embeddings (Word2Vec, GloVe), and N-grams**. Identifying and selecting key linguistic patterns that influence sentiment classification.

4. Model Development & Training

Implementing different **ML models** (**Logistic Regression, Naïve Bayes, Random Forest, SVM**). Developing and training **deep learning models** (LSTMs, Transformer-based models like BERT). Comparing models to select the most **precise and efficient** approach.

5. Model Evaluation & Optimization

Evaluating model performance employing metrics such as **Recall, Precision, Accuracy and F1-score**. Fine-tuning hyperparameters to improve accuracy and reduce overfitting. Testing model generalization across different datasets.

6. Backend Development (*Optional, if required*)

Implementing a **Flask-based API** to handle text input and return sentiment predictions. Ensuring seamless integration with the trained **machine learning model**.

7. Frontend Development (*Optional, if required*)

Designing a **simple UI** where users can manually input text and view sentiment analysis results. Displaying **extracted keywords** and **sentiment classification** in an intuitive manner.

8. System Integration & Testing (*Optional, if required*)

Connecting the **frontend, backend, and machine learning model** for a seamless user experience. Conducting extensive testing to ensure stability, accuracy, and performance.

9. Final Deployment & Documentation

Deploying the system on a suitable platform (e.g., **Heroku, AWS, or a local server**). Preparing project reports, **technical documentation, and user guides**.

5.2 Estimated Time Duration in Days

Activity	Estimated Duration (Days)
Research & Exploration	10 Days
Data Collection & Database Setup	8 Days
Data Preprocessing & Feature Engineering	12 Days
Model Development & Training	14 Days
Model Evaluation & Tuning	10 Days
Backend Development (Optional)	12 Days
Frontend Development (Optional)	12 Days
System Integration (Optional)	8 Days
Testing & Debugging (Optional)	10 Days
Final Deployment & Documentation (Optional)	6 Days

Total Estimated Duration: ~2.5 Months

Table 5.1 Task Completion Estimated Time Duration in Days

CHAPTER: 6 IMPLEMENTATION DETAILS

CHAPTER 6 IMPLEMENTATION DETAIL

This chapter provides an in-depth explanation of the implementation of the Sentiment Analysis Demo Application. It covers the system architecture, workflow, integration of technologies, and a step-by-step breakdown of each module involved.

6.1 System Architecture

The Sentiment Analysis system follows a client-server architecture, where the frontend, backend, and machine learning model work together to analyze user reviews and determine their sentiment. The main components of the system include:

Frontend (React.js)

- The frontend is developed using React.js, providing a simple and interactive user interface.
- Users enter text-based reviews, which are then sent to the backend for sentiment analysis.
- The results (sentiment score and keywords) are displayed to the user.

Backend (Flask API)

- The backend is implemented using Flask, which handles user requests and model predictions.
- It receives input from the frontend and preprocesses the text before passing it to the trained ML model.
- The backend returns sentiment analysis results to the frontend.

Machine Learning Model (Trained Sentiment Model)

- A pretrained sentiment analysis model (such as a Naïve Bayes Classifier, LSTM, or Transformer-based model) is used.
- It classifies the input review into positive, negative, or neutral sentiment and extracts important keywords.

6.2 Workflow Implementation

The sentiment analysis workflow involves multiple stages, from user input to model prediction and result display.

6.2.1 User Input & Frontend Processing

- The user enters a text review into the React-based frontend.
- The input is structured into a JSON request and sent to the Flask backend through an API call.

6.2.2 Text Preprocessing (Backend Processing)

- The backend receives the text and performs preprocessing, which includes:
 - Removing stopwords, special characters, and punctuation.
 - Converting the text to lowercase.
 - Tokenization and lemmatization.
- The cleaned text is then passed to the sentiment analysis model.

6.2.3 Sentiment Prediction Using Machine Learning Model

- The trained sentiment model analyzes the input text and predicts whether the sentiment is negative, positive or neutral.
- The model also extracts key terms that influenced the sentiment classification.

6.2.4 Query Execution & Result Fetching

- The backend processes the model's output and formats it into a structured JSON response.
- If necessary, the backend may fetch additional reference data (e.g., predefined sentiment scores for certain words).

6.2.5 Displaying Results to the User

- The frontend receives the sentiment prediction and displays:
 - The sentiment category (Positive/Negative/Neutral).
 - Confidence score of the prediction.
 - Important keywords that influenced the sentiment.

6.3 Machine Learning Model Integration and Prediction Workflow

This section describes how the machine learning model interacts with the backend and frontend to provide sentiment predictions.

6.3.1 Feature Engineering & Preprocessing

- The raw text input undergoes preprocessing to ensure better accuracy.
- Text cleaning techniques (tokenization, lemmatization, and stopword removal) are applied.
- TF-IDF, word embeddings (Word2Vec, GloVe), or transformer embeddings may be used to convert text into a numerical format.

6.3.2 Model Processing & Sentiment Prediction

- The preprocessed text is passed into the trained sentiment analysis model.
- The model predicts:
 - Sentiment label (Positive/Negative/Neutral).
 - Confidence score (e.g., 85% positive sentiment).
 - Top keywords contributing to the sentiment.

6.3.3 Response Evaluation & Data Processing

- The backend formats the prediction output into a structured response.
- The final result is returned to the frontend.

6.4 Backend Implementation - Flask API

- Handles API requests from the frontend.
- Processes text input and forwards it to the model.
- Returns predictions and extracted keywords.

6.5 Frontend Implementation - React.js

- Captures user input (text reviews).
- Displays sentiment results in a user-friendly format.

6.6 Conclusion

This chapter provides a comprehensive overview of the implementation details of the Sentiment Analysis Demo Application, highlighting the current progress and outlining the next steps required to complete the project. While key components, such as the sentiment analysis model and preprocessing pipeline, have been implemented, several aspects of the system are still under development.

Key Areas of Development

The following components require further work to achieve full functionality:

- **Backend Development:** The Flask-based backend, responsible for handling user inputs, processing text, and interacting with the machine learning model, is still being refined. Additional API endpoints for improved data handling may be introduced.
- **Frontend Development:** The React-based user interface, which enables users to input text and view sentiment analysis results, is in progress and will be further optimized for better user experience.
- **Error Handling & Robustness:** Implementing comprehensive error handling to manage invalid inputs, API failures, and unexpected outputs from the model is a key next step.
- **Performance Optimization:** Enhancing the speed of model inference, API response times, and frontend rendering will be addressed to ensure seamless user interactions.
- **Security Enhancements:** Implementing data validation, input sanitization, and API security measures to prevent vulnerabilities.
- **Cloud Deployment:** Currently, the system runs locally, and future deployment to a cloud platform (e.g., AWS, GCP, or Heroku) will be considered for scalability and accessibility.



CHAPTER: 7 CONCLUSION AND FUTURE WORK

7.1 Conclusion

This project aims to develop a Sentiment Analysis Demo Application that leverages machine learning techniques to analyze textual data and determine sentiment polarity. The system integrates Natural Language Processing (NLP) models with a user-friendly web interface, allowing users to input text and receive sentiment predictions along with relevant insights.

So far, we have successfully:

- Developed and trained a sentiment analysis model using a labeled dataset from Kaggle.
- Implemented data preprocessing techniques, including text cleaning, tokenization, and vectorization.
- Designed an initial Flask backend to process user input, pass it through the model, and return predictions.
- Created the basic frontend framework using React for user interaction.

Although full-stack integration is still in progress, the key machine learning components—model training, evaluation, and prediction workflow—have been successfully implemented, proving the feasibility of the approach. Once fully developed, the system will provide a seamless and efficient tool for real-time sentiment analysis, benefiting users in various domains such as customer feedback analysis, social media monitoring, and business insights.

7.2 Future Work

To enhance the Sentiment Analysis Application, several key improvements and expansions are planned:

1. Full-Stack Integration:
 - Establish seamless communication between the React frontend and the Flask backend using REST APIs.
 - Enable real-time sentiment analysis, allowing users to input text dynamically and receive instant feedback.
2. Advanced Feature Extraction & Model Enhancement:

5

- Implement more advanced NLP techniques, such as word embeddings (Word2Vec, GloVe, or BERT), to improve model accuracy.
- Experiment with alternative models, including transformer-based models (BERT, RoBERTa), to compare their performance with the current approach.

3. Improved Error Handling & Security Measures:

13

- Add input validation to handle missing or inappropriate text inputs.
- Implement role-based access control (RBAC) to restrict unauthorized access.

4. User Interface Optimization & Testing:

- Conduct user testing to gather feedback and refine the UI for a more intuitive experience.
- Implement visual elements (e.g., graphs, confidence scores) to improve data interpretation.

5. System Deployment & Scalability:

- Deploy the application to a cloud platform (AWS, Heroku, or GCP) for accessibility and scalability.
- Optimize backend performance to handle high volumes of user requests efficiently.

CHAPTER: 8 REFERENCES

1. Pandas Documentation (`read_sql` method)
https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.read_sql.html
2. FastAPI Documentation
<https://fastapi.tiangolo.com/>
3. React.js Documentation
<https://reactjs.org/docs/getting-started.html>
4. Dataset
<https://www.kaggle.com/datasets/andrewmvd/trip-advisor-hotel-reviews>
5. Flask Documentation
<https://flask.palletsprojects.com/en/stable/>