High-Level Design (HLD)

Restaurant Rating Prediction

Revision Number: 1.0

Last date of revision: 25/02/2025

iNeuron

Document Version Control:

Date Issued	Version	Description	Author
25/02/2025	**1**	Initial HLD-V1.0	Alieh Hassanzadeh

Contents

Abstract

With the rapid growth of the restaurant industry, customers often struggle to choose dining options that meet their expectations. Inaccurate or biased ratings can mislead potential customers, impacting both consumer satisfaction and business performance. This project aims to develop a Restaurant Rating Prediction system using machine learning techniques. By analysing various factors such as customer reviews, restaurant attributes, pricing, and location, the system predicts ratings to provide more reliable insights. The model processes diverse data sources, including sentiment analysis of reviews and structured restaurant information, to enhance prediction accuracy. The implementation of this system will help diners make informed choices and assist restaurant owners in understanding factors influencing their ratings, ultimately leading to improved service quality and customer satisfaction.

1. Introduction

1.1 Why this High-Level Design Document?

The purpose of this High-Level Design (HLD) Document is to provide a structured representation of the Restaurant Rating Prediction system. It details the architecture, key components, and interaction between modules, ensuring clarity before the implementation phase. This document also helps identify potential design conflicts early and serves as a reference for future enhancements.

The HLD will:

- Define all design aspects of the project
- Describe the user interface and its components
- Outline the hardware and software interfaces
- Specify performance requirements
- Detail the design features and architecture of the system
- Address non-functional attributes, including:
    - Security
    - Reliability
    - Maintainability
    - Portability
    - Reusability
    - Application compatibility
    - Resource utilization
    - Serviceability

1.2 Scope

This document provides an overview of the Restaurant Rating Prediction system, including its database structure, application architecture (layers), system flow, and technology stack. The HLD is written in non-technical to mildly-technical terms to ensure readability for both technical and non-technical stakeholders, including system administrators.

1.3 Definitions

Term	Description
ML Model	Machine learning model used for rating prediction (e.g., Random Forest)
Database	Collection of structured restaurant data used for model training and predictions
IDE	Integrated Development Environment for coding and model implementation
Cloud Platform	Cloud-based or local environment used for model deployment(e.g., AWS)

2. General Description

2.1 Product Perspective

The Restaurant Rating Prediction system is a machine learning-based model designed to provide an accurate rating prediction for restaurants based on structured attributes. This system predicts ratings using numerical and categorical features such as restaurant type, location, votes, online ordering availability, and cost for two plates. The system aims to help customers make informed dining decisions while enabling restaurant owners to improve their services.

2.2 Problem Statement

Customers often rely on ratings to choose restaurants, but these ratings can be biased, inconsistent, or manipulated. Additionally, analysing textual reviews requires Natural Language Processing (NLP), which is beyond the scope of this project. This system aims to:

- Predict restaurant ratings based on structured attributes instead of textual reviews.
- Provide data-driven insights for both customers and restaurant owners.
- Improve accuracy and reliability by leveraging machine learning models like Random Forest and other techniques.

2.3 Proposed Solution

The solution uses machine learning algorithms to analyse structured restaurant data and predict ratings. Key features of the solution include:

- Feature Engineering: Extracting meaningful insights from categorical and numerical restaurant attributes.
- Model Selection: Using Random Forest and other ML models to achieve optimal performance.
- Data Processing: Handling missing values, encoding categorical variables, and normalizing numerical data.

- Performance Evaluation: Assessing the model using metrics like RMSE, R^2 score, and MAE to ensure accuracy.
- User Interface: Displaying predictions via a simple dashboard or API for integration with restaurant platforms.

2.4 Further Improvements

Potential improvements for future iterations include:

- **Incorporating Sentiment Analysis**: Using NLP to analyses customer reviews alongside structured data.
- **Adding More Features**: Considering factors like restaurant ambiance, service quality, and seasonal trends.
- **Real-time Predictions**: Integrating the model with live restaurant data for dynamic rating updates.
- **Handling Class Imbalance**: The dataset is imbalanced, with under 3,000 samples out of 51,000 having ratings below 3.0. This can lead to biased predictions. Future work could explore techniques like oversampling, undersampling, SMOTE, or weighted models to improve model fairness.

2.5 Technical Requirements

The Restaurant Rating Prediction system requires:

- Structured Data: Input features include restaurant name, location, category, restaurant type, online order availability, table booking options, cost for two plates, votes, and grouped cuisines.
- Machine Learning Frameworks: Python-based libraries for model training and evaluation.
- Data Preprocessing Tools: Handling missing data, encoding categorical variables, and scaling numerical values.
- Deployment Infrastructure: Cloud or local deployment for accessibility and scalability.

2.6 Data Requirements

The dataset should:

- Contain a balanced distribution of restaurant ratings (if possible). However, in the current dataset, ratings below 3.0 are significantly underrepresented (**with** fewer than 3,000 samples out of 51,000), which may affect model performance.
- Include at least 10,000 restaurant records for effective training.
- Have minimal missing values and well-structured categorical variables.
- Feature normalized numerical values for better model performance.
- Be stored in a NoSQL database (Cassandra) or CSV format for easy access.

2.7 Tools Used




- **Programming Language:** Python
- **Machine Learning Libraries:** Scikit-learn, Pandas, NumPy
- **Visualization:** Matplotlib, Seaborn
- **Database Management:** Cassandra
- **Version Control:** GitHub
- **IDE:** Visual Studio Code, Jupyter Notebook
- **Model Deployment:** Flask for API, AWS for cloud deployment
- **Frontend Development:** HTML, CSS
- **Automation & CI/CD:** Integrated CodePipeline and GitHub Actions

2.7.1 Hardware Requirements

- **System Requirements:** 32GB+ RAM (recommended for efficient ML processing and large datasets)
- **GPU Support (Optional):** For large-scale model training, especially for boosting performance in complex models.
- **Cloud Infrastructure:** AWS (for cloud deployment, data storage, and processing)
- **Additional Requirements:** Reliable internet connection for AWS services and continuous integration (CI/CD) processes via CodePipeline and GitHub Actions.
-

2.8 Constraints

- **Structured Data Only**: The system relies solely on structured attributes and does not analyse text reviews due to the complexity of NLP.
- **Data Availability**: The accuracy depends on the quality and completeness of the dataset.
- **Class Imbalance**: The dataset is imbalanced, with fewer than 3,000 out of 51,000 records having ratings below 3.0, which may affect prediction accuracy for lower-rated restaurants.
- **Model Interpretability**: Some ML models, such as Random Forest, may lack explainability, requiring feature importance analysis.

2.9 Assumptions

- The dataset used is representative of real-world restaurant ratings.
- Class imbalance exists, with significantly fewer restaurants rated below 3.0, which may impact model performance.
- Restaurants maintain consistent operational features (e.g., location, type, pricing).
- The machine learning model can provide reliable rating predictions based on the available attributes.

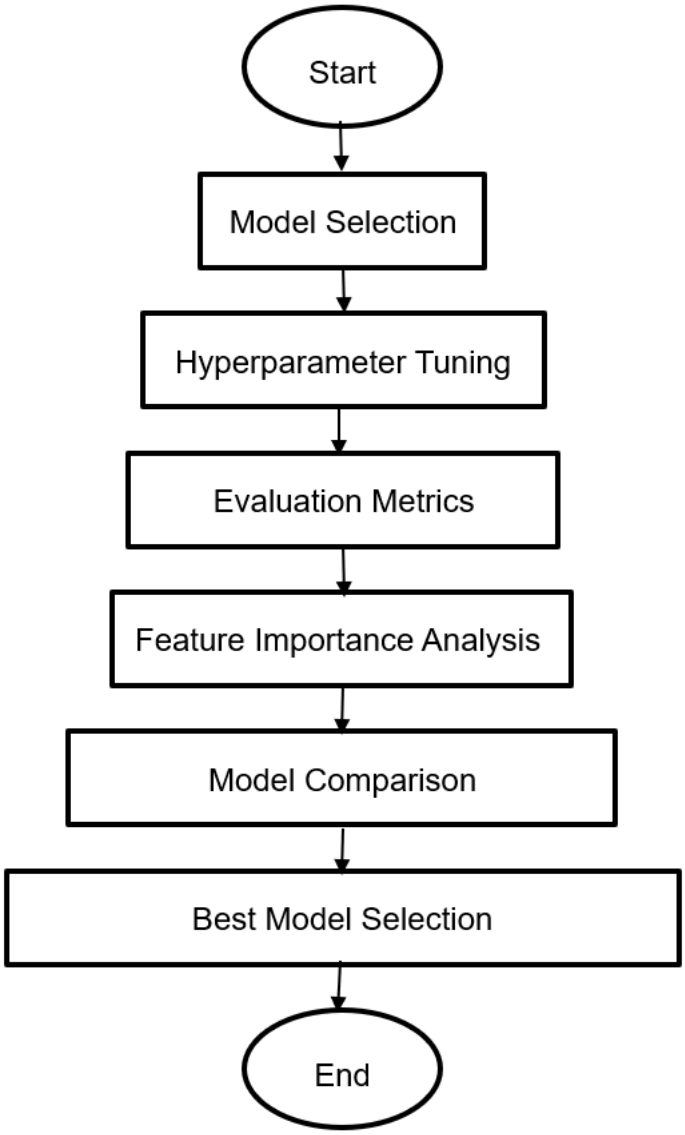
3. Design Details

3.1 Process Flow

This section outlines the structured process for restaurant rating prediction using machine learning models. The system follows a step-by-step approach to ensure accurate predictions based on structured attributes.

3.1.1 Model Training and Evaluation

1. **Model Selection:**
    - Train machine learning models such as Random Forest, Decision Tree, K-Neighbors Regressor, Gradient Boosting Regressor using structured data.
    - Optimize performance through hyperparameter tuning.
2. **Evaluation Metrics:**
    - Measure performance using r2_score.
3. **Feature Importance Analysis:**
    - Identify significant features contributing to predictions to enhance model interpretability.
4. **Comparison of Models:**
    - Evaluate multiple models and select the best-performing one for deployment.


Start
Model Selection
Hyperparameter Tuning
Evaluation Metrics
Feature Importance Analysis
Model Comparison
Best Model Selection
End

3.1.2 Deployment Process

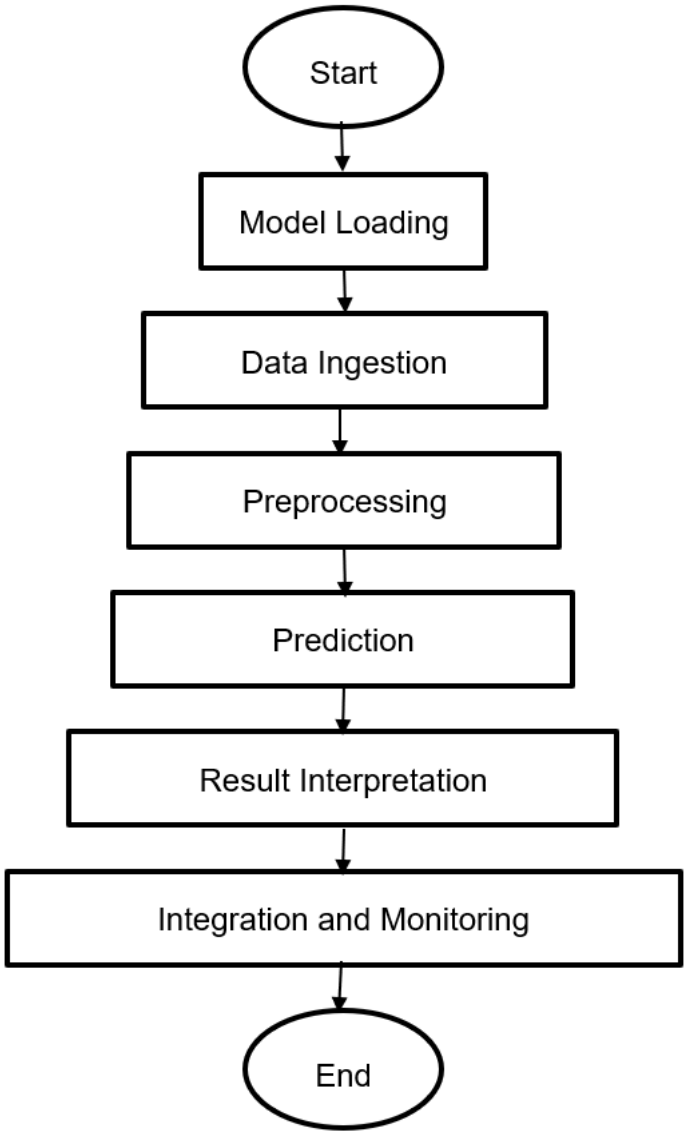
1. **Model Loading:**
    - Load the trained machine learning model into the deployment environment.
2. **Data Ingestion:**
    - Receive new restaurant data as input.
3. **Preprocessing:**
    - Apply necessary preprocessing steps to the input data to match the model's expected format.
4. **Prediction:**
    - Use the trained model to predict restaurant ratings.
5. **Result Interpretation:**
    - Output the predicted rating for further analysis and decision-making.
6. **Integration & Monitoring:**
    - Deploy the model via an API (e.g., Flask) or integrate it into an application.
    - Continuously monitor performance and update the model as needed.

Deployment Tools:

- Used AWS for deployment.
- Integrated CodePipeline and GitHub Actions for automation and CI/CD processes.

Deployment Process Flow Diagram

Refer to the flowchart below for a visual representation of the deployment process:



3.2 Event Log

The system should log every event to allow users to track the processes running internally.

Initial Step-By-Step Description:

1. The system identifies at which step logging is required.
2. The system should be able to log each and every system flow.
3. Developers can choose the logging method, such as database logging or file logging.
4. The system should not hang even with extensive logging, ensuring smooth performance. Logging is mandatory as it allows easy debugging of issues.

3.3 Error Handling

Should errors be encountered, the system will provide an explanation of what went wrong. An error will be defined as anything that falls outside the normal and intended usage.

4. Performance

The restaurant rating prediction model is designed for high accuracy to ensure reliable predictions of restaurant ratings. The model avoids misleading stakeholders (e.g., restaurant owners, customers) by providing precise ratings based on data. To maintain and improve its performance, the model undergoes periodic retraining with new data, ensuring that it adapts to evolving trends and continues to deliver accurate predictions.

4.1 Reusability

The code and components developed for the restaurant rating prediction system are fully reusable. This includes the models, preprocessing pipelines, and the API configuration, all designed with modularity in mind. This approach allows for easy updates or adaptations in future iterations or if the system is applied to other similar datasets.

4.2 Application Compatibility

Python is used as the primary interface between the components of the restaurant rating prediction system. Each component (model, data preprocessing, and prediction output) performs a specific task, with Python handling the seamless transfer of data and actions between them. This ensures smooth operation and communication across the system.

4.3 Resource Utilization

The system efficiently utilizes available resources during model training and deployment. During training, the system optimizes the use of available CPU and memory resources to handle large datasets and complex models. After deployment, it continues to perform efficiently, utilizing the necessary resources for real-time predictions while maintaining optimal performance.

4.4 Deployment

The restaurant rating prediction model is deployed using AWS for cloud infrastructure, with Flask serving as the API for making real-time predictions on new

restaurant data. AWS services, along with CodePipeline and GitHub Actions, are used to automate and streamline the deployment process. The system is designed to scale based on demand, ensuring consistent performance and easy updates.

5. Conclusion

The designed Restaurant Rating Prediction System successfully predicts restaurant ratings based on structured data, utilizing machine learning models such as Random Forest, Decision Tree, K-Neighbors Regressor, and Gradient Boosting Regressor. By analyzing various features, including restaurant type, location, online ordering availability, and customer votes, the system provides valuable insights into restaurant performance.

To ensure high accuracy and reliability, the model undergoes hyperparameter tuning. The deployment process is streamlined using AWS, CodePipeline, and GitHub Actions, ensuring scalability and automation. The system is designed to be efficient, reusable, and compatible, allowing for seamless integration and future improvements.

With this predictive capability, restaurant owners and stakeholders can make data-driven decisions to enhance their services, leading to improved customer satisfaction and business growth.

6. References

1. **Machine Learning Model Training & Evaluation**
    - Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow* (2nd ed.). O'Reilly Media.
    - Scikit-learn Developers. *Scikit-learn: Machine Learning in Python*. Retrieved from https://scikit-learn.org/
2. **Hyperparameter Tuning & Model Selection**
    - Probst, P., Wright, M. N., & Boulesteix, A. L. (2019). *Hyperparameters and Tuning Strategies for Random Forest*. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 9(3).
3. **Deployment & CI/CD (AWS, CodePipeline, GitHub Actions)**
    - Amazon Web Services. *AWS Documentation*. Retrieved from https://docs.aws.amazon.com/
    - GitHub. *GitHub Actions Documentation*. Retrieved from https://docs.github.com/en/actions
    - MLOps & CI/CD Best Practices. Retrieved from https://ml-ops.org/
4. **Logging & Error Handling**
    - Python Logging Module. Retrieved from https://docs.python.org/3/library/logging.html
5. **Reusability & Application Compatibility**
    - Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley.
    - Python Software Foundation. *PEP 8 – Style Guide for Python Code*. Retrieved from https://peps.python.org/pep-0008/