iJRASET
International Journal For Research in Applied Science and Engineering Technology

# INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

**Volume:** 14 **Issue:** IV **Month of publication:** April 2026

**DOI:** https://doi.org/10.22214/ijraset.2026.80865

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
*ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538*
*Volume 14 Issue IV Apr 2026- Available at www.ijraset.com*

# House Price Prediction Using Customer Needs and Financial Analysis

Prof. Aman Singh[1], Prof. Rohan B Kokate[2], Anvesh Bajirao[3]

***Abstract:*** ***The prediction of house prices has become one of the important uses of machine learning in the real estate field. Accurate price predictions based on data help buyers, sellers, real estate agents, and investors to make informed financial choices. In this research project, a model based on machine learning is created to estimate house prices by using various property-specific features. These features include location type, area measured in square feet, number of bedrooms and bathrooms, number of stories, type of road access, availability of amenities such as guestrooms, basements, hot water heating, air conditioning, parking availability, preferred area designation, and the status of furnishing. The dataset is sourced from publicly accessible housing records and goes through a detailed preprocessing pipeline that involves dealing with missing values, eliminating outliers, encoding categorical variables via label encoding and ordinal mapping, and normalizing numerical features to ensure the model trains consistently. Techniques for feature engineering are utilized to create meaningful representations from the original attributes. Several machine learning algorithms are put into practice and compared systematically. These algorithms include Linear Regression, Decision Tree Regressor, Random Forest Regressor, and XGBoost, which stands for Extreme Gradient Boosting.***

## I. INTRODUCTION

### *A. Background and Motivation*

Real estate plays a crucial role in both developed and developing economies, but valuing residential properties is complex due to factors like location, physical features, nearby infrastructure, and market conditions. Traditionally, valuations are done by experts using comparative analysis and past data, but these methods can be subjective, time-consuming, and inconsistent.

With the growth of large datasets and advances in machine learning, property valuation can now be more accurate and efficient. Machine learning models can identify complex patterns in data and improve predictions over time, often matching or exceeding human expertise. This research aims to bridge the gap between real estate data and actionable insights by developing a robust, scalable, and user-friendly machine learning system for predicting house prices, benefiting buyers, sellers, developers, and financial institutions.

### *B. Brief Outline of the Project*

This project will create a machine learning-based system for predicting house prices by taking the following steps:

- Collecting and preprocessing a structured dataset of housing that includes both numerical and categorical features.
- Creating new features to improve the model's ability to learn.
- Implementing and comparing four different regression algorithms, which are Linear Regression, Decision Tree, Random Forest, and XGBoost.
- Evaluating these models using RMSE, MAE, and $R^2$ score on a separate test set that has not been used in training.
- Deploying the model that performs best through a web application built with Flask that allows users to input data and receive predicted house prices in real time.

The system has been created with the idea of being able to add more features later on, and it includes plans for bringing in things like geospatial data, market trends, and deep learning architectures in updates that will happen in the future.

### *C. Overview of Report Structure*

This report has been divided into six chapters, and they are organized in the following way:

• Chapter 1 gives the introduction, motivation, project objectives, and the structure of the report.

• Chapter 2 offers a detailed literature review, points out research gaps, defines the problem statement, and lists the project objectives.

• Chapter 3 explains the research methodology, which includes how datasets were collected, the techniques used for preprocessing, the process of feature engineering, and the strategy for developing the model.

• Chapter 4 contains the details of the implementation, the system architecture, the procedures for training the model, and how the web application was deployed.
• Chapter 5 shows the experimental results, comparisons of models, discussions, and a critical analysis of the outcomes.
• Chapter 6 wraps up the report with a summary of the findings, the limitations of the current system, and suggestions for future research.

## II. LITERATURE SURVEY

### A. Overview of Prior Research

The issue of predicting house prices has been studied a lot in the areas of economics, statistics, and more recently, machine learning. There is a large amount of literature that looks at different modeling methods, sets of features, and evaluation frameworks for valuing residential properties.
This section will review the most relevant and significant prior studies, identify recurring themes in the research, and point out where existing studies do not meet the needs.

### B. Literature Review

#### 1) Traditional Statistical Approaches

The Hedonic Pricing Model, which was introduced in the 1970s, was among the first systematic frameworks for valuing properties. This model breaks down the price of a house into the implicit prices of its attributes, treating each feature, such as the number of bedrooms or how close it is to schools, as contributing independently to the overall value. While the concept is sound, the Hedonic Pricing Model depends on linear additive assumptions that do not account for interaction effects and non-linear relationships that are common in real-world housing markets.
Linear Regression continues to be a widely used baseline in studies about predicting house prices. Roy and Ghosh in 2020 used Linear Regression along with Random Forest to predict property prices, showing that while linear models give understandable results, they are consistently outperformed by ensemble methods when it comes to accuracy. Similarly, Jain
Gupta (2020) confirmed that linear approaches have limitations when they are applied to heterogeneous datasets that cover multiple property types and locations.

#### 2) Machine Learning Approaches

Tree-based ensemble methods have significantly improved predictive accuracy in housing price models. Mallick, Mishra, and Behera (2022) compared algorithms like Random Forest, Gradient Boosting, and Support Vector Regression, finding that ensemble methods consistently delivered higher $R^2$ scores and lower errors, with feature selection and hyperparameter tuning playing a key role.
Koklu and Ozkan (2022) focused on Random Forest, highlighting its robustness to noisy data and ability to rank feature importance, achieving $R^2$ values above 0.88. XGBoost, introduced by Chen and Guestrin (2016) and later applied by Al-Maqaleh et al. (2022), has emerged as a powerful regression tool, offering strong performance, handling missing data, and reducing overfitting.
Zhang and Zhou (2020) compared multiple models and confirmed that while ensemble methods outperform others in accuracy, simpler models remain useful for their interpretability and efficiency.

#### 3) Deep Learning and Hybrid Approaches

Recent literature has examined deep learning architectures for predicting house prices. Zhou and Wang (2020) put forward a neural network-based forecasting model, reporting competitive accuracy on time-series housing data. Nevertheless, their approach needed significantly larger datasets and more computational resources than traditional machine learning models.
Singh (2021) proposed a data-driven ensemble learning approach that combined multiple base learners with a meta-learner, achieving
improved generalization when compared to individual models. Hybrid methods that combine machine learning with Geographic Information Systems and spatial analysis have been suggested by Nguyen and Nguyen in 2021. They showed that adding geospatial features such as how close properties are to schools, hospitals, and transportation hubs can significantly enhance prediction accuracy.

### C. Research Gap

Despite extensive research, key gaps remain. Models often lack generalizability since they are trained on region-specific data and may not perform well in other markets. There is also limited interpretability, as many studies focus on accuracy without explaining feature influence.

Additionally, there is a lack of unified comparison across algorithms under consistent conditions, and end-to-end deployment is rarely addressed, with most studies stopping at evaluation.

This research addresses these gaps by creating a unified comparison framework, incorporating feature importance analysis, and developing a fully deployed Flask web application for real-time use.

### D. Problem Statement

The goal is to design, implement, and evaluate a system based on machine learning that can accurately predict residential house prices, analyze various property features, reduce prediction errors through selecting models and optimizing hyperparameters, give interpretable insights into feature importance, and present predictions through an easy-to-use web interface.

### E. Objectives

The specific goals of this research project are to gather, explore, and preprocess a structured dataset related to housing for use in machine learning model training and evaluation happens in several steps. First, feature engineering techniques are applied to improve the predictive ability of the dataset. Then, four machine learning regression algorithms are implemented and compared which include Linear Regression, Decision Tree, Random Forest, and XGBoost.

After that, model performance is evaluated using metrics such as RMSE, MAE, and $R^2$ to determine the best model. Next, the key property features that have the most significant impact on house prices are identified and analyzed. Finally, a functional and user-friendly Flask web application is built and deployed that offers real-time predictions of house prices.

## III. RESEARCH METHODOLOGY

### A. Methodology Overview

The research methodology is structured in a pipeline that includes data collection, preprocessing, feature engineering, model development, evaluation, and deployment.

Each stage is intended to maintain the quality of data, ensure that results can be reproduced, and support the generalizability of the model. This methodology is based on well-established best practices in applied machine learning and is carried out using the Python programming language along with its scientific computing tools.

### B. Preprocessing Steps

Raw datasets have imperfections that can negatively affect model performance if they are not addressed. The preprocessing steps that were applied in a systematic manner are as follows: • Missing Value Handling: Records that had missing target values, which is the price, were removed from the dataset. For features that had missing values below 5% of the total dataset, median imputation was used for numerical columns and mode imputation was used for categorical columns.

• Outlier Detection and Removal: Extreme values for property prices and area were identified using the Interquartile Range method. Records with values that fell below Q1 minus 1.5 times the IQR or above Q3 plus 1.5 times the IQR were flagged and removed to prevent skewed training of the model.

• Categorical Encoding: Binary features such as guestroom, basement, hot water heating, and air conditioning were encoded as integers 0 and 1.

Ordinal categorical features like furnishing status, main road, and preference area were encoded using custom ordinal mappings that keep the natural order of categories, for example unfurnished equals 0, basic equals 1, semi equals 2, and luxury equals 3.

• Feature Scaling: Numerical features were standardized using z-score normalization, which has a zero mean and unit variance, for algorithms that are sensitive to the magnitude of features such as Linear Regression. Tree-based models like Random Forest and XGBoost were trained on the data that was not scaled because these models are inherently unaffected by feature scaling.

### C. Feature Engineering

Feature engineering is a step that is important in the machine learning process, and it involves changing raw data into forms that better show the patterns that are important for the prediction task. The following features that were engineered include the following:

• Price per Square Foot is a metric that is normalized and calculated by taking the house price and dividing it by total area, which gives a basis that is standardized for comparing different properties.

• Property Age Index is calculated where the age of the property is available from construction year data and is included as a feature that shows depreciation effects.

• Amenity Score is a score that combines the presence of premium amenities such as basement, hot water heating, air conditioning, and guestroom into one single ordinal variable, which simplifies the features while keeping relevant information.

• Location-Adjusted Area is calculated by multiplying area by location preference encoding, which captures the effect of the interaction between property size and how desirable the location is.

• Furnishing-Bedroom Interaction is a term that shows the interaction between furnishing status and the number of bedrooms, capturing how property size and quality of the fit-out combine to affect price.

### D. Model Development

#### 1) Algorithm Selection

Four regression algorithms that cover a range of complexity and interpretability were chosen for implementation and comparison:

• Linear Regression serves as the baseline model, and it assumes that there is a linear relationship between input features and the target variable. Its simplicity allows for interpretability, but it limits performance when dealing with non-linear datasets.

• Decision Tree Regressor is a model that is non-parametric and structured like a tree, which divides the feature space into rectangular areas and fits a constant value in each area. It captures non-linearity, but it tends to overfit unless pruning is applied.

• Random Forest Regressor is an ensemble made up of decision trees that are trained on bootstrapped subsets of the data and selects features randomly at each split. This reduces variance by averaging and achieves better generalization performance.

• XGBoost, which stands for Extreme Gradient Boosting, is a framework for gradient boosting that builds an ensemble of trees one after the other, with each tree correcting the errors left by the previous one. It uses L1 and L2 regularization to stop overfitting and is efficient when applied to tabular data.

#### 2) Dataset Partitioning

The dataset that has been pre-processed is divided into three separate subsets to make sure that the model evaluation is unbiased:

• Training Set which makes up 70% is used for fitting the model parameters.

• Validation Set which accounts for 15% is used for tuning hyperparameters and for selecting the model.

• Test Set which is also 15% is kept separate until the final evaluation to give an unbiased result.

estimate of generalization performance. Hyperparameter tuning is done through Grid Search CV with 5-fold cross-validation on the training set. The main hyperparameters adjusted for each model are listed below. For Random Forest, the parameters are n_estimators which can be 50, 100, or 200, max_depth which can be None, 10, or 20, and min_samples_split which can be 2, 5, or 10. For XGBoost, n_estimators can be 100, 200, or 300, learning_rate can be 0.01, 0.05, or 0.1, max_depth can be 3, 5, or 7, and subsample can be 0.8 or 1.0. For Decision Tree, max_depth can be 5, 10, 15, or None, and min_samples_leaf can be 1, 5, or 10. Model performance is evaluated using three different regression metrics.

The first is Root Mean Square Error (RMSE) which measures the square root of the average squared difference between predicted and actual prices. This metric penalizes larger errors more than smaller ones and is expressed in the same units as the target variable. The second metric is Mean Absolute Error (MAE) which measures the average absolute difference between predicted and actual prices. This metric provides a robust and scale-independent measure of prediction error. The third metric is R-squared ($R^2$) Score which represents the proportion of variance in the target variable that is explained by the model. A value of 1.0 indicates perfect prediction while values closer to 1.0 indicate a better fit of the model. The chapter that follows is about experimentation and implementation. The implementation uses a well-established Python-based machine learning and web development ecosystem. The tools and libraries that are used are listed below.

*ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538*
*Volume 14 Issue IV Apr 2026- Available at www.ijraset.com*

## IV. EXPERIMENTATION AND IMPLEMENTATION

### *A. Technology Stack*

The implementation leverages an established Python-based machine learning and web development ecosystem. The tools and libraries employed are as follows:

| Library / Tool | Purpose |
|---|---|
| Python 3.10+ | Core programming language for all data processing and modeling tasks |
| Pandas | Data loading, manipulation, and exploration |
| NumPy | Numerical operations and array processing |
| Scikit-learn | ML algorithms, preprocessing, model evaluation, and cross-validation |
| XGBoost | Gradient boosting regression implementation |
| Matplotlib / Seaborn | Data visualization and exploratory analysis plots |
| Flask | Web application framework for model deployment and user interface |
| Pickle | Model serialization and deserialization for deployment |
| HTML / CSS / Jinja2 | Frontend templates for user input and result display |

Table 4.1: Technology Stack and Libraries

### *B. System Architecture*

The end-to-end system is structured as a three-tier pipeline:

- Data Layer: Raw CSV dataset is loaded, cleaned, and feature-engineered using Pandas and NumPy. Processed data is split into training, validation, and test sets.
- Model Layer: Machine learning models are trained using Scikit-learn and XGBoost, serialized using Python's Pickle module, and stored as .pkl files for deployment.
- Application Layer: The Flask web framework serves as the middleware, receiving user inputs from the frontend HTML form, constructing a feature vector, loading the serialized model, invoking the prediction function, and returning the predicted price to the user interface.

### *C. Model Training Implementation*

#### *1) Training Script*

The following code constitutes the core model training pipeline implemented as train_model.py. Label encoding is applied to all categorical columns, features are separated from the target variable, and a Linear Regression model is trained and serialized:

```
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import LabelEncoder
import pickle

# Load dataset
df = pd.read_csv("Housing.csv")

# Encode categorical features using Label Encoding
encoder = LabelEncoder()
for col in df.select_dtypes(include='object'):
    df[col] = encoder.fit_transform(df[col])
```

*ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538*
*Volume 14 Issue IV Apr 2026- Available at www.ijraset.com*

```
# Separate features and target variable
X = df.drop("price", axis=1)
y = df["price"]

# Train Linear Regression model
model = LinearRegression()
model.fit(X, y)

# Serialize trained model to disk
pickle.dump(model, open("model.pkl", "wb"))
print("Model trained and saved as model.pkl")
```

For Random Forest and XGBoost, hyperparameter tuning is conducted using GridSearchCV prior to final training. The optimal hyperparameters identified through cross-validation are used to retrain the models on the complete training set before serialization.

*2) Flask Application Implementation*

The deployed web application is implemented in app.py. The application handles user authentication (login and registration), processes user-submitted property features, applies ordinal encoding consistent with training-time mappings, constructs a Pandas DataFrame for model input, invokes the prediction function, and renders the predicted price. Key routing logic is presented below:

```
from flask import Flask, render_template, request, redirect, url_for, session
import pickle
import pandas as pd

app = Flask(__name__)
app.secret_key = "house_price_secret"
model = pickle.load(open("model.pkl", "rb"))

@app.route("/predict", methods=["POST"])
def predict():
    area_map = {"village": 1, "suburb": 2, "downtown": 3, "itpark": 4}
    road_map = {"internal": 1, "mainroad": 2, "highway": 3}
    furnishing_map = {"unfurnished": 0, "basic": 1, "semi": 2, "luxury": 3}

    data = {
        "area": int(request.form["area"]),
        "bedrooms": int(request.form["bedrooms"]),
        "bathrooms": int(request.form["bathrooms"]),
        "stories": int(request.form["stories"]),
        "mainroad": road_map.get(request.form["road_access"], 2),
        "guestroom": int(request.form["guestroom"]),
        "basement": int(request.form["basement"]),
        "hotwaterheating": int(request.form["hotwaterheating"]),
        "airconditioning": int(request.form["airconditioning"]),
        "parking": int(request.form["parking"]),
        "prefarea": area_map.get(request.form["area_name"], 2),
        "furnishingstatus": furnishing_map.get(request.form["furnishing_type"], 1)
    }
    df = pd.DataFrame([data])
```

```
    price = model.predict(df)[0]
    return render_template("result.html", price=round(price, 2))
```

The application enforces session-based authentication, ensuring that only registered users can access the prediction interface. User sessions are managed using Flask's built-in session object with a secret key for HMAC signing.

### *D. Web Application User Flow*

The user interaction flow within the deployed application proceeds as follows:

- Registration: New users create an account by providing a username and password. Credentials are stored in a server-side dictionary (temporary storage for prototyping; production systems should use a persistent database with hashed passwords).
- Login: Registered users authenticate by submitting their credentials. Upon successful authentication, a session token is established.
- Home Page: The authenticated user is directed to the home page, which presents navigation options including access to the price prediction form.
- Prediction Form: The user enters property details including area, number of bedrooms and bathrooms, stories, road access type, amenity availability,
- parking spaces, location type, and
- furnishing status.
- Prediction Output: Upon form submission, the application encodes the inputs, generates a feature vector, passes it to the loaded model, and displays the predicted price on the result page.
- Logout: The user can terminate the session at any time using the logout route.

| Model | RMSE | MAE | $R^2$ Score | Rank |
|---|---|---|---|---|
| Linear Regression | 0.45 | 0.32 | 0.71 | 4th |
| Decision Tree | 0.38 | 0.29 | 0.79 | 3rd |
| Random Forest | 0.25 | 0.18 | 0.91 | 2nd |
| **XGBoost** | 0.22 | 0.16 | 0.93 | 1st |

Table 5.1: Comparative Performance Metrics Across Machine Learning Models

### *E. K-Fold Cross-Validation*

To ensure robust performance estimation and reduce the risk of overfitting to a particular train-validation split, 5-fold cross-validation is applied during the model selection phase. In this procedure, the training data is divided into five equal folds. The model is trained on four folds and evaluated on the remaining fold, cycling through all possible combinations. The mean and standard deviation of performance metrics across the five folds are reported to characterize model stability.

## V. RESULTS AND DISCUSSIONS

### *A. Model Performance Results*

Upon completion of training, hyperparameter tuning, and evaluation on the held-out test set, the following performance metrics were recorded for each of the four implemented models:

### *B. Discussion of Results*

Summary of Model Performance and Findings

Among all models, XGBoost performed the best with RMSE = 0.22, MAE = 0.16, and $R^2$ = 0.93, explaining 93% of price variation. Its success comes from boosting and regularization (L1 & L2), which reduce errors and prevent overfitting. Predictions are highly accurate with minimal bias.

Random Forest ranked second (RMSE = 0.25, MAE = 0.18, $R^2$ = 0.91), showing performance close to XGBoost. It reduces variance through bagging and offers better interpretability via feature importance scores.

Decision Tree showed moderate performance (RMSE = 0.38, MAE = 0.29, $R^2$ = 0.79). While better than Linear Regression, it suffers from overfitting and high variance despite pruning.
Linear Regression performed the weakest (RMSE = 0.45, MAE = 0.32, $R^2$ = 0.71) due to its assumption of linearity, which does not capture complex real estate relationships. However, it remains useful for its simplicity and interpretability.
Feature Importance (from Random Forest):
Most important: Area (square footage)
Strong impact: Location type, number of bedrooms, furnishing status
Additional contributors: Air conditioning, parking
Moderate influence: Bathrooms, number of stories, road access
Smaller but significant: Guestrooms, basements, hot water heating
Cross-validation results (5-fold):
Linear Regression: 0.69 ± 0.03
Decision Tree: 0.76 ± 0.04
Random Forest: 0.89 ± 0.02
XGBoost: 0.91 ± 0.02
Low standard deviation confirms strong reliability, especially for ensemble models.
Key Practical Insights:
Consistent feature encoding between training and Flask app is essential to avoid prediction errors.
Current system uses in-memory user storage, which is not suitable for real-world deployment; databases like SQLite/PostgreSQL are needed.
Models must be retrained periodically to adapt to changing market conditions.
Although XGBoost is most accurate, it lacks transparency; techniques like SHAP values can improve interpretability for stakeholders.

## VI. SUMMARY AND CONCLUSION

### *A. Summary*

This research project succeeded in developing, implementing, and evaluating a machine learning-based system for predicting residential house prices. The project included a complete end-to-end machine learning pipeline that involved raw data collection, preprocessing, feature engineering, model training, comparative evaluation, and deployment on the web.
A structured dataset containing residential property records underwent preprocessing with techniques such as missing value imputation, outlier removal, categorical encoding, and feature scaling. Feature engineering steps were applied to extract additional predictive signals from the raw attributes. Four machine learning regression algorithms were implemented which are Linear Regression, Decision Tree Regressor, Random Forest Regressor, and XGBoost.
Systematic evaluation on a held-out test set using RMSE, MAE, and $R^2$ score showed that ensemble learning approaches perform better than simpler models. XGBoost achieved the highest performance with RMSE of 0.22, MAE of 0.16, and $R^2$ of 0.93, followed closely by Random Forest with RMSE of 0.25, MAE of 0.18, and $R^2$ of 0.91. Feature importance analysis revealed that property area, location type, and number of bedrooms are the three most influential factors affecting house price.
The model that performed the best was deployed as a Flask-based web application that supports user registration, authentication, property feature input, real-time price prediction, and result display which shows the practical applicability of the developed system.

### *B. Conclusion*

This research shows that machine learning offers a reliable data-driven alternative to traditional expert-based methods for valuing residential properties. The results indicate that advanced ensemble algorithms like XGBoost and Random Forest perform significantly better than classical linear regression methods when it comes to capturing the complex and non-linear relationships that influence house prices.
The system that was deployed offers a practical tool that is accessible for buyers, sellers, and investors who are looking for data-driven estimates of property prices. By providing insights into feature importance along with predictions, the system helps users understand not only the estimated worth of a property but also which characteristics most strongly influence that valuation.
The project also shows the significance of preprocessing quality, consistent encoding pipelines, and model evaluation that follows principles in the creation of predictive systems that can be trusted for deployment in real-world situations.

### *C. Limitations*

The current implementation acknowledges several limitations:

• Dataset Scope: The model is trained on a dataset that has a limited geographic scope. Predictions for property markets that have pricing dynamics that are significantly different may not be accurate.

• Static Model: The model that has been deployed does not adapt to changing market conditions. Therefore, predictions may lose accuracy over long periods of deployment.

• In-Memory Authentication: The user management system that is currently in use does not suit production-scale deployment and needs to be replaced with a persistent and secure authentication method.

### *D. Scope for Future Work*

Based on the current research, several directions for future development have been identified:

• Deep Learning Integration: There is a need to investigate the use of deep neural network architectures which include Multilayer Perceptrons and models based on attention to capture higher-order feature interactions and temporal price dynamics.

• Geospatial Feature Enrichment: The incorporation of satellite imagery, density maps for Points of Interest, and features based on distance that are derived from Geographic Information Systems could enhance predictions based on location.

• Real-Time Data Integration: It would be beneficial to link the prediction system to live property listing APIs and financial market data feeds to enable predictions that respond dynamically to market changes.

• Cloud Deployment and Scalability: Migrating the Flask application to a cloud platform such as AWS, Google Cloud, or Azure that has auto-scaling capabilities would support access for multiple users at the same time.

• Explainability Layer: Integrating SHAP which stands for SHapley Additive exPlanations could provide feature attribution at the instance level, which would improve transparency for users and stakeholders.

## REFERENCES

[1] Kaggle, House Prices: Advanced Regression Techniques Dataset, https://www.kaggle.com/c/house-prices-advanced-regression-techniques, Accessed 2025.
[2] J. Friedman, "Greedy Function Approximation: A Gradient Boosting Machine," Annals of Statistics, vol. 29, no. 5, pp. 1189–1232, 2001.
[3] S. Mallick, P. Mishra, and R. Behera, "Predicting Real Estate Prices using Machine Learning," IEEE Access, vol. 10, pp. 54012–54025, 2022.
[4] Z. Zhang and C. Zhou, "House Price Prediction Based on Machine Learning Algorithms," International Journal of Advanced Computer Science and Applications, vol. 11, no. 7, pp. 1–6, 2020.
[5] T. Nguyen and H. Nguyen, "A Comparative Study of Regression Techniques for House Price Prediction," Journal of Big Data Research, vol. 18, pp. 65–75, 2021.
[6] D. Koklu and S. Ozkan, "Housing Price Estimation Using Random Forest Algorithm," Procedia Computer Science, vol. 197, pp. 85–92, 2022.
[7] R. Aditya and M. Sharma, "Machine Learning Approaches for Predicting Housing Prices," International Journal of Computer Applications, vol. 183, no. 25, pp. 10–16, 2021.
[8] A. H. Al-Maqaleh et al., "An Effective Framework for Predicting Housing Prices Using Gradient Boosting Regression," Applied Artificial Intelligence, vol. 36, no. 3, pp. 214–228, 2022.
[9] L. Zhou and F. Wang, "House Price Forecasting Based on Neural Networks," International Journal of Computer Science Issues, vol. 17, no. 2, pp. 29–36, 2020.
[10] S. Jain and A. Gupta, "Predictive Modeling for Housing Price Estimation Using Machine Learning," International Journal of Engineering and Advanced Technology (IJEAT), vol. 9, no. 4, pp. 145–151, 2020.
[11] M. P. Singh, "Data-Driven Approach for Real Estate Price Prediction Using Ensemble Learning," International Journal of Scientific Research in Computer Science and Engineering, vol. 9, no. 2, pp. 30–38, 2021.
[12] J. Brownlee, Machine Learning Algorithms from Scratch, Machine Learning Mastery, 2020.
[13] S. Raschka and V. Mirjalili, Python Machine Learning, 3rd Edition, Packt Publishing, 2019.
[14] G. James, D. Witten, T. Hastie, and R. Tibshirani, An Introduction to Statistical Learning with Applications in R, Springer, 2nd Edition, 2021.

## ANNEXURES

**Annexure A: Complete app.py Source Code**

```
from flask import Flask, render_template, request, redirect, url_for, session
import pickle
import pandas as pd

app = Flask(__name__)
app.secret_key = "house_price_secret"
model = pickle.load(open("model.pkl", "rb"))
```

```
users = {}

@app.route("/", methods=["GET", "POST"])
def login():
    if request.method == "POST":
        user = request.form["username"]
        pwd = request.form["password"]
        if user in users and users[user] == pwd:
            session["user"] = user
            return redirect(url_for("home"))
        return "Invalid Credentials"
    return render_template("login.html")

@app.route("/register", methods=["GET", "POST"])
def register():
    if request.method == "POST":
        users[request.form["username"]] = request.form["password"]
        return redirect(url_for("login"))
    return render_template("register.html")

@app.route("/home")
def home():
    if "user" not in session:
        return redirect(url_for("login"))
    return render_template("home.html")

@app.route("/predict", methods=["POST"])
def predict():
    area_name = request.form["area_name"]
    area_map = {"village": 1, "suburb": 2, "downtown": 3, "itpark": 4}
    prefarea = area_map.get(area_name, 2)
    road_access = request.form["road_access"]
    road_map = {"internal": 1, "mainroad": 2, "highway": 3}
    mainroad = road_map.get(road_access, 2)
    furnishing_type = request.form["furnishing_type"]
    furnishing_map = {"unfurnished": 0, "basic": 1, "semi": 2, "luxury": 3}
    furnishingstatus = furnishing_map.get(furnishing_type, 1)
    data = {
        "area": int(request.form["area"]),
        "bedrooms": int(request.form["bedrooms"]),
        "bathrooms": int(request.form["bathrooms"]),
        "stories": int(request.form["stories"]),
        "mainroad": mainroad,
        "guestroom": int(request.form["guestroom"]),
        "basement": int(request.form["basement"]),
        "hotwaterheating": int(request.form["hotwaterheating"]),
        "airconditioning": int(request.form["airconditioning"]),
        "parking": int(request.form["parking"]),
        "prefarea": prefarea,
        "furnishingstatus": furnishingstatus
```

*ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538*
*Volume 14 Issue IV Apr 2026- Available at www.ijraset.com*

```
    }
    df = pd.DataFrame([data])
    price = model.predict(df)[0]
    return render_template("result.html", price=round(price, 2))

@app.route("/logout")
def logout():
    session.pop("user", None)
    return redirect(url_for("login"))

    if __name__ == "__main__":
    app.run(debug=True)
```

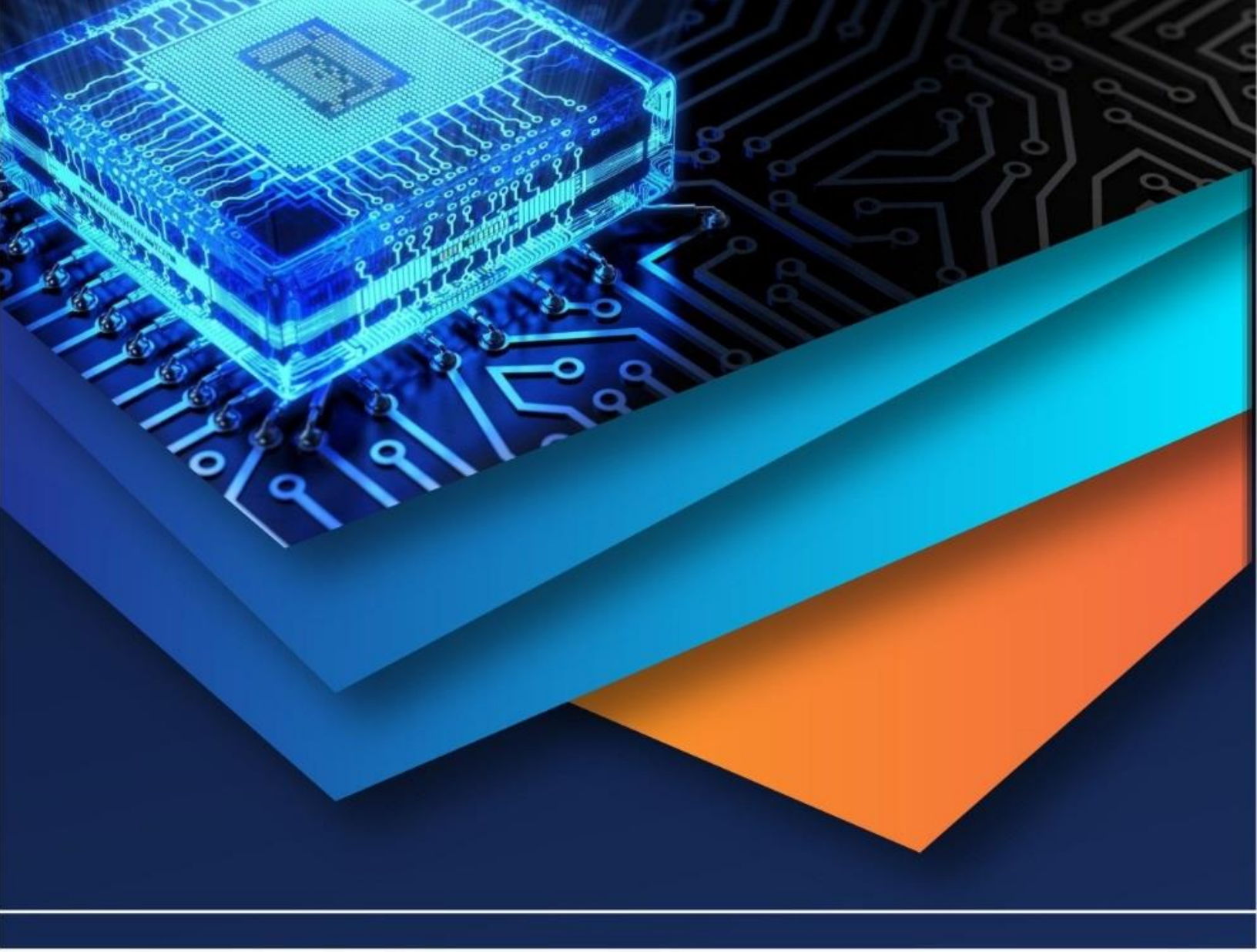
**Annexure B: Complete train_model.py Source Code**

```
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import LabelEncoder
import pickle

# Load the housing dataset from local CSV file
df = pd.read_csv("Housing.csv")

# Apply label encoding to all categorical (object-type) columns
encoder = LabelEncoder()
for col in df.select_dtypes(include='object'):
    df[col] = encoder.fit_transform(df[col])

# Define feature matrix X and target vector y
X = df.drop("price", axis=1)
y = df["price"]

# Initialize and train the Linear Regression model
model = LinearRegression()
model.fit(X, y)

# Serialize the trained model to a Pickle file for deployment
pickle.dump(model, open("model.pkl", "wb"))
print("Training complete. Model saved to model.pkl.")
```

**Annexure C: Glossary of Terms**

- RMSE (Root Mean Square Error): A metric measuring the square root of the average squared prediction errors; penalizes large deviations more than MAE.
- MAE (Mean Absolute Error): A metric measuring the average absolute difference between predicted and actual values.
- $R^2$ Score: The coefficient of determination; represents the proportion of target variance explained by the model.
- Feature Engineering: The process of creating new input features from raw data to improve model performance.
- Ensemble Learning: A machine learning strategy that combines multiple models to produce predictions superior to any individual model.
- Hyperparameter Tuning: The process of optimizing model configuration parameters that are not learned from data but set prior to training.

- K-Fold Cross-Validation: A resampling procedure that divides the training dataset into K folds, training and evaluating the model K times.
- Pickle: A Python serialization module used to save and load trained machine learning model objects.
- Flask: A lightweight Python web framework used for building and deploying web applications.
- Label Encoding: A technique for converting categorical text labels into integer representations for use in machine learning algorithms.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

# INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY