International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 11 | Nov - 2026 SJIF Rating: 8.659 ISSN: 2582-3930

PropVal.AI: An AI-Based Property Price Prediction and Real Estate Valuation Web Application

A Machine Learning Approach Using Linear Regression

Author Name: **Pranit Bhanudas Sardar** Department / Institution: D.G Ruparel College
Academic Year: 2026-2027 Email id:- pranitsardar00@gmail.com

Mrs. Manali Churi Assistant Professor
D.G Ruparel College of Arts,
Science & Commerce manalichuri@ruparel.edu

Abstract

Property valuation is an important activity for buyers, sellers, investors, and real-estate professionals. Estimating a property's value can be difficult because price is influenced by several characteristics, including location, property type, floor area, number of bedrooms and bathrooms, floor level, age, furnishing, and parking availability. This paper presents PropVal.ai, a web-based property price prediction and portfolio-management application developed using Python, Flask, SQLite, and Scikit-learn. The system applies a Linear Regression model within a preprocessing pipeline that standardizes numerical attributes and one-hot encodes categorical attributes. It provides authenticated user accounts, user-specific property records, price predictions, prediction history, dashboard summaries, and interactive visual reports. The application can generate synthetic training data during initial setup and stores the trained pipeline for reuse. Model evaluation is designed around the coefficient of determination (R^2), Mean Absolute Error (MAE), and Root Mean Squared Error (RMSE). Since the documented training process uses generated synthetic data, the resulting metrics should be interpreted as development or internal validation indicators rather than proof of performance on live market transactions. PropVal.ai is therefore presented as an indicative statistical decision-support tool, not as a certified appraisal service.

Keywords: Property valuation, house price prediction, machine learning, Linear Regression, Flask, Scikit-learn, SQLite, real estate analytics.

1. Introduction

Real-estate decisions often require an estimate of a property's market value. Buyers may use an estimate to assess affordability, sellers may use it as an initial pricing reference, and property managers or analysts may use valuation records to monitor a portfolio. Traditional valuation may involve comparable-property analysis, professional judgement, and local market knowledge. These approaches remain important, but they can be time-consuming and may not be readily accessible to every user.

Machine learning provides a way to model relationships between property characteristics and observed or constructed prices. A supervised regression model learns a numerical relationship from examples and uses that relationship to estimate the target value for a new input. The usefulness of such a model depends on the quality, coverage, and representativeness of its training data, as well as the appropriateness of its features and evaluation method.

PropVal.ai combines a regression-based valuation engine with a web application for managing property information. The application is implemented with Flask and Flask-Login, uses SQLite through Flask-SQLAlchemy for persistence, and uses Scikit-learn for data preprocessing and model training. Its interface includes property management, prediction history, dashboard metrics, interactive charts, and a quick-prediction API.

2. Problem Statement and Motivation

The project addresses the difficulty of obtaining a consistent, accessible, and repeatable initial estimate of property value from a set of property attributes. Users may otherwise need to compare information manually or rely on informal estimates. In addition, property details and previous estimates can become scattered when there is no centralized application for recording and reviewing them.

The motivation is to create a single platform that combines an indicative price estimate with property-record management and visual analysis. The project also aims to demonstrate an end-to-end implementation in which model preprocessing, prediction, persistence, authentication, and dashboard reporting are integrated into a usable web application.

2.1 Research Questions

- Can a web application accept structured property attributes and generate a repeatable statistical price estimate using a regression pipeline?
- Can property records and prediction history be stored in a relational database and isolated by user account?
- Can dashboard visualizations help users inspect recorded properties and understand basic patterns in the stored portfolio?
- What additional data and validation are required before the model can be considered suitable for real-market use?

3. Research Objectives

- Develop a responsive web application for indicative property price estimation.
- Implement a Linear Regression model using Scikit-learn.
- Preprocess numerical and categorical property features through a reusable pipeline.
- Store user accounts, property records, and prediction history in an SQLite relational database.
- Provide registration, login, session management, and ownership-based access control.
- Present portfolio summaries and interactive visualizations using Chart.js.
- Evaluate model behaviour using R^2, MAE, and RMSE, while clearly documenting the limitations of synthetic training data.

4. Related Work

Automated property valuation is commonly framed as a supervised regression problem: property attributes are used as input variables and transaction or listing price is the numerical target. Hedonic pricing research provides a conceptual basis for modelling price as a function of a property's characteristics. Rosen's hedonic framework explains how the prices of differentiated goods can reflect the value associated with their individual attributes [1].

Linear Regression is a widely used baseline method for estimating relationships between numerical outcomes and explanatory variables. It is relatively interpretable and provides a useful benchmark, although its assumptions and ability to represent non-linear relationships must be considered [2]. Modern machine-learning libraries make it possible to combine preprocessing and estimation in a single repeatable workflow. Scikit-learn provides tools for transformations, estimators, and evaluation metrics [3].

Web-based decision-support applications extend predictive models by making inputs, outputs, and historical records accessible through an interface. PropVal.ai follows this integrated approach. The README identifies the technologies and application features but does not provide a completed literature review, a comparative benchmark against other algorithms, or evidence from a real-world property transaction dataset. Consequently, this section establishes the conceptual background and does not claim that a systematic review of all existing valuation systems has been conducted.

4.1 Research Gap Addressed by the Project

The project contribution is primarily an integrated software implementation rather than a newly invented regression algorithm. It brings together an indicative valuation workflow, account-based property-record management, prediction history, dashboard visualizations, and a quick-prediction endpoint in one application. The README does not establish a new mathematical model or prove superior predictive accuracy over existing methods.

5. Proposed System and Contributions

PropVal.ai is designed as a browser-based application in which a registered user can manage property information, request a price estimate, and review previous predictions. The main contributions documented in the project are:

- An end-to-end Flask application connecting user-facing pages, API endpoints, database operations, and a machine-learning component.
- A reusable Scikit-learn preprocessing and regression pipeline for numerical and categorical property attributes.
- User-specific ownership isolation for property records and prediction history.
- A dashboard containing portfolio summary metrics and multiple Chart.js visualizations.
- A persisted trained model pipeline and recorded prediction inputs and outputs.
- An explicit disclaimer that generated prices are indicative statistical estimates.

6. System Architecture

The application follows a modular web architecture. The browser-based presentation layer is implemented with HTML templates, Tailwind CSS, custom CSS, Vanilla JavaScript, and Chart.js. The Flask application layer handles routes and API endpoints. Authentication is separated into an authentication blueprint. The data layer uses Flask-SQLAlchemy with SQLite. The machine-learning module generates or prepares training data, builds the preprocessing pipeline, trains the Linear Regression estimator, computes evaluation metrics, and performs predictions.

6.1 Main Components

- Presentation layer: responsive templates for login, registration, dashboard, property records, property entry, prediction, and prediction history.
- Application layer: app.py coordinates routes, request handling, and API endpoints.
- Authentication layer: auth.py provides registration, login, and session-related functionality using Flask-Login and Werkzeug password hashing.
- Data layer: models.py defines the Users, Properties, and Predictions relational entities through Flask-SQLAlchemy.
- Machine-learning layer: ml_model.py generates synthetic training examples, constructs preprocessing and regression steps, trains the model, and returns estimates and metrics.
- Visualization layer: Chart.js displays price distribution, area-versus-price relationships, average price by property type, and location-based value breakdown.

6.2 Prediction Workflow

1. The user enters or adjusts property characteristics through the web interface.
2. The application validates and passes the feature values to the prediction logic.
3. The preprocessing pipeline scales numerical features and encodes categorical features.
4. The trained Linear Regression estimator calculates an estimated property price.
5. The application can save the prediction with its user, associated property where applicable, model name, timestamp, and input data.
6. The dashboard and prediction-history page make stored information available to the authenticated user.

7. Technologies Used

Technology	Role	Project usage
Python	Programming language	Backend and machine-learning implementation
Flask	Web framework	Routes, application logic, and API endpoints
Flask-Login	Authentication support	User login and session management
Werkzeug	Security utilities	Password hashing
SQLite	Relational database	Persistent application records
Flask-SQLAlchemy	ORM	Database models and operations
Scikit-learn	Machine learning	Preprocessing, Linear Regression, and metrics
StandardScaler	Numerical preprocessing	Standardization of numerical features
OneHotEncoder	Categorical preprocessing	Encoding of categorical features
Joblib	Model persistence	Saving the trained pipeline to disk
Tailwind CSS / custom CSS	Frontend styling	Responsive interface and visual theme
Vanilla JavaScript	Client-side behaviour	AJAX quick prediction and interface interactions
Chart.js	Data visualization	Interactive dashboard charts

8. Methodology

8.1 Data and Feature Representation

The documented property schema contains property_type, location, area_sqft, bedrooms, bathrooms, floor, property_age, furnishing, parking, and actual_price. The first nine fields describe the property and are candidate predictor variables; actual_price is the target field for model training where labelled examples are available. The README states that realistic synthetic training data are generated automatically on first run, with feature correlations intended to resemble plausible property relationships.

The README does not specify the number of generated records, the exact probability distributions, the precise feature-generation equations, the train/test split, or whether the generated data are intended to represent a particular city or market. These details should be documented from the implementation before publication.

8.2 Data Preprocessing

A Scikit-learn ColumnTransformer is used to apply different transformations to numerical and categorical variables. StandardScaler is applied to numerical features, while OneHotEncoder converts categorical variables into a numerical representation. This makes the mixed feature set suitable for the regression estimator. The transformation and estimator are assembled into a pipeline so that the same preprocessing operations are applied during training and prediction.

8.3 Linear Regression Model

Linear Regression estimates a numerical target as a weighted combination of input features. In general form, the model can be expressed as:

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p$$

Here, $\hat{y}$ represents the estimated property price, β_0 is the intercept, β_i are learned coefficients, and x_i are the transformed property features. The actual fitted coefficients are learned from the training dataset. The README

identifies Linear Regression as the estimator but does not provide coefficient values or a feature-importance analysis.

8.4 Model Persistence and Prediction

After training, the pipeline is cached to a Joblib file named trained_linear_model.joblib. This allows the application to reuse a trained pipeline rather than retraining it for every prediction. The README also states that model accuracy metrics are computed and that predictions are associated with a model name and input-data record.

8.5 Prediction Output

The system returns a predicted price based on the submitted property characteristics. The prediction is an indicative statistical estimate. It should not be interpreted as a guaranteed selling price, a formal valuation, or professional financial or legal advice.

9. Database Design

The project uses SQLite as its relational database and Flask-SQLAlchemy to define and access the schema. Three principal tables are documented.

- Users: id, username, email, password_hash, created_at. This table stores account identity and the hashed password representation.
- Properties: id, user_id, property_type, location, area_sqft, bedrooms, bathrooms, floor, property_age, furnishing, parking, actual_price, created_at. The user_id field associates a property record with its owner.
- Predictions: id, user_id, property_id, predicted_price, model_name, prediction_date, input_data_json. These fields record who requested a prediction, the related property when available, the output, model identifier, date, and input snapshot.

At a conceptual level, one user can own multiple property records and can have multiple prediction records. A property may be associated with prediction records through property_id. The README documents these fields but does not specify all database constraints, indexes, deletion rules, or migration strategy; these should be confirmed from models.py.

9.1 Data Isolation

The README states that users can only view, edit, or delete their own property records and prediction history. This ownership isolation is an important application-level security requirement. Every read and mutation involving a user-owned record should verify that the record belongs to the currently authenticated user.

10. Application Implementation and Features

10.1 Authentication

Registration and login are implemented with Flask-Login and Werkzeug password hashing. The application includes a pre-seeded analyst account for demonstration and testing. The README lists demo credentials; in a deployed environment, default credentials should be changed or disabled, and secret keys and database configuration should be managed securely.

10.2 Property Management

The application supports full property-record CRUD operations: adding, viewing, editing, and deleting records. The properties page provides filtering and record management. The add-property template is used for entering or editing property details.

10.3 Prediction Interface and Quick Prediction

The prediction page provides an AI valuation interface with live sliders. A quick-prediction endpoint at /api/quick-predict supports dynamic price calculation through AJAX, allowing the user interface to request an updated estimate without a full page reload.

10.4 Dashboard and Visualization

The dashboard includes summary cards for total tracked properties, average portfolio price, latest prediction value, and model R^2. The README lists four visualizations: a price-distribution histogram, an area (square feet) versus price scatter plot, average price by property type, and location-based value breakdown. These charts provide descriptive views of the records and predictions stored by the application.

10.5 Prediction History

A dedicated prediction-history view provides a timeline of previous estimates. The Predictions table preserves the prediction value, model name, date, input data, user association, and optional property association.

11. Testing and Evaluation Plan

The project includes a test_app.py suite described in the README as containing 11 unit and integration tests. The tests are intended to verify application models, machine-learning predictions, authentication routes, and access-security rules. The README states that tests run against an isolated in-memory test database.

11.1 Software Testing

- Authentication tests: verify registration and login behaviour and relevant access restrictions.
- Ownership tests: verify that one user cannot view, edit, or delete another user's property records or prediction history.
- Database tests: verify creation and retrieval of user, property, and prediction records.
- Machine-learning tests: verify that the model pipeline can be initialized and returns a valid numerical prediction for supported inputs.
- API tests: verify that the quick-prediction endpoint handles valid requests and expected validation failures.
- Integration tests: verify that the major application components work together using the test database.

11.2 Predictive Performance Metrics

The README identifies three evaluation metrics. R^2 measures the proportion of target variance explained by the model on the evaluated dataset. MAE is the average absolute difference between predicted and actual values. RMSE is the square root of the average squared prediction error and penalizes larger errors more strongly.

$$\mathbf{MAE} = \frac{1}{n}\sum |y_i - \hat{y}_i| \quad \mathbf{RMSE} = \sqrt{\left[\frac{1}{n}\sum (y_i - \hat{y}_i)^2\right]}$$

The numerical values of R^2, MAE, and RMSE are not included in the README. They must be obtained from an actual execution of the model and reported with the dataset-generation procedure and validation method. No performance value is asserted in this paper.

11.3 Recommended Evaluation Protocol

7. Record the size and generation procedure of the synthetic dataset, including feature ranges and target-price rules.
8. Use a clearly documented training and hold-out test split, or cross-validation where appropriate.
9. Fit all preprocessing steps only on the training data to avoid information leakage.
10. Report R^2, MAE, and RMSE on data not used to fit the model.
11. Compare predicted and actual values using residual plots and error summaries across property types and locations.
12. Where possible, repeat the evaluation on a representative real-world dataset with verified transaction prices.
13. Compare Linear Regression with suitable baseline and non-linear models before making claims about model quality.

12. Results and Discussion

The README supports a description of the implemented application capabilities: user registration and login, user-owned property CRUD, a Linear Regression valuation pipeline, persisted model caching, prediction history, a quick-prediction API, dashboard summary metrics, and four categories of visual reporting. It also documents an automated suite of 11 tests. These are implementation facts rather than measured research outcomes.

A quantitative predictive-performance conclusion cannot be drawn from the README alone because it does not report the metric values, dataset size, train/test methodology, or real-market validation. In particular, synthetic data generated by programmed correlations may make the model appear consistent with the assumptions used to create that data. Such results do not establish that the model can accurately estimate prices in a changing real-estate market.

The most defensible current conclusion is that PropVal.ai demonstrates an integrated prototype for property data management and indicative machine-learning valuation. Further empirical validation is required before making claims about accuracy, generalization, fairness, or suitability for commercial valuation.

13. Novelty and Practical Significance

The project's novelty should be described carefully. Linear Regression, Flask, SQLite, and dashboard charts are established technologies. The distinctive project-level contribution is their integration into a single workflow that combines user-specific property management, a reusable preprocessing-and-regression pipeline, prediction logging, a responsive dashboard, and on-demand AJAX prediction. This integration improves accessibility and traceability within the prototype.

The system also makes a useful distinction between generating an estimate and managing the information around that estimate. By storing the property details and prediction snapshot, it allows users to revisit prior estimates rather than treating each prediction as a one-time output. The documented legal and ethical disclaimer further communicates that the result is indicative.

14. Limitations

- Synthetic training data are used during initial setup; they do not substitute for verified property transaction data.
- The README does not disclose dataset size, generation equations, feature distributions, train/test split, or numerical evaluation results.
- Linear Regression may not capture complex non-linear relationships, location-specific effects, or interactions among property features.
- Important market variables such as recent comparable transactions, neighbourhood amenities, transport access, supply-demand conditions, and market timing are not listed in the documented property schema.
- Location is represented as a categorical feature according to the documented preprocessing approach; the README does not describe geospatial distance or neighbourhood modelling.
- The application disclaimer appropriately limits interpretation, but the README does not document a formal professional appraisal or regulatory validation process.
- Security deployment details such as production secret management, HTTPS configuration, rate limiting, and operational monitoring are not specified in the README.

15. Future Scope

- Replace or supplement synthetic training data with reliable, recent, geographically representative property transaction datasets.
- Add richer location features such as coordinates, locality-level statistics, distance to transport, schools, hospitals, and commercial centres, subject to lawful data availability.
- Evaluate additional regression algorithms such as Random Forest Regressor and Gradient Boosting methods,

using the same hold-out data and metrics.

- Investigate feature interactions and non-linear relationships while preserving interpretability through suitable explanation methods.
- Add model versioning, reproducible training records, drift monitoring, and periodic retraining.
- Provide confidence intervals or prediction ranges where statistically justified, rather than presenting a single value as certain.
- Improve data validation, input constraints, audit logging, deployment security, and backup procedures.
- Conduct user testing to assess usability, accessibility, and the usefulness of the dashboard for different user groups.

16. Ethical, Legal, and Responsible Use

A property estimate can influence financially significant decisions. Users should be informed about the data used, the limits of the model, and the uncertainty associated with the output. The current application describes its output as an indicative statistical estimate. That disclaimer should remain visible wherever a prediction is presented.

If real property data are introduced, the project should consider data provenance, licensing, privacy, consent where applicable, and the risk of biased or incomplete coverage across locations and property types. The model should not be presented as a substitute for a qualified valuation professional. The application should also avoid implying certainty where the model has not been validated on representative market data.

17. Conclusion

This paper presented PropVal.ai, an AI-based property price prediction and real-estate valuation web application. The system integrates a Flask web application, SQLite relational storage, account authentication, property-record management, a Scikit-learn Linear Regression pipeline, prediction history, and interactive dashboard visualizations. Numerical inputs are standardized and categorical inputs are
one-hot encoded through a preprocessing pipeline. The trained pipeline is cached for reuse, while the application records prediction information for later review.

The project demonstrates how a machine-learning component can be integrated into a practical web-based decision-support workflow. Its current contribution is the end-to-end application and its connected features, rather than a newly proposed prediction algorithm. Since the documented training data are synthetic and the README does not provide numerical evaluation results, the current system should be treated as a prototype for indicative estimates. Validation using reliable real-world transaction data, transparent experimental reporting, and comparison with alternative models are essential next steps.

References

[1] S. Rosen, “Hedonic Prices and Implicit Markets: Product Differentiation in Pure Competition,” Journal of Political Economy, vol. 82, no. 1, pp. 34–55, 1974.

[2] G. James, D. Witten, T. Hastie, R. Tibshirani, and J. Taylor, An Introduction to Statistical Learning: with Applications in Python. Springer, 2023.

[3] F. Pedregosa et al., “Scikit-learn: Machine Learning in Python,” Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011.

[4] Pallets Projects, “Flask Documentation.” Available: https://flask.palletsprojects.com/