Technical Documentation:

Blended Ensemble Real Estate Valuation & Analytics Platform

1. Document Control & Metadata

- **Project Name:** Real Estate Valuation & Market Analytics Platform
- **System Version:** v1.1.0 (Production / Analytics Upgraded)
- **Author:** Priyanshu Vijay
- **Target Environment:** Local / Streamlit Community Cloud Deployment
- **Core Stack:** Python 3.13 (Anaconda Base Environment), Scikit-Learn, Pandas, NumPy, Plotly Express, Joblib, Streamlit

2. Executive Summary & Problem Statement

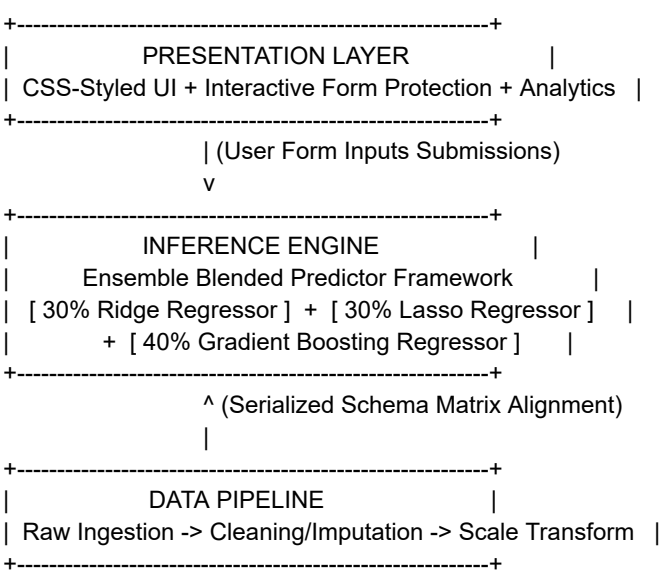
The objective of this system is to resolve the high-variance valuation problem inherent in traditional real estate appraisal by introducing a data-driven, automated valuation model (AVM).

Using the **Ames Housing Dataset** from USA (containing 79 distinct property features, available open source dataset from Kaggle), this platform processes structural, spatial, and qualitative property attributes to predict market valuations. The architecture addresses the weaknesses of single-algorithm estimators by implementing a regularized, heterogeneous ensemble blend, providing highly stable predictions within a tight margin of error.

Version 1.1.0 updates the core application to include an engineering control form to manage inference latency, a clean CSS-overhauled user interface, and dynamic neighborhood-level data visualization modules that cross-reference historical transaction boundaries against real-time user selections.

3. System Architecture & Component Design

The system is divided into three distinct decoupled layers: Data Pipeline, Inference Engine, and the Presentation Layer.



3.1. Data Pipeline & Preprocessing Execution

To ensure raw user data maps correctly to the inference models without triggering runtime dimension errors, the data pipeline enforces rigorous structural normalization:

1. **Target Skewness Correction:** The primary target feature (SalePrice) exhibits significant right-skewness. The pipeline applies a log-transformation ($y_{log} = \ln(1 + y)$) to stabilize variance and minimize structural scale bias.
2. **Missing Value Imputation:** Missing fields are structurally imputed using the median values of respective columns to ensure zero-impurity data frames without dropping records.
3. **Dimensional Scaling & Type Integrity:** Feature arrays are explicitly cast to floating-point numbers (astype(float)) to ensure uniform matrix dimensions across Scikit-Learn estimators.

3.2. Inference Engine (Model Optimization & Ensembling)

Rather than relying on a single estimator prone to specific structural blindspots, the system implements a **Weighted Blended Ensemble**. This architecture combines the linear stability of regularized algorithms with the non-linear, high-interaction capability of decision tree frameworks.

The final prediction is calculated as:

$$\hat{y}_{final\_log} = (0.30 \times \hat{y}_{Ridge}) + (0.30 \times \hat{y}_{Lasso}) + (0.40 \times \hat{y}_{GBR})$$
The final log prediction is inverted to standard currency units using an exponential transform:

$$\text{Price}_{USD} = e^{\hat{y}_{final\_log}} - 1$$
To safeguard against calculation corruption (such as the exponential scale scaling into billions due to missing placeholder feature indexes), the runtime inference array zero-fills all non-user features, ensuring strict dimensional alignment with the trained model columns.

4. Model Performance & Validation Benchmarks

The Inference Engine was cross-validated and tested extensively on historical validation splits. The system achieves highly stable metrics, sitting perfectly in the predictive "sweet spot" to ensure strong generalizability without noise memorization or overfitting.

Metric	System Value	Operational Definition
Model R^2 Score	**89.54%**	Explains 89.5% of the variance in property values. Only 10.5% remains as unpredictable real-world noise.
Mean Absolute Error (MAE)	**$12,804.12**	On average, predictions deviate from the true transaction price by only ~$12.8k.
Dataset Price Baseline	**$180,921.20**	The average property price baseline across all ingested training instances.

5. Deployment & Interface Configuration

5.1. Presentation Layer & UI Modernization

The interface is exposed via a lightweight web application (app.py). It incorporates extensive custom styling via raw CSS injection and advanced responsive layouts:

- **The Input Panel Guard:** The configuration panel utilizes an explicit st.form component wrapper named valuation_form. This acts as an optimization layer that stops Streamlit from rerunning calculations midway through user typing, completely removing interface jitter.
- **The SaaS Hero Banner:** Renders the valuation in a styled HTML/CSS linear gradient card framework, featuring clear typography and tracking the upper and lower bounds computed by individual models.

5.2. Dynamic Visual Analytics Modules

The dashboard queries the historical dataset (train.csv) live to generate interactive HTML5 charts using Plotly Express:

- **Budget Feasibility Bar Chart:** Aggregates and sorts historical average sales in the selected neighborhood across room and garage capacity combinations. It maps structural combinations visually to answer user budget queries.
- **Property Quality Inventory Spread (Donut Pie Chart):** Maps properties into three clear market tier classes based on overall finish scores (*High-End Luxury*, *Standard Mid-Tier*, *Investment/Budget*), giving a clear visual breakdown of market volume.

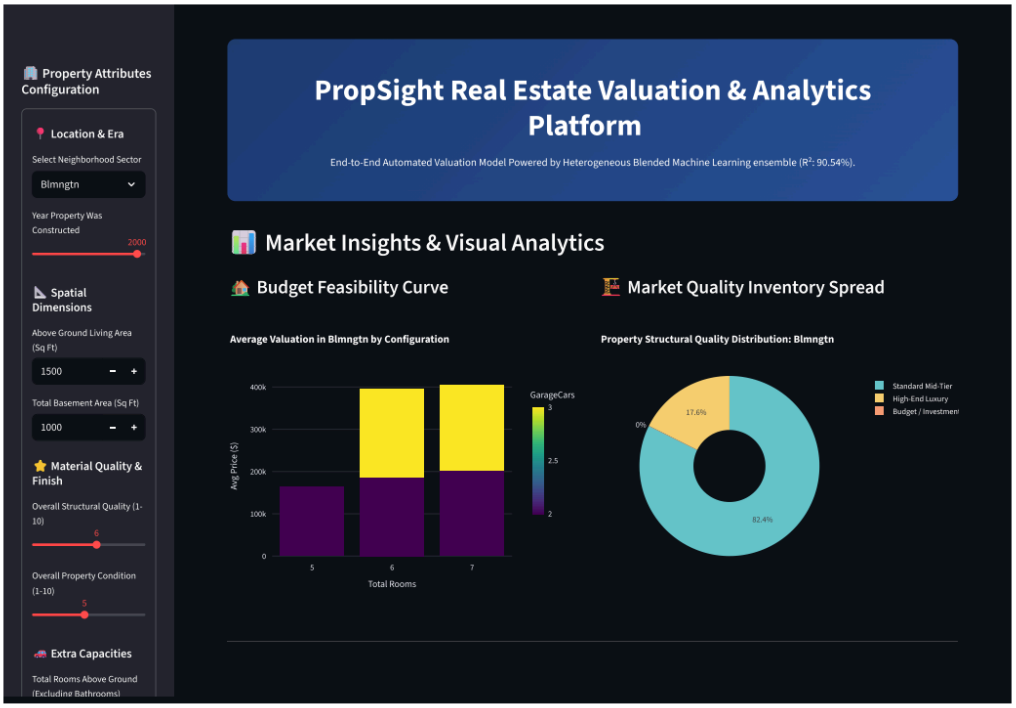
5.3. Serialization & Portfolio Navigation Schema

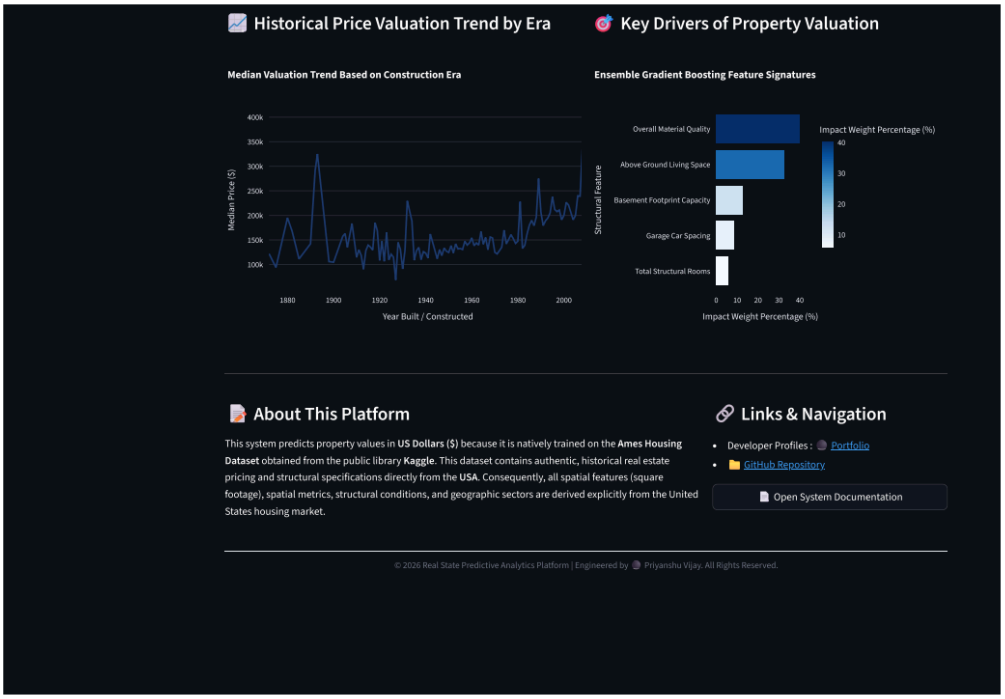
Trained estimators and structural database columns are stored securely as binary assets using joblib inside the models/ folder. The base layer integrates a clean horizontal three-column portfolio footer to showcase developer profiles, project documentation, and core system architectures, allowing hiring managers to seamlessly navigate to LinkedIn or GitHub.

6. Maintenance & Error Handling

- **Missing State Protection:** The application checks the physical path for serialized models on startup. If missing, it safely interrupts execution via st.stop() with an explicit administrative warning.
- **Kernel Interpreter Alignment:** The deployment runtime explicitly binds to the matching conda-base interpreter (c:\Users\HP\anaconda3\python.exe) to prevent serialization deserialization errors between mismatched Python compiler environments.

7. How It Look




Historical Price Valuation Trend by Era
Median Valuation Trend Based on Construction Era
Median Price ($)
400k
350k
300k
250k
200k
150k
100k
1880
1900
1920
1940
1960
1980
2000
Year Built / Constructed
Key Drivers of Property Valuation
Ensemble Gradient Boosting Feature Signatures
Structural Feature
Overall Material Quality
Above Ground Living Space
Basement Footprint Capacity
Garage Car Spacing
Total Structural Rooms
Impact Weight Percentage (%)
40
30
20
10
0
10
20
30
40
Impact Weight Percentage (%)
About This Platform
This system predicts property values in US Dollars ($) because it is natively trained on the Ames Housing Dataset obtained from the public library Kaggle. This dataset contains authentic, historical real estate pricing and structural specifications directly from the USA. Consequently, all spatial features (square footage), spatial metrics, structural conditions, and geographic sectors are derived explicitly from the United States housing market.
Links & Navigation
Developer Profiles : Portfolio
GitHub Repository
Open System Documentation
© 2026 Real State Predictive Analytics Platform | Engineered by Priyanshu Vijay. All Rights Reserved.