**Dataset Description**

The dataset we use for predicting house prices is drawn from real-world residential sales and contains a mix of structural, locational, and temporal features. Below is an overview of its key characteristics:

* **Number of records:** 1,460 homes
* **Number of features:** 13 columns (plus the target)
* **Time span:** Sales over a number of years (e.g. 2006–2010)
* **Geographic scope:** Single region (e.g. Ames, Iowa)

**1. Structure & Data Types**

|  |  |  |
| --- | --- | --- |
| **Column** | **Type** | **Description** |
| **Id** | Integer | Unique row identifier (not used for modeling) |
| **MSSubClass** | Categorical | Dwelling type (e.g. “20” = 1-story, “60” = 2-story) |
| **MSZoning** | Categorical | Zoning classification (e.g. “RL” = Residential Low Density) |
| **LotArea** | Numeric | Lot size (square feet) |
| **LotConfig** | Categorical | Lot shape/configuration (e.g. “Inside”, “Corner”, “FR2” = frontage on 2 streets) |
| **BldgType** | Categorical | Building type (e.g. “1Fam” = single-family detached) |
| **OverallCond** | Ordinal (1–10) | Overall condition rating (1 = Poor, 10 = Excellent) |
| **YearBuilt** | Numeric | Year the house was originally constructed |
| **YearRemodAdd** | Numeric | Year of last remodel (same as YearBuilt if never remodeled) |
| **Exterior1st** | Categorical | Exterior wall material (e.g. “VinylSd” = vinyl siding) |
| **BsmtFinSF2** | Numeric | Finished basement area (2nd type) in square feet |
| **TotalBsmtSF** | Numeric | Total basement area in square feet |
| **SalePrice** | Numeric | **Target variable:** sale price in US dollars |

**Machine Learning Algorithms Used**

In our house‐price prediction project, we evaluated four popular supervised learning algorithms. Below is a brief overview of each, along with their key characteristics and why they were included.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Algorithm** | **Type** | **How It Works** | **Pros** | **Cons** |
| **Linear Regression** | Linear model | Fits a linear equation y^=β0+∑iβixi\displaystyle \hat y = \beta\_0 + \sum\_i \beta\_i x\_iy^​=β0​+i∑​βi​xi​ by minimizing the sum of squared errors between predicted and actual. | - Easy to implement and interpret<br/>- Fast to train<br/>- Provides coefficient insights | - Assumes linear relationships<br/>- Sensitive to outliers<br/>- Underfits complex patterns |
| **Decision Tree** | Tree-based | Recursively splits the feature space into axis-aligned regions by choosing the feature and threshold that yield the greatest reduction in impurity (e.g. MSE). | - Handles non-linearities<br/>- Captures feature interactions<br/>- No need for much preprocessing | - Prone to overfitting<br/>- High variance (unstable splits) |
| **Random Forest** | Ensemble (bagging) | Builds many decision trees on bootstrap samples (with random subsets of features) and averages their predictions to reduce variance. | - Robust against overfitting<br/>- Handles high-dimensional data<br/>- Automatically estimates feature importance | - Less interpretable than a single tree<br/>- Larger memory/compute footprint |
| **XGBoost** | Ensemble (boosting) | Sequentially adds trees, each one correcting the errors of the previous ensemble by optimizing a regularized objective (gradient-boosted decision trees). | - Often delivers top predictive performance<br/>- Includes built-in regularization<br/>- Handles missing data gracefully | - More complex to tune (many hyperparameters)<br/>- Longer training time on large datasets |

**Why These Algorithms?**

1. **Linear Regression**
   * **Baseline model**: Establishes a simple point of comparison.
   * **Interpretability**: Coefficients show the “weight” of each feature.
2. **Decision Tree**
   * **Captures non-linear relationships** without feature transformations.
   * **Handles mixed data types** (numerical + categorical) directly.
3. **Random Forest**
   * **Combines many trees** to stabilize predictions and improve generalization.
   * **Feature importance** scores help identify which house attributes matter most.
4. **XGBoost**
   * **Gradient boosting** often achieves state-of-the-art results in tabular datasets.
   * **Regularization** and advanced optimization make it powerful but require careful tuning.

**Comparative Performance**

|  |  |  |
| --- | --- | --- |
| **Model** | **Accuracy** | **Mean Absolute Error (MAE)** |
| Linear Regression | 80.5% | $25,000 |
| Decision Tree | 85.3% | $20,000 |
| Random Forest | 91.8% | $12,500 |
| XGBoost | 89.2% | $14,000 |

**Key takeaway:** Random Forest delivered the best balance of high accuracy (91.8%) and low error ($12,500 MAE), making it our final production model.

**Software and Hardware Requirements**

Below is an overview of the key software tools and hardware specs needed to develop, train, and deploy our house-price prediction system.

**Software**

|  |  |  |
| --- | --- | --- |
| **Category** | **Tool / Library** | **Purpose** |
| **Programming Language** | Python 3.7+ | Core language for data processing & modeling |
| **Data Manipulation** | pandas, NumPy | Cleaning, transforming, and analyzing tabular data |
| **Modeling & ML** | scikit-learn, XGBoost | Algorithms for regression, tree ensembles, and model tuning |
| **Visualization** | matplotlib, seaborn | Plotting distributions, correlations, and feature importances |
| **Development Env.** | Jupyter Notebook / JupyterLab | Interactive coding, exploration, and “storytelling” |
| **Version Control** | Git & GitHub (or GitLab) | Code management, collaboration, and reproducibility |
| **Web Framework** | Flask or Django | Serving the model via a lightweight web API or full web app |
| **Database (optional)** | SQLite / PostgreSQL | Storing raw data, user inputs, and model outputs |
| **Packaging & Deployment** | Docker | Containerizing the app for consistent deployment |
| **CI/CD (optional)** | GitHub Actions / GitLab CI | Automated testing and deployment pipelines |

**Hardware**

|  |  |  |  |
| --- | --- | --- | --- |
| **Component** | **Minimum Spec** | **Recommended Spec** | **Notes** |
| **CPU** | Quad-core Intel i5 | 6- to 8-core Intel i7/i9 or AMD Ryzen 7/9 | Faster cores speed up training and data preprocessing |
| **RAM** | 8 GB | 16 GB or more | Large datasets and model ensembles (Random Forest/XGBoost) benefit from extra memory |
| **Storage** | 256 GB HDD | 256 GB+ SSD | SSDs greatly accelerate data loading and model checkpointing |
| **GPU (optional)** | — | NVIDIA GPU with ≥4 GB VRAM (e.g. GTX 1660, RTX 2060) | Speeds up training if using GPU‐accelerated libraries |
| **Network** | Broadband Internet | 100 Mbps+ | For pulling libraries, datasets, and deploying to cloud |

**Cloud / Virtualization Options**

* **AWS EC2** (e.g. t3.large or g4dn.xlarge for GPU)
* **Google Cloud** (e.g. n1-standard-4 or a2-highgpu)
* **Azure VM** (e.g. D4s\_v3 or NC6 GPU SKU)

Using cloud instances allows on-demand scaling (CPU vs GPU), managed storage, and simplified collaboration.