Approach and Summary

Data Preparation and Preprocessing

1. Reading the Dataset:

- Load the Austin housing dataset and drop unnecessary columns such as 'streetAddress' and 'description'.
- Create a new column 'logLatestPrice' by applying the natural logarithm to 'latestPrice' for better handling in regression models.

2. Initial Data Exploration:

- Display the first few rows of the modified dataset.
- Print column names and identify categorical variables.

3. Dropping Irrelevant Columns:

- Drop columns with weak correlation with 'latestPrice' or those that provide redundant information:
  - `numOfPhotos` (weak correlation with 'latestPrice').
  - `homeType` (only contains a single value 'Single Family').
  - `hasGarage` (information captured by `garageSpaces`).

4. Feature Engineering:

- Calculate new features:
  - `property_age`: Age of the property based on the current year and 'yearBuilt'.
  - `time_since_last_sale`: Time since the last sale based on the current year and 'latest_saleyear'.
  - Drop the original columns after calculation:
  - `yearBuilt`, `latest_saledate`, `latest_saleyear`.

5. Handling Binary Columns:

- Convert binary columns (`hasAssociation`, `hasSpa`, `hasView`) to numeric.
- Create a new column `combined_features` by summing these binary columns.
- Drop the original binary columns after creating the combined column.

6. Clustering and Seasonal Features:

- Perform K-means clustering on latitude and longitude to create `location_cluster` with 7 clusters.
- Create a new column `season` based on `latest_salemonth`, categorizing into Winter, Spring, Summer, and Fall.

7. Aggregating External Features:

- Create `external_features` by summing `numOfParkingFeatures`, `numOfPatioAndPorchFeatures`, `numOfSecurityFeatures`, `numOfWaterfrontFeatures`, `numOfWindowFeatures`, `numOfCommunityFeatures`, and `numOfAccessibilityFeatures`.
- Drop the original columns after creating the aggregated column.

8. Aggregating Amenities:

- Create `total_amneties` by summing `numOfBathrooms`, `numOfBedrooms`, `numOfStories`, `numOfAppliances`, and `garageSpaces`.
- Drop the original columns after creating the aggregated column.

Model Training and Evaluation

1. Data Splitting:

- Split the dataset into training (80%) and testing (20%) sets.

2. Decision Tree Model:

- Train a decision tree model using selected features.
- Evaluate the model using Mean Squared Error (MSE).
- Perform cross-validation and pruning to optimize the decision tree.

3. Bagging Model:

- Train a Bagging model using Random Forest with all features (`mtry = 17`).
- Evaluate the model using MSE.
- Identify important variables and visualize them using a variable importance plot.

4. Random Forest Model:

- Train multiple Random Forest models with different numbers of features (`mtry` values).
- Evaluate each model's performance and identify the optimal `mtry` value.
- Train the final Random Forest model with the best `mtry` value and visualize important variables.

5. BART Model:

- Train a Bayesian Additive Regression Trees (BART) model using the training data.
- Evaluate the BART model using MSE.

Model Comparison and Selection

- After comparing the MSE values of the models, the BART model had the lowest MSE. However, due to overfitting, the Bagging model (Random Forest) was selected as the final model for prediction on the holdout dataset.

Prediction on Holdout Dataset

1. Preprocessing Holdout Dataset:

- Read the holdout dataset and apply the same preprocessing steps as the training dataset.
- Handle missing values by replacing them with -1 and dropping rows with NA values.

2. Ensure Consistency:

- Ensure consistent columns between the training and holdout datasets.

3. Final Prediction:

- Use the Bagging model to predict housing prices on the holdout dataset.
- Write the final predictions to a CSV file.

This comprehensive approach ensures that the dataset is well-prepared, feature-engineered, and that multiple models are trained and evaluated for robust housing price prediction. The Bagging model is ultimately selected to balance performance and prevent overfitting.