Ames Housing Price Prediction Project

Project report

Project Overview

The Ames Housing Price Prediction project aims to develop a robust and accurate model for predicting the sale price of houses in Ames, Iowa. Leveraging a comprehensive dataset with 79 explanatory variables describing various aspects of residential homes, this project explores different machine learning techniques to identify the most influential factors affecting housing prices and to create a predictive model that can generalize well to unseen data. This report details the project's methodology, results, and potential future enhancements.

Data Processing Pipeline

The data processing pipeline consists of three main stages: **Data Cleaning**, **Feature Engineering**, and **Model Development**.

Data Cleaning

1. Outlier Detection

- Used scatter plots to identify outliers by plotting features against SalePrice
- Key features analyzed:
  - Property dimensions (LotFrontage, LotArea)
  - Building characteristics (OverallQual, YearBuilt)
  - Space measurements (various square footage values)
  - Room and feature counts
- Identified and removed 24 outlier properties

2. Missing Value Treatment

- Categorical Features:
  - Filled 'No' for features like:
    - Alley access
    - Fence type
    - Masonry veneer type
    - Fireplace quality

        - Garage conditions
- Numerical Features:
    - Filled 0 for:
        - LotFrontage
        - MasVnrArea
        - Other measurement-based features

3. Feature Engineering

- Created new features:
    - houseage: Age of house at sale time
    - houseremodelage: Years since last remodel
    - totalsf: Combined square footage
    - totalarea: Total area including basement
    - totalbaths: Total number of bathrooms
    - totalporchsf: Combined porch areas

4. Feature Reduction

- Removed redundant features:
    - Dropped low-importance features (PoolQC, MiscFeature)
    - Eliminated highly correlated pairs
        - Example: Removed GarageArea (correlated with GarageCars)
    - Dropped original features used in engineering new ones

5. Data Export

- Final cleaned datasets saved as:
    - train_df.csv
    - test_df.csv

This cleaning process resulted in a more efficient dataset with meaningful features and fewer redundancies, ready for model development.

EDA Analysis

1. Target Variable Analysis (SalePrice)

- **Visualization**: Histogram with KDE
    - Original distribution showed right skewness
    - Log transformation resulted in normal distribution
    - **Insight**: Log transformation of SalePrice would be beneficial for modeling

2. Numerical Features Analysis

- **Method**: Correlation Analysis with SalePrice
- **Visualization**: Bar plot of top 10 correlations

- **Key Findings**:
  - Overall Quality had highest correlation
  - Total area showed strong positive correlation
  - Living area and garage features were significant predictors

3. Categorical Features Analysis

- **Visualizations**: Combined Box plots and Count plots
- **Features Analyzed**:
  - MSZoning (Zoning classification)
  - Neighborhood
  - BldgType (Building type)
  - HouseStyle
- **Insights**:
  - Different neighborhoods showed distinct price ranges
  - Building types had significant price variations
  - Zoning classification impacted property values

4. Numerical Features Deep Dive

- **Visualizations**: Scatter plots and Histograms
- **Features Analyzed**:
  - OverallQual
  - totalarea
  - TotRmsAbvGrd
- **Key Findings**:
  - Linear relationship between quality and price
  - Total area showed positive correlation with some outliers
  - Room count had moderate correlation with price

5. Engineered Features Analysis

- **Visualization**: Correlation Heatmap
- **Features Analyzed**:
  - houseage
  - houseremodelage
  - totalsf
  - totalarea
  - totalbaths
  - totalporchsf
- **Key Insights**:
  - Strong positive correlation between totalarea and price
  - Moderate negative correlation with house age
  - Total bathrooms showed positive correlation
  - Porch area had weak correlation

6. Key EDA Outcomes

- Log transformation necessary for target variable

- Several strong predictors identified
- Engineered features proved valuable
- Some features showed non-linear relationships
- Categorical variables needed appropriate encoding
- Outliers identified in several features

Data Transformation and Scaling

1. Feature Type Separation

- **Categorical Features**: Identified object dtype columns
- **Numerical Features**: Identified int64 and float64 dtype columns
- **Target Variable**: Separated 'SalePrice' from features

2. Encoding Strategy Division

- **Ordinal Encoding (18 features):**

```
ode_cols = ['LotShape', 'LandContour','Utilities','LandSlope', 'BsmtQual', 'BsmtFinType1', 'CentralAir', 'Functional', \
            'FireplaceQu', 'GarageFinish', 'GarageQual', 'PavedDrive', 'ExterCond', 'KitchenQual', 'BsmtExposure', 'HeatingQC','ExterQual', 'BsmtCond']
```

✓ 0.0s Python

- **One-Hot Encoding (19 features):**

```
ohe_cols = ['Street', 'LotConfig','Neighborhood', 'Condition1', 'Condition2', 'BldgType', 'HouseStyle', 'RoofStyle', 'Exterior1st', 'Exterior2nd',
            'MasVnrType','Foundation',  'Electrical',  'SaleType', 'MSZoning', 'SaleCondition', 'Heating', 'GarageType', 'RoofMatl']
```

3. Pipeline Creation

1. **Numerical Pipeline**:
   - Imputation: Mean strategy
   - Scaling: StandardScaler
2. **Ordinal Pipeline**:
   - Imputation: Most frequent strategy
   - OrdinalEncoder with unknown value handling
3. **One-Hot Pipeline**:
   - Imputation: Most frequent strategy
   - OneHotEncoder with unknown category handling

4. Combined Transformation

- Used ColumnTransformer to combine all pipelines
- Applied parallel processing (n_jobs=-1)
- Maintained column order and handled remaining features

5. Target Variable Transformation

- Applied log transformation to SalePrice
- Reason: To achieve normal distribution (based on EDA)
- Used np.log1p() to handle zero values

6. Final Preprocessing

- Created single pipeline combining all transformations
- Applied transformation to both training and test sets
- Preserved feature relationships while scaling

Building Regression Models

Models Implemented

1. **Base Models**:
- Linear Regression (baseline model)
- Random Forest Regressor
- XGBoost Regressor
- Ridge Regressor
- Gradient Boosting Regressor
- Light GBM Regressor
- CatBoost Regressor
2. **Ensemble Methods**:
- Voting Regressor (combining GBR, XGBoost, Ridge)
- Stacking Regressor (combining all models)

Model Training Approach

- Train-test split: 80-20 ratio
- Cross-validation: 5-fold for most models, 3-fold for LGBM and CatBoost
- Hyperparameter tuning using GridSearchCV
- Parallel processing (n_jobs=-1) for faster computation

Evaluation Metrics Used

- Mean Absolute Error (MAE)
- Mean Squared Error (MSE)
- Root Mean Squared Error (RMSE)
- R^2 Score

Hyperparameter Tuning Details

1. **Random Forest**:
   - max_depth: [5, 10, 15]
   - n_estimators: [100, 200, 500]
   - min_samples_split: [3, 5, 10]
2. **XGBoost**:

    - learning_rate: [0.5, 0.1, 0.2]
    - n_estimators: [300]
    - max_depth: [3]
    - other parameters for fine control
3. **Gradient Boosting**:
    - max_depth: [12, 15, 20]
    - n_estimators: [200, 300, 1000]
    - learning_rate: [0.001, 0.01, 0.1]

Ensemble Methods Details

1. **Voting Regressor**:
    - Combined best models: GBR, XGBoost, Ridge
    - Weighted voting: [2, 3, 1]
    - Better generalization than individual models
2. **Stacking Regressor**:
    - Base estimators: GBR, XGBoost, CatBoost, LGBM, Random Forest
    - Final estimator: Voting Regressor
    - Best overall performance

Performance Results

- Stacking Regressor achieved best performance
- Voting Regressor second best
- Among individual models:
    1. XGBoost
    2. CatBoost
    3. LGBM
    4. Random Forest
    5. Gradient Boosting
    6. Ridge
    7. Linear Regression

Key Advantages of Final Model

- Combines strengths of multiple models
- Reduces overfitting through ensemble approach
- Better generalization on unseen data
- Robust to different types of features

Models Implemented

1. **Base Models**:

- Linear Regression (baseline model)
- Random Forest Regressor
- XGBoost Regressor
- Ridge Regressor
- Gradient Boosting Regressor
- Light GBM Regressor

- CatBoost Regressor
2. **Ensemble Methods**:
- Voting Regressor (combining GBR, XGBoost, Ridge)
- Stacking Regressor (combining all models)

Model Training Approach

- Train-test split: 80-20 ratio
- Cross-validation: 5-fold for most models, 3-fold for LGBM and CatBoost
- Hyperparameter tuning using GridSearchCV
- Parallel processing (n_jobs=-1) for faster computation

Evaluation Metrics Used

- Mean Absolute Error (MAE)
- Mean Squared Error (MSE)
- Root Mean Squared Error (RMSE)
- R^2 Score

Hyperparameter Tuning Details

1. **Random Forest**:
   - max_depth: [5, 10, 15]
   - n_estimators: [100, 200, 500]
   - min_samples_split: [3, 5, 10]
2. **XGBoost**:
   - learning_rate: [0.5, 0.1, 0.2]
   - n_estimators: [300]
   - max_depth: [3]
   - other parameters for fine control
3. **Gradient Boosting**:
   - max_depth: [12, 15, 20]
   - n_estimators: [200, 300, 1000]
   - learning_rate: [0.001, 0.01, 0.1]

Ensemble Methods Details

1. **Voting Regressor**:
   - Combined best models: GBR, XGBoost, Ridge
   - Weighted voting: [2, 3, 1]
   - Better generalization than individual models
2. **Stacking Regressor**:
   - Base estimators: GBR, XGBoost, CatBoost, LGBM, Random Forest
   - Final estimator: Voting Regressor
   - Best overall performance

Performance Results

- Voting Regressor achieved best performance

- Stacking Regressor second best
- Among individual models:
  1. CatBoost
  2. XGBoost
  3. LGBM
  4. Random Forest
  5. Gradient Boosting
  6. Ridge
  7. Linear Regression

Key Advantages of Final Model

- Combines strengths of multiple models
- Reduces overfitting through ensemble approach
- Better generalization on unseen data
- Robust to different types of features

Different model's scores...

	Model	Mean Absolute Error	Mean Squared Error	Root Mean Squared Error	R2 Score
0	LinearRegression	0.083423	0.017723	0.133129	0.887847
1	RandomForestRegressor	0.094013	0.019620	0.140073	0.875843
2	XGBRegressor	0.076325	0.014855	0.121881	0.905999
3	Ridge	0.078876	0.015090	0.122842	0.904510
4	GradientBoostingRegressor	0.074417	0.014886	0.122009	0.905800
5	LGBMRegressor	0.083637	0.016371	0.127950	0.896404
6	CatBoostRegressor	0.071815	0.013239	0.115059	0.916227
7	VotingRegressor	0.071740	0.014013	0.118376	0.911326
8	StackingRegressor	0.079573	0.014425	0.120106	0.908716

Conclusion: Ames Housing Price Prediction Project

Key Findings

1. **Data Distribution**
- Sale prices showed right-skewed distribution
- Log transformation significantly improved model performance
- Engineered features demonstrated strong correlations with target variable
2. **Feature Importance**
- Most influential features:
  - Overall Quality
  - Total Area
  - Total Square Footage
  - Location (Neighborhood)
  - Building Type

3. **Model Performance**

- Best performing models (by MAE):
    1. Stacking Regressor
    2. Voting Regressor
    3. XGBoost
    4. CatBoost
    5. Random Forest

4. **Ensemble Advantages**

- Stacking approach improved accuracy by:
    - Combining strengths of multiple models
    - Reducing overfitting
    - Better handling of feature interactions
- Weighted voting enhanced prediction stability

5. **Feature Engineering Impact**

- Created features showed significant correlation:
    - totalarea: Strong positive correlation
    - totalbaths: Moderate positive correlation
    - houseage: Negative correlation
    - Combined features improved model performance

Project Achievements

- Successfully developed a robust price prediction model
- Achieved high R^2 score and low errors with ensemble methods
- Created effective feature engineering pipeline
- Implemented comprehensive data preprocessing strategy

Future Improvements

1. **Model Enhancements**

- Fine-tune hyperparameters further
- Experiment with different ensemble combinations
- Implement neural network approaches

2. **Feature Engineering**

- Include temporal market trends

3. **Deployment Considerations**

- Build API for model serving
- Create monitoring system for model performance
- Implement regular retraining pipeline

The final model has been saved as *ames_house_prediction_model.joblib* for future use and deployment.