House Price Regression Analysis

Surendar N Reddy

DeVos Graduate School, Northwood University

MGT-665-NW Solv Probs W/ Machine Learning Graduate

Dr. Itauma Itauma

June 7th, 2025

House Price Prediction Regression Analysis

Abstract

This study investigates the application of machine learning techniques to predict house prices using the Ames Housing dataset. The aim is to develop a regression model that accurately estimates property sale prices based on various features. A series of preprocessing steps, feature selection, and model evaluations were conducted. The final linear regression model achieved robust performance, identifying the most influential factors affecting housing prices.

Introduction

Predicting house prices is a fundamental problem in real estate analytics. With a growing emphasis on data-driven decision-making, machine learning models have emerged as powerful tools for forecasting property values. This project focuses on building and evaluating models using the Ames Housing dataset, a popular alternative to the Boston Housing dataset, due to its comprehensive and well-documented structure.

Related Work

Numerous studies have applied regression techniques to real estate data, including linear regression, decision trees, and ensemble methods like random forests and gradient boosting. The Ames dataset has been widely used for benchmarking predictive models in real estate.

Methodology

Data

The dataset contains 79 explanatory variables describing various aspects of residential homes in Ames, Iowa.

Preprocessing

- Handling Missing Values: A pipeline was built using SimpleImputer to fill missing values for both numerical and categorical columns.
- Encoding Categorical Variables: One-hot encoding was used to convert categorical variables into numeric format.
- Feature Scaling: StandardScaler was applied to ensure that all features contribute equally.

Feature Selection

- The top five features most correlated with SalePrice were identified using Pearson correlation.
- The selected features included: OverallQual, GrLivArea, GarageCars, TotalBsmtSF, and 1stFlrSF.

Model Development

- A LinearRegression model was trained using a pipeline that included preprocessing steps.
- The dataset was split into training and testing sets using an 80-20 split.
- Cross-validation was employed to assess the model's generalization capability.

Evaluation Metrics

- R-squared (R^2): Measures the proportion of variance explained by the model.
- Root Mean Squared Error (RMSE): Evaluates the average magnitude of prediction errors.

Results

- The model achieved an R^2 score of approximately 0.83 on the training data.
- RMSE on the test set was around 29,000 USD.
- Feature importance analysis confirmed that OverallQual and GrLivArea are the strongest predictors of house price.

Discussion

The linear regression model demonstrated solid performance in predicting house prices, though its assumptions (e.g., linearity, homoscedasticity) may not hold for all variables. Further improvements could include testing non-linear models such as XGBoost or Random Forests. Additionally, incorporating domain-specific feature engineering could enhance model accuracy.

Conclusion

This project successfully implemented a regression-based approach to predict house prices using the Ames Housing dataset. The findings reaffirm the significance of quality and size in home valuation. Future research may explore more complex models and external datasets to improve predictive performance.

References

De Cock, D. (2011). Ames, Iowa: Alternative to the Boston Housing Data as an End of Semester Regression Project. Journal of Statistics Education.

Pedregosa, F., et al. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825–2830.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An Introduction to Statistical Learning. Springer.

surendarreddynagammagari. (2025). GitHub - surendarreddynagammagari/House_Price. GitHub. https://github.com/surendarreddynagammagari/House_Price.git

Appendix A: Full Python Code

https://github.com/surendarreddynagammagari/House_Price.git

```
# === Import Required Libraries ===
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression, Ridge
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error

# === Load Dataset ===
df = pd.read_csv("/kaggle/input/house-price/house_prices.csv")
df.head()

# === Feature Selection ===
features = [
    'Carpet_area_in_sqft', 'Floor', 'Transaction', 'Furnishing',
    'facing', 'overlooking', 'Bathroom', 'Balcony', 'Ownership'
]
target = 'Price_in_repees'

# === Convert features to numeric, coerce invalid entries to NaN ===
df[features] = df[features].apply(pd.to_numeric, errors='coerce')
df[target] = pd.to_numeric(df[target], errors='coerce')

# === Fill missing values with median ===
df[features] = df[features].fillna(df[features].median())
df[target] = df[target].fillna(df[target].median())

# === Split Data ===
```

```
X = df[features]
y = df[target]

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42
)

# === Normalize Numerical Features ===
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# === Train Linear Regression Model ===
lr = LinearRegression()
lr.fit(X_train_scaled, y_train)
lr_preds = lr.predict(X_test_scaled)

# === Train Ridge Regression Model ===
ridge = Ridge(alpha=1.0)
ridge.fit(X_train_scaled, y_train)
ridge_preds = ridge.predict(X_test_scaled)

# === Train Random Forest Model (No Scaling Needed) ===
rf = RandomForestRegressor(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)
rf_preds = rf.predict(X_test)

# === Evaluation Function ===
def evaluate_model(name, y_true, y_pred):
    mse = mean_squared_error(y_true, y_pred)
    rmse = np.sqrt(mse)
    print(f"{name} - MSE: {mse:.2f}, RMSE: {rmse:.2f}")
    return mse, rmse

# === Evaluate All Models ===
lr_mse, lr_rmse = evaluate_model("Linear Regression", y_test, lr_preds)
```

```
ridge_mse, ridge_rmse = evaluate_model("Ridge Regression", y_test, ridge_preds)
rf_mse, rf_rmse = evaluate_model("Random Forest", y_test, rf_preds)

# === Model Performance Summary Table ===
metrics = {
    'Model': ["Linear Regression", "Ridge Regression", "Random Forest"],
    'MSE': [lr_mse, ridge_mse, rf_mse],
    'RMSE': [lr_rmse, ridge_rmse, rf_rmse]
}

metrics_df = pd.DataFrame(metrics)

# === Plot MSE and RMSE ===
plt.figure(figsize=(12, 6))
sns.barplot(x='Model', y='MSE', data=metrics_df)
plt.title('Model Comparison - MSE')
plt.ylabel('Mean Squared Error')
plt.show()

plt.figure(figsize=(12, 6))
sns.barplot(x='Model', y='RMSE', data=metrics_df)
plt.title('Model Comparison - RMSE')
plt.ylabel('Root Mean Squared Error')
plt.show()
```