📝 Project Report: Ames Housing Price Prediction

Author: Raghav Agarwal
Date: 01-07-2003
Repository: https://github.com/RaghavAgarwal-01/house-price-prediction.git

---

1. Introduction

This project predicts house sale prices in Ames, Iowa, using the Ames Housing dataset. We performed thorough data cleaning, feature engineering, and trained a regularized regression model (LassoCV) for robust price predictions.

---

2. Data Cleaning

- **Columns with missing values**: Handled using median/mode imputations or dropping where necessary.
- **Categorical features**: One-hot encoded.
- **Numerical features**: Standard-scaled for model performance.

---

3. Exploratory Data Analysis (EDA)

- **Correlation Analysis** identified features highly correlated with SalePrice (e.g., OverallQual, GrLivArea, GarageCars).
- **Visualization** of target variable, feature distributions, and missing data patterns guided data cleaning and feature selection.

---

4. Feature Engineering

- Encoded categorical features with OneHotEncoder.
- Scaled numerical features with StandardScaler.
- Built a unified pipeline to process the data consistently.

---

5. Model Building

- Tested **Linear Regression**, **Ridge**, and **Lasso** regularization techniques.
- Selected LassoCV as the final model due to superior generalization on test data.
- Tuned the regularization parameter (alpha) automatically with LassoCV.

---

6. Final Results

- **Train RMSE:** 35440.25
- **Test RMSE:** 38116.38
- **Train R^2:** 0.7894
- **Test R^2:** 0.8106
- Final model saved as final_pipeline.pkl for deployment.

---

7. Prediction Example

For a single input sample, predicted sale price was:

```less
Predicted Sale Price: [237817.87]
```

---

8. Conclusion

- Achieved a reliable predictive model using a well-regularized Lasso regression pipeline.
- Generalization gap between train and test scores indicates a reasonable fit without significant overfitting.
- Ready for deployment or further improvement.

---

9. How to Run the Project

1. Install dependencies from requirements.txt.
2. Load and preprocess data using eda.ipynb.
3. Train the pipeline with train_model.ipynb.
4. Make predictions with predict.ipynb.
5. Use saved pipeline final_pipeline.pkl for deployment.

---

10. Future Improvements

- Perform advanced feature selection (e.g., SHAP values).
- Incorporate tree-based models (e.g., XGBoost) for potentially higher accuracy.
- Build a web app (e.g., Streamlit) for user-friendly prediction interface.