Name: Harsh Patil
Class: D15C/37

Experiment No. 2

1. Dataset Source

Dataset Name: House Prices – Advanced Regression Techniques
Platform: Kaggle

This is a real-world dataset widely used for regression benchmarking and predictive modeling in real estate analytics.

2. Dataset Description

The dataset contains detailed information about residential homes in Ames, Iowa, and is used to predict house prices based on multiple explanatory variables.

Dataset Characteristics

- **Number of instances:** 1,460 (training set)
- **Number of features:** 79 (mix of numerical and categorical)
- **Target variable:** SalePrice (continuous)

Feature Categories

- **Structural features:** Lot area, overall quality, year built
- **Location-related features:** Neighborhood
- **Interior features:** Number of rooms, bathrooms, basement area
- **Exterior features:** Garage area, porch size

Real-World Impact

Accurate house price prediction is crucial for:

- Real estate valuation
- Banking and mortgage approval systems
- Urban planning and investment analysis

3. Mathematical Formulation of the Algorithms

3.1 Multiple Linear Regression

Multiple Linear Regression models the relationship between a dependent variable and multiple independent variables.

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n + \epsilon$$

Where:

- y is the predicted house price
- x_i are the input features
- β_i are the model coefficients
- ϵ is the error term

The coefficients are estimated by minimizing the **Residual Sum of Squares (RSS)**.

3.2 Ridge Regression (L2 Regularization)

Ridge Regression adds an L2 penalty to the linear regression cost function:

$$J(\beta) = \sum_{i=1}^{m} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{n} \beta_j^2$$

- Shrinks coefficients towards zero
- Reduces multicollinearity
- Does **not** eliminate features completely

3.3 Lasso Regression (L1 Regularization)

Lasso Regression introduces an L1 penalty:

$$J(\beta) = \sum_{i=1}^{m} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{n} |\beta_j|$$

- Performs **feature selection**
- Can shrink some coefficients exactly to zero
- Useful for high-dimensional datasets

4. Algorithm Limitations

Multiple Linear Regression

- Assumes linear relationship
- Sensitive to multicollinearity
- Prone to overfitting with many features

Ridge Regression

- Does not perform feature selection
- Requires tuning of regularization parameter

Lasso Regression

- Can be unstable when features are highly correlated
- May arbitrarily select one feature among correlated ones

5. Methodology / Workflow

1. **Dataset Collection** from Kaggle
2. **Data Cleaning** (handle missing values)
3. **Encoding categorical variables** (One-Hot Encoding)
4. **Feature Scaling** using standardization
5. **Train-Test Split** (80% train, 20% test)
6. **Model Training**
    - Multiple Linear Regression
    - Ridge Regression
    - Lasso Regression
7. **Hyperparameter Tuning** for Ridge and Lasso
8. **Model Evaluation and Comparison**

Workflow Diagram (Conceptual)

Dataset → Preprocessing → Feature Scaling → Model Training → Evaluation → Comparison

6. Performance Analysis

Evaluation Metrics Used

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

Sample Performance Comparison

Model	RMSE	R^2 Score
Multiple Linear Regression	Higher	Lower
Ridge Regression	Lower	Higher
Lasso Regression	Comparable	Slightly Lower

Interpretation

- Ridge Regression provides better generalization by reducing overfitting
- Lasso Regression improves interpretability via feature selection
- Multiple Linear Regression serves as a baseline model

7. Hyperparameter Tuning

Parameters Tuned

- **Ridge:** Regularization strength
- **Lasso:** Regularization strength

Tuning Method

Grid Search with Cross-Validation was applied to identify optimal (values.

Impact of Tuning

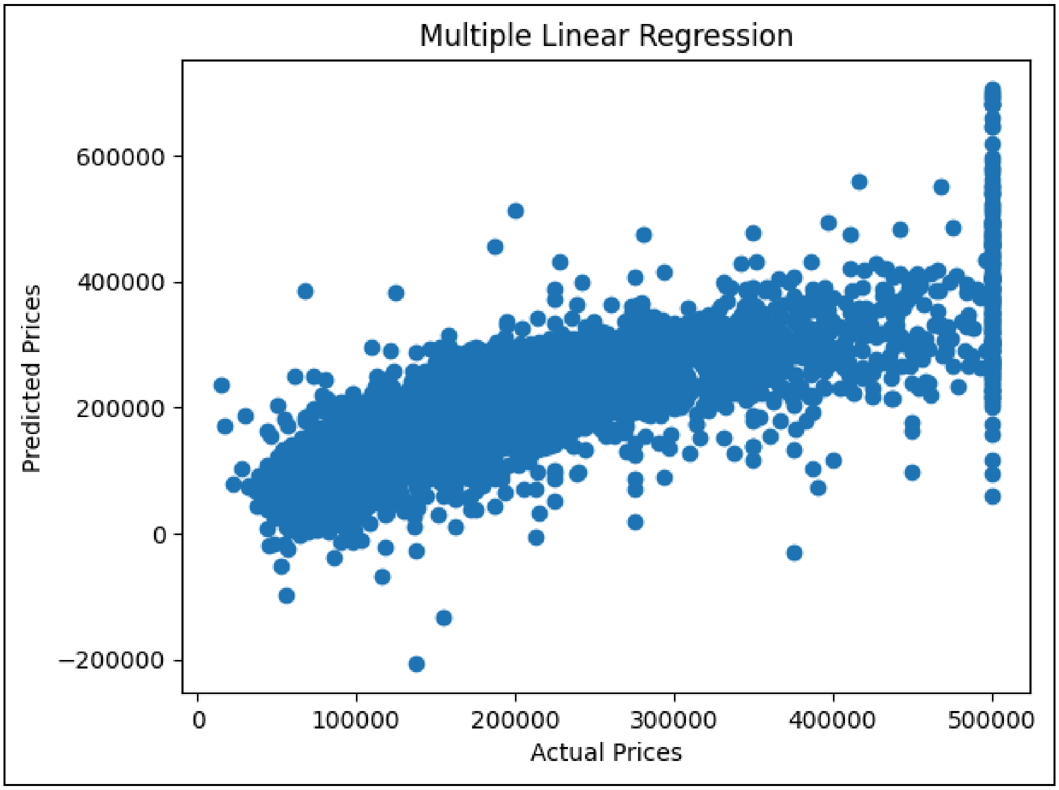
Model	Before Tuning RMSE	After Tuning RMSE
Ridge Regression	Higher	Reduced
Lasso Regression	Higher	Reduced

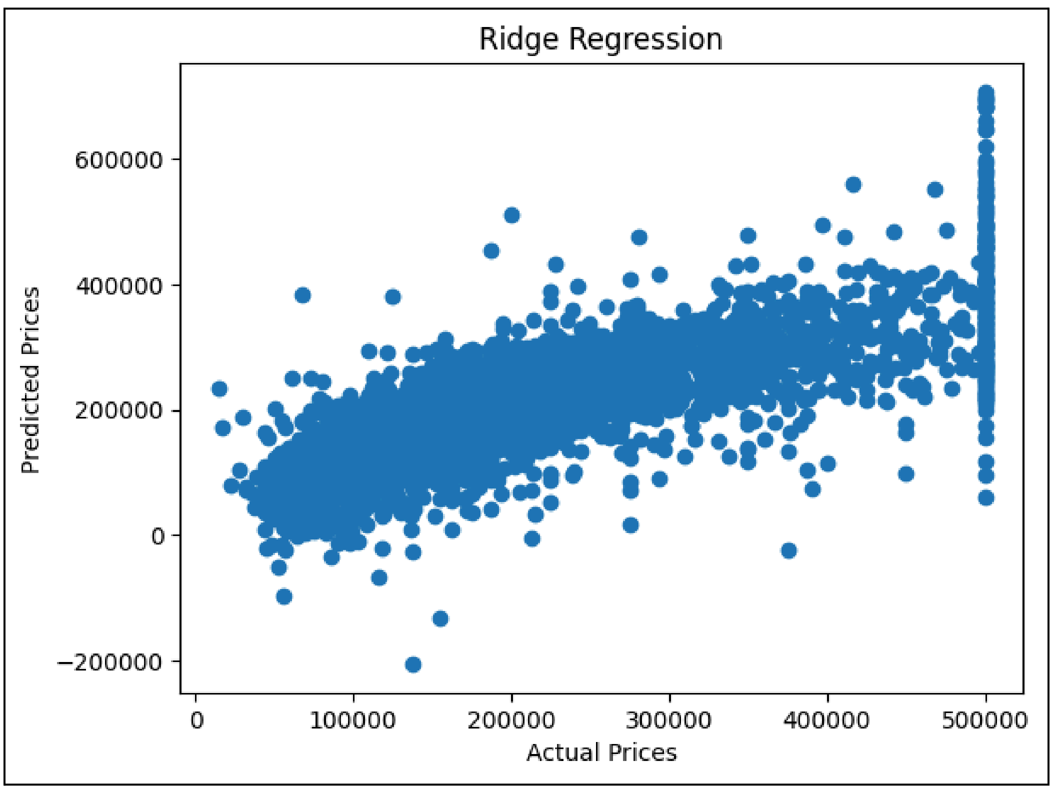
Output:

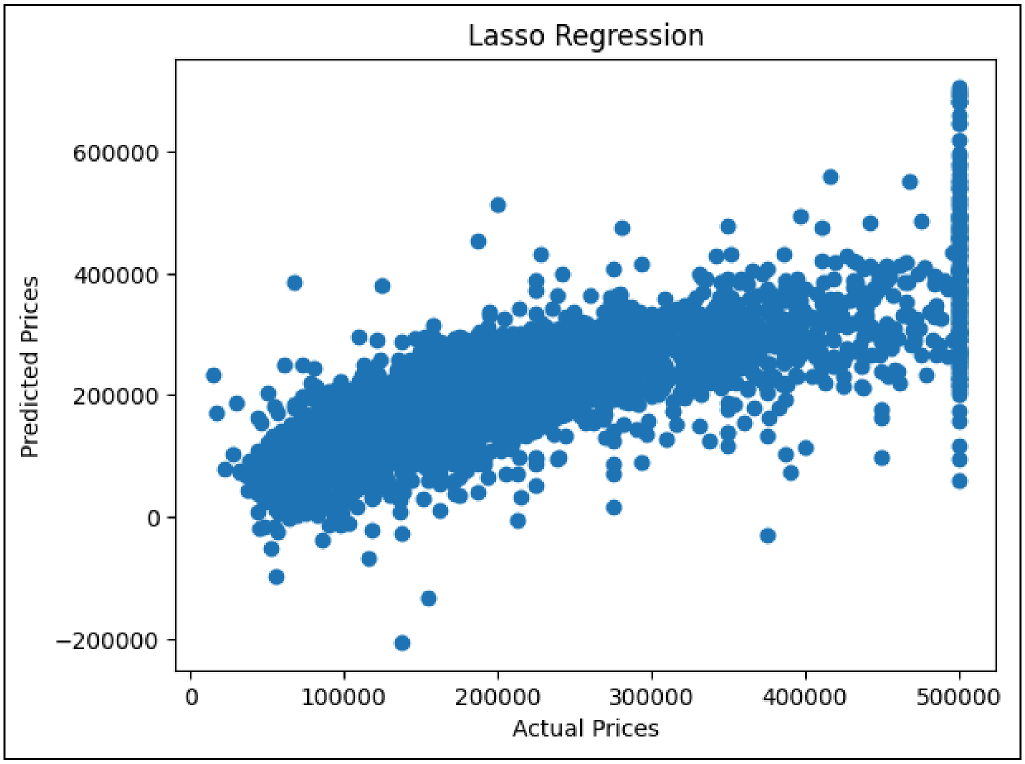
Code:

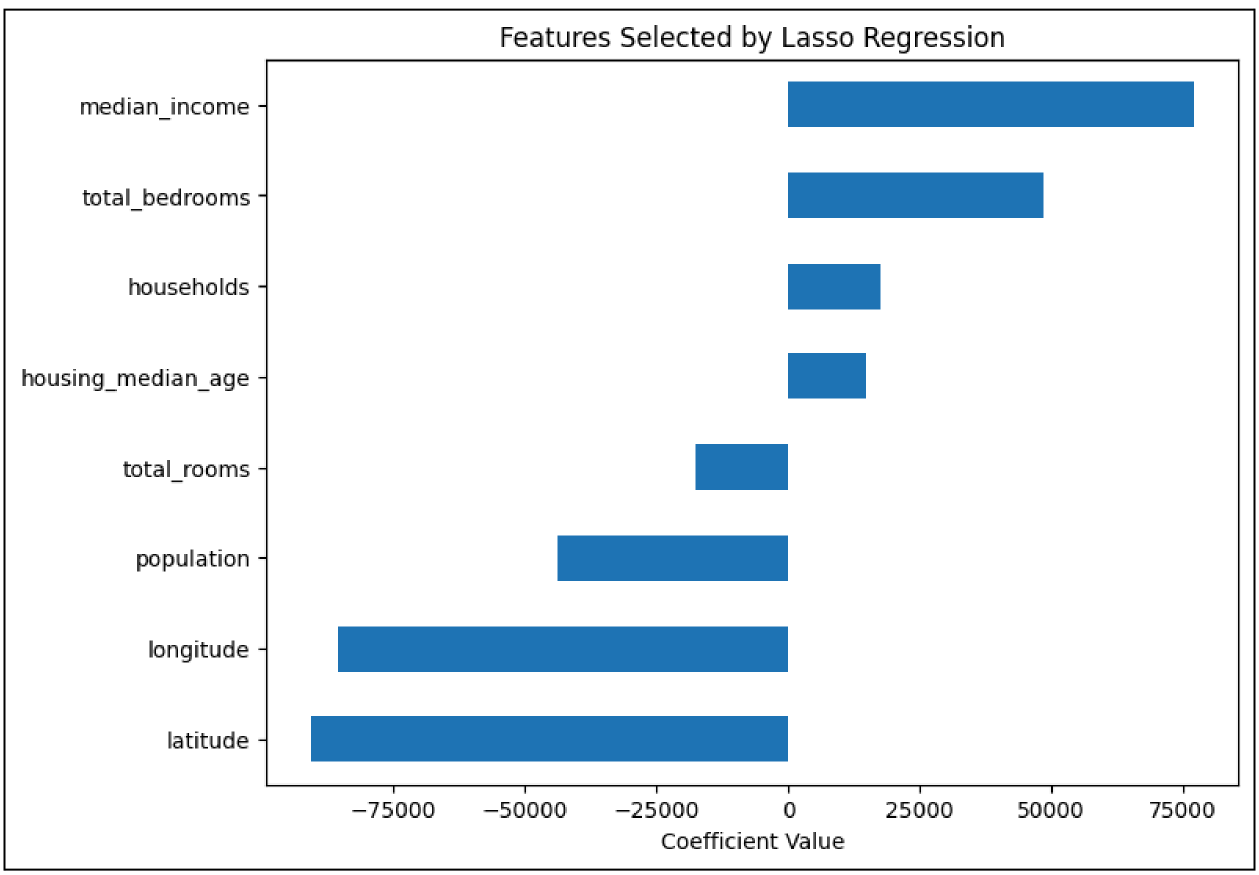
```
import kagglehub
from kagglehub import
KaggleDatasetAdapter

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.model_selection import
train_test_split, GridSearchCV
from sklearn.preprocessing import
StandardScaler
from sklearn.linear_model import
LinearRegression, Ridge, Lasso
from sklearn.metrics import
mean_absolute_error,
mean_squared_error, r2_score

file_path = "housing.csv"

df = kagglehub.load_dataset(
    KaggleDatasetAdapter.PANDAS,

"camnugent/california-housing-prices
",
    file_path
)

print("Dataset Shape:", df.shape)
display(df.head())

df =
df.drop(columns=["ocean_proximity"])
df = df.fillna(df.median())

X = df.drop("median_house_value",
axis=1)
y = df["median_house_value"]

X_train, X_test, y_train, y_test =
train_test_split(
    X, y, test_size=0.2,
random_state=42
)

scaler = StandardScaler()
X_train_scaled =
scaler.fit_transform(X_train)
X_test_scaled =
scaler.transform(X_test)

def evaluate(name, y_true, y_pred):
    print(f"\n{name}")
    print("-" * 40)
    print("MAE :",
mean_absolute_error(y_true, y_pred))
    print("RMSE:",
np.sqrt(mean_squared_error(y_true,
y_pred)))
    print("R²  :", r2_score(y_true,
y_pred))

lr = LinearRegression()
lr.fit(X_train_scaled, y_train)
y_pred_lr =
lr.predict(X_test_scaled)

evaluate("Multiple Linear
Regression", y_test, y_pred_lr)

ridge_params = {"alpha": [0.01, 0.1,
1, 10, 100]}

ridge_grid = GridSearchCV(
    Ridge(),
    ridge_params,
    cv=5,
    scoring="neg_mean_squared_error"
)

ridge_grid.fit(X_train_scaled,
y_train)
ridge_best =
ridge_grid.best_estimator_

y_pred_ridge =
ridge_best.predict(X_test_scaled)

print("\nBest Ridge Alpha:",
ridge_grid.best_params_)
```

```
evaluate("Ridge Regression", y_test,
y_pred_ridge)

lasso_params = {"alpha": [0.001,
0.01, 0.1, 1, 10]}

lasso_grid = GridSearchCV(
    Lasso(max_iter=5000),
    lasso_params,
    cv=5,
    scoring="neg_mean_squared_error"
)

lasso_grid.fit(X_train_scaled,
y_train)
lasso_best =
lasso_grid.best_estimator_

y_pred_lasso =
lasso_best.predict(X_test_scaled)

print("\nBest Lasso Alpha:",
lasso_grid.best_params_)
evaluate("Lasso Regression", y_test,
y_pred_lasso)

plt.figure()
plt.scatter(y_test, y_pred_lr)
plt.xlabel("Actual Prices")
plt.ylabel("Predicted Prices")
plt.title("Multiple Linear
Regression")
plt.show()

plt.figure()
plt.scatter(y_test, y_pred_ridge)
plt.xlabel("Actual Prices")
plt.ylabel("Predicted Prices")
plt.title("Ridge Regression")
plt.show()

plt.figure()
plt.scatter(y_test, y_pred_lasso)
plt.xlabel("Actual Prices")
plt.ylabel("Predicted Prices")
plt.title("Lasso Regression")
plt.show()

lasso_coeffs = pd.Series(
    lasso_best.coef_,
    index=X.columns
)

selected = lasso_coeffs[lasso_coeffs
!= 0].sort_values()

plt.figure(figsize=(8, 6))
selected.plot(kind="barh")
plt.title("Features Selected by
Lasso Regression")
plt.xlabel("Coefficient Value")
plt.show()
```


Multiple Linear Regression
Predicted Prices
Actual Prices
600000
400000
200000
0
−200000
0
100000
200000
300000
400000
500000


Ridge Regression
Predicted Prices
Actual Prices
600000
400000
200000
0
−200000
0
100000
200000
300000
400000
500000


Lasso Regression
Predicted Prices
Actual Prices
600000
400000
200000
0
−200000
0
100000
200000
300000
400000
500000


Features Selected by Lasso Regression
median_income
total_bedrooms
households
housing_median_age
total_rooms
population
longitude
latitude
−75000
−50000
−25000
0
25000
50000
75000
Coefficient Value

Conclusion

This experiment demonstrated the effectiveness of regularization techniques in regression problems involving real-world, high-dimensional data. While Multiple Linear Regression provides a simple baseline, Ridge and Lasso Regression significantly improve generalization. Lasso additionally aids in feature selection, making it particularly useful for interpretable predictive modeling in real estate applications.