Phase-3

Student Name: Roy bennar.s

Register Number: 421223104069

Institution: karpaga Vinayaga collage of engineering

Department: BE - Computer Science and Engineering

Date of Submission: 26.04.2025

Githup Respository Link:
https://github.com/roybenar7/Forecasting-House-Prices-Accurately-Using-Smart-Regression-Techniques-in-Data-Science

---

PROJECT TITLE :
Forecasting House Prices Accurately Using Smart Regression Techniques in Data Science

1. Problem Statement

Accurately predicting house prices is a challenging task that holds great importance for buyers, sellers, and real estate investors. Traditional methods often fall short in capturing complex, non-linear relationships among various factors such as location, size, amenities, and market trends. This project aims to address these challenges by leveraging advanced regression techniques in data science to develop a robust and highly accurate house price prediction model.

2. Objectives of the Project

Build predictive models capable of accurately estimating house prices based on various features.

Explore and implement smart regression techniques beyond traditional linear models.

Provide interpretable insights into the key factors influencing house prices.

Develop a user-friendly application where users can input house details and receive price predictions.

3. Scope of the Project

Features:

Extensive feature analysis (location, number of rooms, area, amenities)

Advanced regression models (ensemble methods, regularized regression)

Limitations:

Focused on datasets from specific regions (generalization to all markets may require retraining)

Static historical data used (real-time market fluctuations out of scope)

Constraints:

Only publicly available datasets will be used

Focus on explainable machine learning models

4. Data Sources

Dataset: House Prices: Advanced Regression Techniques

Source: Kaggle
(https://www.kaggle.com/c/house-prices-advanced-regression-techniques)

Type: Public, static

Description: Contains detailed records of residential houses in Ames, Iowa, including features like size, location, year built, and sale price.

5. High-Level Methodology

Data Collection:

Dataset will be downloaded from Kaggle.

Data Cleaning:

Handle missing values, correct anomalies

Encode categorical variables and normalize numerical data

Exploratory Data Analysis (EDA):

Correlation heatmaps, feature importance plots

Outlier detection and treatment

Feature Engineering:

Polynomial features

Interaction terms

Dimensionality reduction (PCA)

Model Building:

Models: Linear Regression, Ridge, Lasso, XGBoost, Random Forest Regressor

Justification: Ensemble and regularized models manage complexity and overfitting

Model Evaluation:

Metrics: RMSE (Root Mean Squared Error), R^2 Score

Validation strategy: K-Fold Cross Validation

Visualization & Interpretation:

Residual plots, learning curves

SHAP values for model interpretability

Deployment:

Build a lightweight web app using Streamlit where users can input house features and view predicted prices

6. Tools and Technologies

Programming Language: Python

Notebook/IDE: Google Colab, Jupyter Notebook

Libraries: pandas, numpy, scikit-learn, matplotlib, seaborn, xgboost, streamlit

Optional Deployment Tools: Streamlit or Flask for web app

7. Source Code

```
  # Import necessary libraries
import pandas as pd
import numpy as np
```

```
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score

# Load dataset (California Housing is a good modern substitute for Boston dataset)
data = fetch_california_housing()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['MedHouseVal'] = data.target

# 1. Exploratory Data Analysis
print(df.head())
print(df.describe())
sns.pairplot(df.sample(500), diag_kind='kde')
plt.show()

# 2. Data Preprocessing
X = df.drop('MedHouseVal', axis=1)
y = df['MedHouseVal']

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2,
random_state=42)

# 3. Regression Models

# Linear Regression
```

```
lr = LinearRegression()
lr.fit(X_train, y_train)
y_pred_lr = lr.predict(X_test)

# Ridge Regression with tuning
ridge = Ridge()
params_ridge = {'alpha': [0.01, 0.1, 1, 10, 100]}
ridge_cv = GridSearchCV(ridge, params_ridge, cv=5)
ridge_cv.fit(X_train, y_train)
y_pred_ridge = ridge_cv.predict(X_test)

# Lasso Regression with tuning
lasso = Lasso()
params_lasso = {'alpha': [0.01, 0.1, 1, 10]}
lasso_cv = GridSearchCV(lasso, params_lasso, cv=5)
lasso_cv.fit(X_train, y_train)
y_pred_lasso = lasso_cv.predict(X_test)

# Random Forest Regressor
rf = RandomForestRegressor(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)
y_pred_rf = rf.predict(X_test)

# 4. Evaluation Function
def evaluate_model(name, y_true, y_pred):
    mse = mean_squared_error(y_true, y_pred)
    rmse = np.sqrt(mse)
    r2 = r2_score(y_true, y_pred)
    print(f"{name}:\n  RMSE: {rmse:.3f}\n  R² Score: {r2:.3f}\n")

# 5. Evaluate All Models
evaluate_model("Linear Regression", y_test, y_pred_lr)
evaluate_model("Ridge Regression", y_test, y_pred_ridge)
evaluate_model("Lasso Regression", y_test, y_pred_lasso)
evaluate_model("Random Forest Regressor", y_test, y_pred_rf)
```

```
# 6. Plot Predictions vs Actual
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred_rf, alpha=0.3, label="Random Forest")
plt.plot([y.min(), y.max()], [y.min(), y.max()], 'r--')
plt.xlabel("Actual")
plt.ylabel("Predicted")
plt.title("Random Forest: Predicted vs Actual")
plt.legend()
plt.grid(True)
plt.show()
```

8. Team Members and Roles

1. Rakesh c - Bussiness analyst

2.simon benen .d - Project Lead

3.thiruselvam N-Data Analysis and Cleaning