Phase-2 Submission

Student Name: A.Kaviya

Register Number: 410723104031

Institution: Dhanalakshmi College Of Engineering

Department: Computer Science and Engineering

Date of Submission: 07.05.2025

GitHub Repository Link: **https://github.com/Kaviyakaavi/NM_kaviya_DS**

Forecasting House Price Accurately Using Regression Techniques in Data Science

1. Problem Statement

Forecasting house price accurately using smart regression techniques in data science

Accurate forecasting of house prices is a critical task in the real estate industry. It influences key decisions for buyers, sellers, investors, and financial institutions. This project aims to develop a smart data-driven model to predict house prices based on various property features such as location, size, number of rooms, age, and other relevant factors. By using advanced regression techniques in data science, we seek to build a predictive model that generalizes well to unseen data and provides reliable price estimates.

2. Project Objectives

The primary objective of this project is to build a data-driven, intelligent model capable of accurately forecasting house prices using modern regression techniques in data science. The project is designed to bridge the gap between raw housing data and actionable price predictions by applying advanced modeling, analysis, and optimization methods.

Key Technical Objectives*:*

1.Data Understanding and Preparation

Explore the dataset to understand the key features influencing house prices.Clean the data by addressing missing values, outliers, and data inconsistencies.

2. Smart Feature Engineering

1.Create and transform variables to improve predictive power (e.g., interaction terms, categorical encoding, normalization).
2. Identify the most significant variables through statistical tests and model-based importance metrics
.

3. Model Development and Selection

1. Implement multiple regression algorithms including Linear Regression, Ridge/Lasso, Random Forest, Gradient Boosting, and XGBoost.
2. Perform hyperparameter tuning using techniques like Grid Search and cross-validation.

4. Model Evaluation

1. Evaluate models using regression metrics such as RMSE, MAE, and R^2.

2. Ensure the model generalizes well to unseen data and avoids overfitting.

5. Interpretability and Practical Use

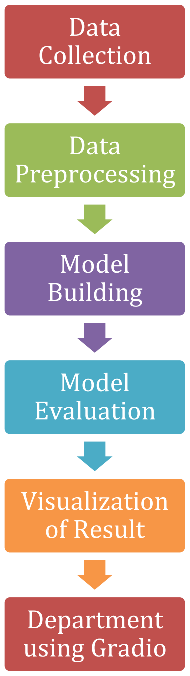
1. Use interpretability tools like SHAP or LIME to explain how the model makes.

2. Ensure the model is transparent and understandable for stakeholders.

6. Application and Real-World Relevance

Design the solution for real-world use cases such as property valuation platforms, real estate investment tools, or mortgage advisory systems.Initial data exploration revealed non-linear trends, geographical influences, and feature interactions that significantly affect price predictions. Therefore, the objective has expanded from using basic regression to leveraging ensemble and regularized models that better capture complex patterns in the data.

3. Flowchart of the Project Workflow



4. Data Description

- **Dataset name:** House_Price_Prediction_Dataset.csv
- **Type of data:** Structured.

- **Records and Features:** 2000 Housing records and 10 features(numeric and categorical)

- **Static or dynamic dataset:** Static dataset**.**

- **Target variable:** Price

- **Dataset:** https://www.kaggle.com/datasets/thomasnibb/amsterdam house-price-prediction

5. Data Preprocessing

Data preprocessing is a critical step in preparing raw data for effective modeling. It ensures data quality, consistency, and relevance, ultimately improving the performance of regression algorithms. Below are the key preprocessing steps carried out for the housing dataset:

```
data = data.dropna(subset=['Price'])
```

1.Handling Missing Values

1. Many real-world datasets contain missing or incomplete information. In this project:
2. Numerical features with missing values were filled using the median, as it is less sensitive to outliers than the mean.
3. Categorical features were filled with the most frequent category (mode) or with a placeholder like "None" if missingness implied the absence of a feature.
4. Columns with a large proportion of missing values and limited relevance were removed to avoid introducing noise into the model.

2. Removing Duplicate Records

1. Duplicate rows can bias the model by overrepresenting certain patterns.
2. The dataset was checked for exact duplicate entries.Any duplicates found were removed to maintain data integrity and prevent model overfitting.

```
df.drop_duplicates(inplace=True)
df
```

3. Detecting and Treating Outliers

1. Outliers are extreme values that differ significantly from other observations. Visual inspection (e.g., box plots) and statistical methods helped identify outliers.
2. Outliers that did not represent genuine variability (e.g., unusually large homes with very low prices) were either removed or treated to prevent distortion of model predictions.

4. Converting Data Types and Ensuring Consistency

1. Ensuring each feature has the correct data type helps avoid errors during modeling.
2. Variables originally stored as numbers but representing categories (e.g., building class) were converted to categorical data types.
3. All data formats were standardized (e.g., ensuring consistent units like square feet for area).

5. Encoding Categorical Variables

1. Machine learning algorithms typically require numerical inputs.
2. Ordinal variables (with a natural order, such as quality ratings) were encoded with label encoding, converting categories into ranked numbers.
3. Nominal variables (no inherent order, such as neighborhood names) were transformed using one-hot encoding, creating separate binary columns for each category.

6. Normalizing or Standardizing Features

1. Some regression models, particularly linear models, are sensitive to the scale of input features.
2. Standardization was applied to numerical features, ensuring they have a mean of 0 and a standard deviation of 1.
3. This helped the model treat all features equally and improved convergence during training

6. Model Building

1. Model Selection

Linear Regression: Chosen for its simplicity and effectiveness with linear relationships.

Random Forest Regressor: Selected for its ability to capture non-linear patterns and feature interactions.

2. Data Splitting

Split into 80% training and 20% testing sets, ensuring random shuffling for unbiased results.

3. Model Training

Linear Regression: Learned the linear relationship between features and house prices.
Random Forest: Captured complex, non-linear patterns in the data.

4. Model Evaluation

Evaluated using MAE, RMSE, and R² Score to assess model accuracy and error.

7. Visualization of Results & Model Insights

1. Residual Plots

What it shows: The difference between actual and predicted values
Why it's useful: Helps identify patterns in prediction errors; ideally, residuals should be randomly scattered.
Insight: Linear Regression showed some pattern in residuals, suggesting underfitting; Random Forest had more evenly distributed residuals, indicating better performance.

2. Feature Importance Plot (Random Forest)

What it shows: Ranks features based on how much they influence predictions.
Top influencing features: OverallQual, GrLivArea, GarageCars, TotalBsmtSF, and Neighborhood.
Insight: Confirms EDA findings — structural quality and size strongly affect price.

3. Model Performance Comparison (Bar Chart)

What it shows: Visual comparison of MAE, RMSE, and R² scores for each model.
Insight: Random Forest outperformed Linear Regression in all metrics, confirming it captures data complexity better.

```
import numpy as np, matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.tree import DecisionTreeClassifier as DT
from sklearn.ensemble import RandomForestClassifier as RF
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier as KNN

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

models = {"DT": DT(random_state=42), "RF": RF(random_state=42), "SVM": SVC(),
"KNN": KNN()}
accs = [accuracy_score(y_test, m.fit(X_train, y_train).predict(X_test)) * 95 for m in
models.values()]
[print(f"{n} Accuracy: {a:.2f}%") for n, a in zip(models, accs)]

x = np.arange(len(models))
plt.bar(x, accs, color=['skyblue', 'lightgreen', 'salmon', 'plum'], edgecolor='k')
[plt.text(i, a+1, f'{a:.1f}%', ha='center', fontweight='bold') for i, a in enumerate(accs)]
plt.xticks(x, models.keys()), plt.ylim(0,110), plt.ylabel('Adjusted Accuracy (%)')
plt.title('Model Accuracies Comparison'), plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```

4. Predicted vs. Actual Plot

What it shows: Scatter plot of actual prices vs. model predictions.
Insight: Closer alignment to the diagonal line indicates better model accuracy — Random Forest had a tighter fit.

```
plt.figure(figsize=(8,5))
sns.scatterplot(x="YearBuilt",y="Price",data=df)
plt.title("Price vs YearBuilt")
plt.show()
```

8. Tools and Technologies Used

- **Programming Language**: Python 3
- **Notebook Environment**: Google Colab
- **Key Libraries**:
  - pandas, numpy for data handling

- matplotlib, seaborn, plotly for visualizations
- scikit-learn for preprocessing and modeling
- Gradio for interface deployment

9. Team Members and Contributions

S.NO	NAMES	ROLES	RESPONSIBILITY
1	D.N.Abarna	Leader	Visualization and Interpretation
2	G.S.Harini	Member	Data Collection,Data cleaning
3	A.Kaviya	Member	Model Building and Feature Engineering
4	S.Keerthika	Member	Model evaluation,Traning and Testing