

## Forecasting House Prices Using Smart Regression Techniques

**Student Name:** S Mohamed umar

**Register Number:** 410623104073

**Institution:** Dhaanish Ahmed College Engineering

**Department:** Computer science and engineering

**Date of Submission:** 16-05-2025

---

### 1.Problem Statement

Accurate forecasting of house prices is crucial for stakeholders like real estate investors, buyers, and policy makers. This project aims to develop a predictive model using smart regression techniques, enabling highly accurate estimation of house prices based on various property, economic, and location features

### 2.Objectives of the Project

- Build a highly accurate model that predicts house prices.
- Understand which factors (like neighborhood, house size, etc.) impact prices the most.
- Create a simple tool that not only datascientists can use but, anyone can use to get predictions.

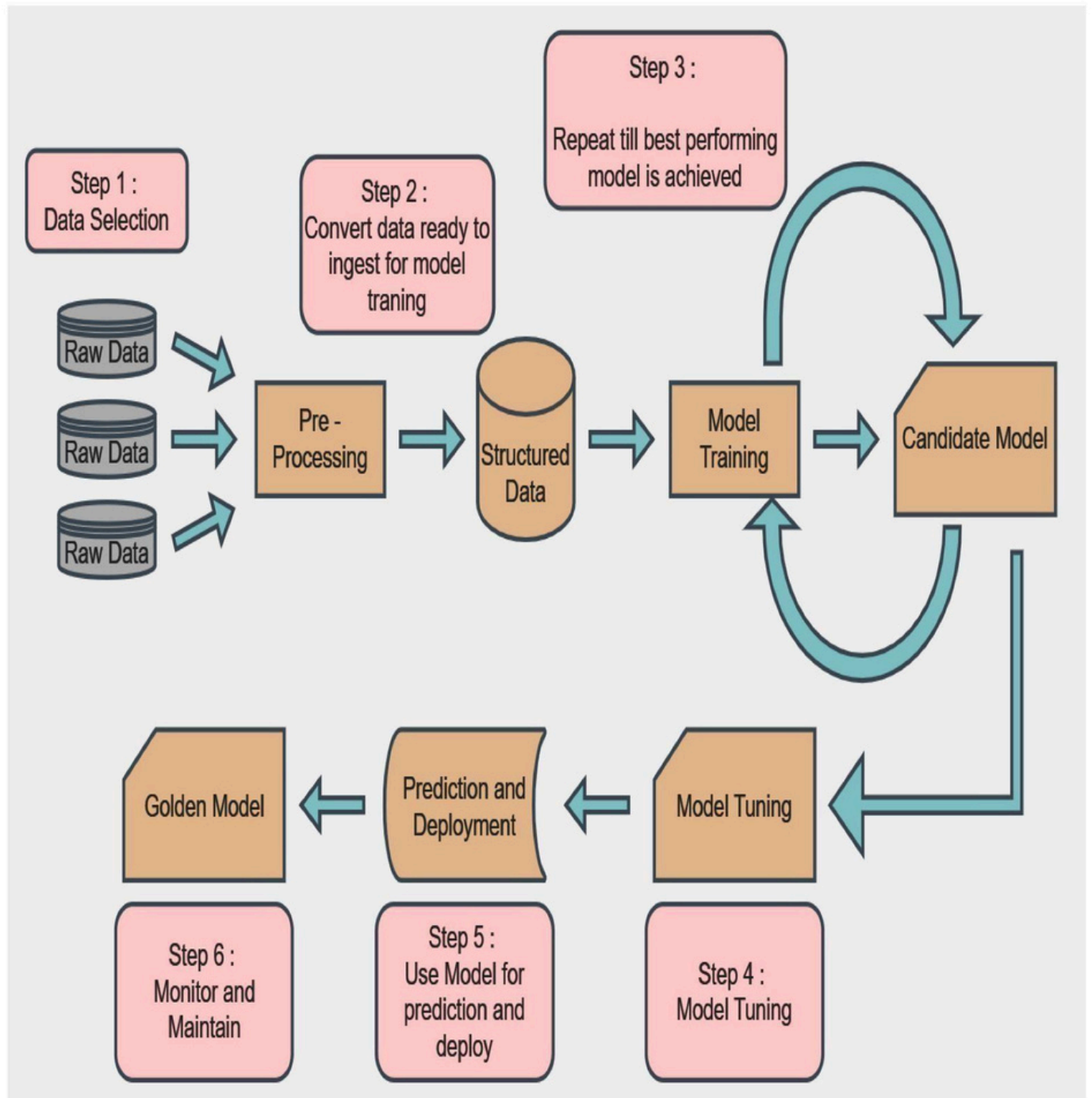
### 3.Scope of the Project

- We'll focus mainly on residential homes, not commercial properties.
- Start with data from one city or region, and later expand if needed.  
Use features like house size, number of bedrooms, proximity to schools, and local economic indicators.
- Try out different machine learning techniques — from simple Linear Regression to more advanced ones like Random Forests and XGBoost.
- We'll also build a clean, user-friendly dashboard where users can input house details and get a price prediction instantly.

### 4.Data Sources

- Public datasets from places like Kaggle (example: the Ames Housing Dataset).
- Real estate listing sites (like Zillow or [Realtor.com](https://www.realtor.com)).
- Government and city property records (for verified historical data)
- Maybe even scrape a little data ourselves if needed — responsibly, of course!

## Process Flowchart:



## 5. High-Level Methodology

Here's the big-picture plan:

1. Collect and prepare the data—clean it up, fill in missing values, and create new features that could help the model.
2. Explore the data — find patterns, outliers, and interesting relationships.
3. Build and test different models—see which regression techniques give us the best results.
4. Fine-tune the models—tweak the settings to squeeze out every bit of accuracy.
5. Deploy the best model—turn it into an dashboard so users can easily get house price estimates.

## 6. Tools and Technologies

- Programming Languages: Python
- Libraries/Frameworks: Scikit-learn, XGBoost, LightGBM, TensorFlow (optional for deep learning models)
- Data Visualization: Matplotlib, Seaborn, Plotly
- Data Processing: Pandas, NumPy

- Model Deployment: Flask/Django for web deployment, Streamlit for dashboards
- Version Control: Git and GitHub
- Environment: Jupyter Notebook, Google Colab, VS Code
- Cloud Platforms (optional for scalability): AWS, Azure

**Code:** import pandas as

pd import numpy as np

import seaborn as sns

import matplotlib.pyplot as

plt

from sklearn.model\_selection import train\_test\_split,

cross\_val\_score

from sklearn.preprocessing import StandardScaler from

sklearn.impute import SimpleImputer

from sklearn.linear\_model import Ridge, Lasso, LinearRegression

from sklearn.ensemble import RandomForestRegressor from

sklearn.metrics import mean\_squared\_error import xgboost as xgb

import lightgbm as lgb

# Load the dataset (replace with actual path or use Kaggle dataset)

```
data = pd.read_csv('train.csv')
```

# Drop outliers (example) data =

```
data[data['GrLivArea'] < 4500]
```

# Target variable y

```
= data['SalePrice']
```

```
X = data.drop(['Id', 'SalePrice'], axis=1)
```

# 1. Data Preprocessing

# Identify numerical and categorical columns

```
num_cols = X.select_dtypes(include=['int64', 'float64']).columns
```

```
cat_cols = X.select_dtypes(include=['object']).columns
```

# Impute missing values

```
num_imputer = SimpleImputer(strategy='median') cat_imputer
```

```
= SimpleImputer(strategy='most_frequent')
```

```
X[num_cols] = num_imputer.fit_transform(X[num_cols])
```

```
X[cat_cols] = cat_imputer.fit_transform(X[cat_cols])
```

# One-hot encoding

```
X = pd.get_dummies(X, drop_first=True)
```

# Feature scaling scaler

```
= StandardScaler()
```

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

# 2. Split the data

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

# 3. Define models models

```
= {  
    'Linear Regression': LinearRegression(),  
    'Lasso': Lasso(alpha=0.001),  
    'Ridge': Ridge(alpha=10),  
    'Random Forest': RandomForestRegressor(n_estimators=100),  
    'XGBoost': xgb.XGBRegressor(),  
    'LightGBM': lgb.LGBMRegressor()  
}
```

# 4. Train and evaluate models results

```
= {} for name, model in  
models.items():  
    model.fit(X_train, y_train)    preds =  
    model.predict(X_test)  
    rmse = np.sqrt(mean_squared_error(y_test, preds))  
    results[name] = rmse    print(f'{name} RMSE:  
{rmse:.2f}')
```

# 5. Visualize model performance plt.figure(figsize=(10,5))

```
sns.barplot(x=list(results.keys()), y=list(results.values()))  
plt.ylabel("RMSE")  
plt.title("Model Comparison - RMSE")  
plt.xticks(rotation=45) plt.tight_layout()  
plt.show()
```

## 7.Team Members and Roles

**1.Yusuf shejin** - Data Collection and Integration:

Responsible for sourcing datasets, connecting APIs,  
and preparing the initial dataset for analysis.dk

**2.Syed sharukh** - Data Cleaning and EDA: Cleans

and preprocesses data, performs exploratory analysis,  
and generates initial insights.



**3.Mohamed umar**-Feature Engineering and Modeling: Workson feature extraction and selection; develops and trains machine learning models.

**4.Tarik Ahamed** - Evaluation and Optimization: Tunes hyperparameters, validates models, and documents performance metrics.

**5.Mohamed danish** - Documentation and Presentation: Compiles reports, prepares visualizations, and handles presentation and optional deployment.

**6.Mohamed irfan**- Data Visualization:

Matplotlib, Seaborn, Plotly.