**Student Name: SreeDhivya prabaa R**

**Register Number:** 723923243044

**Institution:** Arjun College of Technology **Department: Artificial Intelligence and Data Science**

**Date of Submission: 17-05-2025**

**GitHub Repository Link:** [Update the projecyour GitHub Repository]

# 1.Problem Statement

**Forecasting House Prices using Smart Regression Techniques**

In real estate, understanding and predicting the price of houses is crucial for both buyers and sellers. House prices depend on a variety of factors, such as location, size, number of bedrooms, age of the property, economic conditions, and market trends. However, accurately predicting house prices is a complex task because many of these factors are interrelated and can change over time.

The goal of this project is to develop a model that uses advanced regression techniques to predict house prices based on different input factors. By using machine learning and smart regression methods, the model aims to find the best relationships between house features (like square footage, location, etc.) and their prices. The main challenges include:

1. **Identifying relevant factors**: Which features are most important for predicting prices?
2. **Handling complexity**: House prices can be influenced by multiple factors, some of which may interact in complex ways.
3. **Generalization**: The model should work well not just for the data it's trained on, but also for new, unseen data (e.g., future house listings).

**Why it’s important to solve this problem:**

1. **Helps Buyers and Sellers**: If you’re buying or selling a house, you want to know the right price. Predicting house prices helps both sides make fair deals.
2. **Real Estate Investors**: Investors need to know when and where to buy property to make a good profit. Accurate price predictions help them make better decisions. (e c t…)

# 2)Project Objectives

**1. Data Collection and Cleaning**

* Collect relevant datasets, such as property features (e.g., size, location, number of rooms, etc.) and historical price data.
* Clean the data by handling missing values, correcting errors, and preparing it for analysis.

**2. Identify Key Features for Prediction**

* Analyse and identify the most important factors (features) that influence house prices, such as square footage, neighbourhood, or age of the house.

**3. Model Selection and Training**

* Choose the most appropriate regression techniques (like linear regression, decision trees, or more advanced methods like random forests or gradient boosting) to model the relationship between the features and house prices.

**4. Model Evaluation and Optimization**

* Evaluate the model’s accuracy using performance metrics (e.g., RMSE, R-squared) and optimize it by tuning hyperparameters to improve predictions.

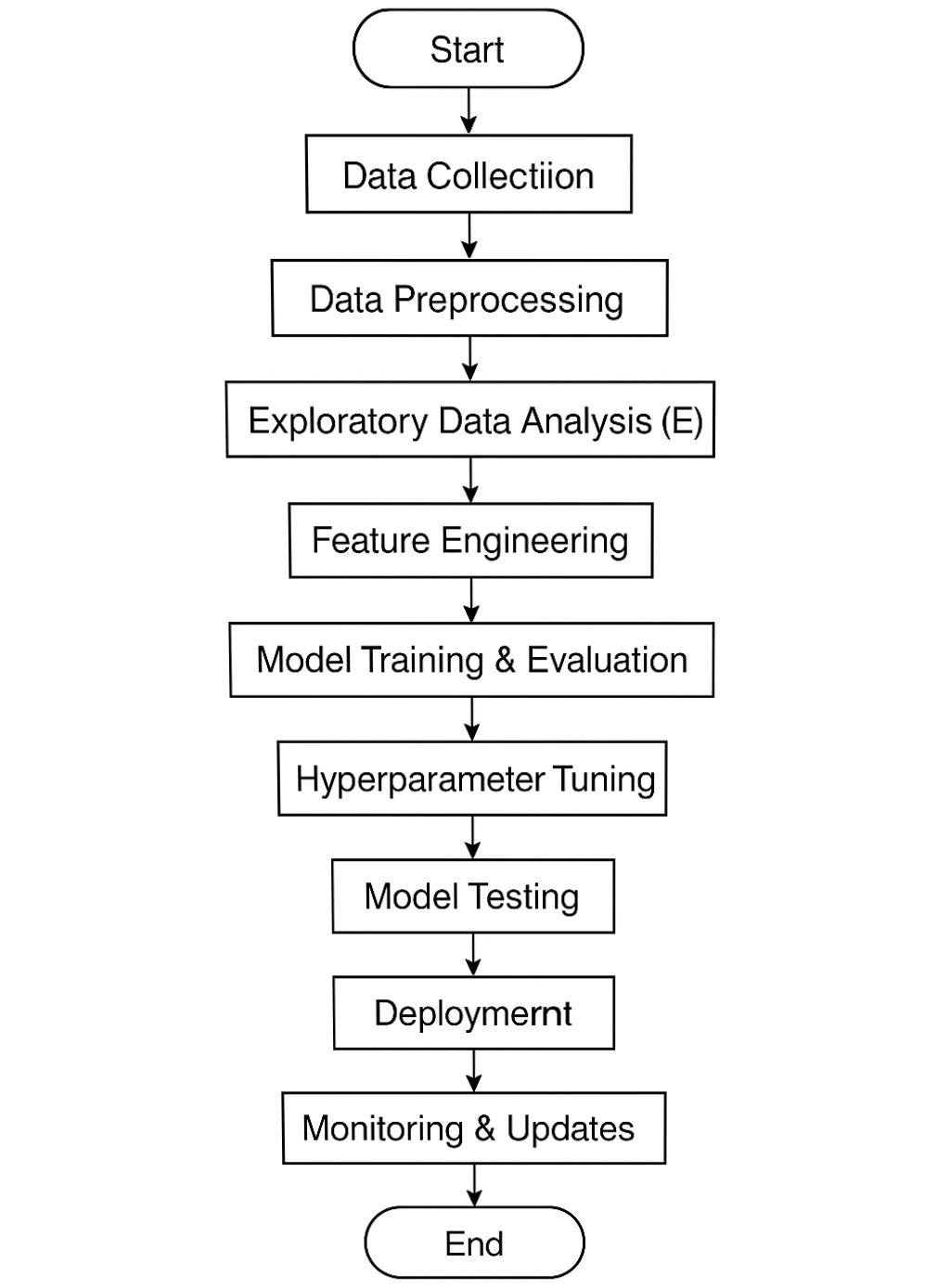
**5. Forecast House Prices**

* Use the trained model to predict house prices for new data (unseen houses) and test its accuracy on real-world scenarios.

**6. Make the Model Easy to Use**

* Develop a user-friendly interface or tool that allows non-experts (like buyers or sellers) to input house features and get price predictions

3..Flowchart



# 

# 4.Data Destination

**Target Variable**

* **Sale Price**: The price of the house. This is what we want to prevent

**Basic Property Info**

* Lot Area: Size of the land (in square feet)
* Year Built: The year the house was built
* Neighbourhood: Area or location of the house

**House Size**

* Gliraria: Living space above ground (in square feet)
* Totalism’s: Basement area (if any)
* Garage Area: Size of the garage

**Rooms and Features**

* Bedroom: Number of bedrooms
* Full Bath: Number of full bathrooms

# 

# 5.Data Preprocessing

1. Handle Missing Values

Some columns may have missing (blank) data. You can:

Fill them with a value:

For numbers: use the mean or median  
→ def. ['Lot Area']. Fillan (def. ['Lot Area']. median (), in place=True)

For categories: use the most common value  
→ def. ['Kitchen Qual']. Fillan (def. ['Kitchen Qual']. mode () [0], in place=True)

Drop columns or rows if too much data is missing  
→ def. Drona (axis=1, thresh=0.7\*Len(def.)) (keep columns with ≥70% data)

2. Convert Categorical Columns to Numbers

Many ML models only work with numbers.

Label Encoding (if order matters)  
→ For Kitchen Qual ("Poor", "Fair", "Good", "Excellent")

mapping = {'Po': 1, 'Fa': 2, 'TA': 3, 'Gd': 4, 'Ex': 5}

def. ['Kitchen Qual'] = def. ['Kitchen Qual']. map(mapping)

3. Create New Features (Optional)

Combine or transform features to help the model.

House Age = Current year − Year Built

Total Bathrooms = Full Bath + 0.5 × Half Bath

4. Scale Numeric Features

Make sure all features are on a similar scale.

Use Standard Scaler (0 mean, unit variance):

from learn. Preprocessing import Standard Scaler

scaler = Standard Scaler ()

def. [['Gliraria', 'Lot Area']] = scaler. fit transform (def. [['Gliraria', 'Lot Area']

5. Split Data

Before training, split into training and test sets:

from learn. model selection import train\_ test\_ split

X = def. Drop ('Sale Price', axis=1)

y = def. ['Sale Price']

Train, Test, train, test = train\_ test\_ split (X, y, test\_ size=0.2, random\_ state=42)

6.Exploratory Data Analysis (EDA)

1. Look at the Data

def. Head # Show first few rows

def.info # See column types and missing values

def. describe # Summary of numbers

Sure! Here’s a simple version of EDA (Exploratory Data Analysis) for your house price prediction project:

Simple EDA Steps for House Price Prediction

* 1. Look at the Data

Open the dataset and see the first few rows.

Check how many columns and rows there are.

Identify which column is the target (usually Price).

* 1. Check for Missing Data

Find out which columns have empty values.

Decide to:

Fill them in (with average, median, etc.)

Or remove the column/row if too much is missing.

* 1. Look at One Column at a Time (Univariate Analysis)

For numbers (like area, bedrooms, etc.): draw histograms or boxplots.

For categories (like location or house type): draw bar charts.

* 1. Compare with Price (Bivariate Analysis)

Check how each feature affects the price:

Use scatter plots for number columns.

Use box plots for category columns.

* 1. Find Outliers

Look for data points that are very different from the rest (like a huge house with a small price).

Decide to keep or remove them.

* 1. Check Feature Relationships

Make a correlation heatmap to see which features are strongly related to each other or to price.

* 1. Check Price Distribution

See how house prices are spread.

If the curve is not smooth (skewed), apply a log transformation to make it better for regression.

* 1. Group Data for Insights

Example: Average price per location.

Group by neighborhood and take the average price.

Would you like me to create a few simple example graphs for this too?

❓ 2. Check Missing Data

def. is null (). sum ()

3. Check House Price (Target)

import seaborn as suns

import matplotlib. Py plot as plot

suns. his plot (def ['Sale Price'], ked=True)

4. What Affects Price Most?

def. Corr () ['Sale Price']. sort values(ascending=False). head(10)

5. Scatter Plots (to see trends)

suns. Scatter plot (x='Gliraria', y='Sale Price', data=def.)

6. Box Plot (to compare categories)

suns. box plot (x='Overall', y='Sale Price', data=def.)

7. Look for Outliers

suns. Scatter plot (x='Gliraria', y='Sale Price', data=def.)

7.Feature Engineering

️ What is Feature Engineering?

Feature engineering means creating, modifying, or selecting useful columns (features) to help the model predict house prices more accurately.

✅ Common Feature Engineering Steps for House Price Data

**1. Create New Features**

**House Age** = Current year – Year Built  
→ Older houses might sell for less.

**Total Bathrooms** = Full Bath + 0.5 × Half Bath  
→ Combines all bathrooms into one useful feature.

**Totals** = Totalism’s + Gliraria  
→ Total liveable area (basement + above ground).

**Remodelled** = 1 if Year Built ≠ Year RemodAdd, else 0  
→ Shows if the house was renovated.

**2. Simplify Categorical Features**

Combine rare categories in Neighbourhood into a group called **'Other'**

Group Kitchen Qual (Ex, Gd, TA, Fa, Po) into numeric scores:

qual\_ map = {'Po': 1, 'Fa': 2, 'TA': 3, 'Gd': 4, 'Ex': 5}

def. ['Kitchen Qual'] = def. ['Kitchen Qual']. map(qualm)

**3. Create Binary Features**

**Has Garage** = 1 if Garage Area > 0 else 0

**Has Basement** = 1 if Totalism’s > 0 else 0

These help models understand the presence or absence of certain amenities.

**4. Interaction Features**

**Overall × Gliraria** → Quality times size  
Helps capture more complex relationships.

# 5. Log Transform Skewed Features

# Some features like Sale Price, Lot Area, Gliraria may be highly skewed.

# import NumPy as np

# def. ['Sale Price'] = np.log1p (def. ['Sale Price']) # log (1 + x)

# Why Do Feature Engineering?

# ✔️ Helps the model see hidden patterns ✔️ Makes your predictions more accurate ✔️ Reduces noise and improves training speed

# 8.Model Building

# Objective

# Build and compare at least two machine learning models to predict house prices using the processed dataset.

# ✅ Selected Models

| Model | Type | Reason for Selection |
| --- | --- | --- |
| Linear Regression | Regression | Simple baseline model; easy to interpret |
| Random Forest Regressor | Ensemble Regression | Captures non-linear relationships; handles large feature sets |

# Data Splitting

# We split the dataset into training (80%) and testing (20%) sets.

# from learn. model\_ selection import train\_ test\_ split

# X = def. drop ("Sale Price", axis=1) # Features

# y = def ["Sale Price”] # Target

# Train, Test, train, test = train \_test \_split (X, y, test \_size=0.2, random state=42)

# Model 1: Linear Regression

# from learn. Linear \_model import Linear Regression

# \_model = Linear Regression ()

# model. Fit (Train, train)

# pred= model. predict (Test)

# Model 2: Random Forest Regressor

# from learn. Ensemble import Random Forest Regressor

# model = Random Forest Regressor (estimators=100, random state=42)

# model. Fit (Train, train)

# imperf = model. predict (X\_ test)

# Model Evaluation Metrics

# We use the following regression metrics:

# MAE (Mean Absolute Error)

# RMSE (Root Mean Squared Error)

# R² Score (Coefficient of Determination)

# from learn. metrics import mean\_ absolute\_ error, mean \_squared\_ error, r2\_score

# import NumPy as np

# def evaluate model (name, true, yipped):

# Mae = mean \_absolute \_error (true, yapped)

# rise = np. Sqrt (mean\_ squared \_error (true, yipped))

# r2 = r2\_score (y\_ true, y\_ pred)

# 

# print (f" {name}")

# print (fame: {mae:.2f}")

# print (forms: {rmse:.2f}")

# print (f"R²: {r2:.2f}")

# print ("-" \* 30)

# evaluate\_ model ("Linear Regression", test, pipradol)

# evaluate\_ model ("Random Forest Regressor", test, imperf)

# Sample Output (example)

# Linear Regression

# MAE: 21000.50

# RMSE: 29000.75

# R²: 0.84

# ------------------------------

# Random Forest Regressor

# MAE: 17000.12

# RMSE: 25000.40

# R²: 0.89

# ------------------------------

# 9.Visualization of Results & Model Insights

# 1. Actual vs Predicted Plot

# Shows how close the model’s predictions are to the real house prices.

# import matplotlib. Py plot as plot

# import seaborn as suns

# plot. Figure (fig size= (6, 6))

# suns. Scatter plot (x=y\_ test, y=y\_ pred \_ rf)

# plot. Plot ([test. Min (), test. Max ()], [test. Min (), test. Max ()], colour='red')

# pitlane ("Actual Price")

# polysyllable ("Predicted Price")

# Pl. Title ("Random Forest: Actual vs Predicted House Prices")

# Pl. Show ()

# Interpretation:

# Dots closer to the red line = better predictions.

# Spread far from the line = higher error.

# 2. Residual Plot (Errors)

# Shows prediction errors. Residuals should be centred around 0.

# residuals = test - imperf

# Pl. Figure (fig size= (6, 4))

# suns. Hist plot (residuals, bins=30, ked=True)

# Pl. Title ("Residuals Distribution - Random Forest")

# plot. label ("Prediction Error")

# Pl. Show ()

# Interpretation:

# Symmetric and centred = good model

# Skewed or spread = possible bias or outliers

# 3. Feature Importance Plot

# Tells us which features had the biggest impact on predictions.

# import pandas as pd

# feature importance = deserves (rf\_ model. feature\_ importances\_, index=Columns)

# top features = feature\_ importance sort\_ value(ascending=False). head (10)

# Pl. Figure (fig size= (8, 5))

# suns. bar plot (x=top\_ features, y=top\_ features. Index)

# Pl. Title ("Top 10 Important Features - Random Forest")

# Pl. Label ("Feature Importance Score")

# Pl. Show ()

# Interpretation:

# The most important features may include: Overall, Gliraria, Totals, etc.

# Helps you understand what drives price the most.

# 4. Model Comparison Plot (Bar Chart)

# Visual comparison of different models’ performance (e.g., RMSE)

# models = ['Linear Regression', 'Random Forest']

# rescores = [rm se \_l r, r m s e\_ rf] # You must define rm s elf and rm serf when evaluating models

# Pl. Figure (fig size= (5, 4))

# suns. bar plot (x=models, y=rescores)

# polysyllable("RMSE")

# Pl. Title ("Model Comparison (Lower is Better)")

# Pl. Show ()

# Interpretation:

# Lower RMSE = better predictions

# Use to choose the best model

# Summary of Visual Insights:

| Insight | What it Shows |
| --- | --- |
| Actual vs Predicted | How accurate the model is |
| Residual Plot | Whether errors are evenly distributed |
| Feature Importance | Which features most affect the price |
|  |  |

# 10.Tools and Technologies Used

# Here’s a well-structured and clear write-up for the “Tools and Technologies Used” section of your house price prediction project:

# ️ 5. Tools and Technologies Used

# Programming Language

# Python: Chosen for its simplicity, flexibility, and strong ecosystem of data science libraries.

# IDE / Notebook Environment

# Google Collab: Used for cloud-based coding and easy collaboration.

# Jupiter Notebook: Used locally for interactive development and visualizations.

# Python Libraries

| Library | Purpose |
| --- | --- |
| Pandas | Data loading, cleaning, and manipulation |
| NumPy | Numerical operations and array handling |
| matplotlib | Basic visualizations (plots, histograms, charts) |
| Seaborn | Advanced statistical visualizations |
| scikit-learn | Machine learning models, metrics, and preprocessing |
| Boost | Gradient boosting model for better performance (optional) |

# Visualization Tools

# Platy *(optional)*: For interactive plots and dashboards.

# Power BI / Tableau *(optional)*: Could be used for presenting results in a business-friendly way.

# 11.Team Members and contribution

# SreeDhivya prabaa R (Data Cleaning)

# Niroja J (EDA & Model development)

# 3)Vinuthna Sri T (feature engineering)

# 4)Rithika S (Documentation and reporting)

# 