Github link: https://github.com/Arthiarthiangamuthu/project-forecasting-house-prices.git

Phase-2

Forecasting house prices accurately using smart regression techniques in data science

1. Problem Statement:

This problem can be tackled using **advanced regression techniques** in data science, which can model and predict house prices with higher accuracy. By leveraging machine learning algorithms such as **linear regression, decision trees, random forests, gradient boosting,** or even **neural networks**, we aim to provide a more reliable and data-driven approach to predicting house prices.

The goal is to develop a regression model that:

- **Accurately predicts house prices** based on available feature
- **Minimizes errors** such as overfitting or underfitting, ensuring that the model generalizes well to new data.
- **Incorporates advanced techniques** like feature engineering, hyperparameter tuning, and ensemble learning to optimize performance.
- Provides **insightful explanations** of how different features (such as location, amenities, or square footage) influence the pricing, offering transparency and interpretability.

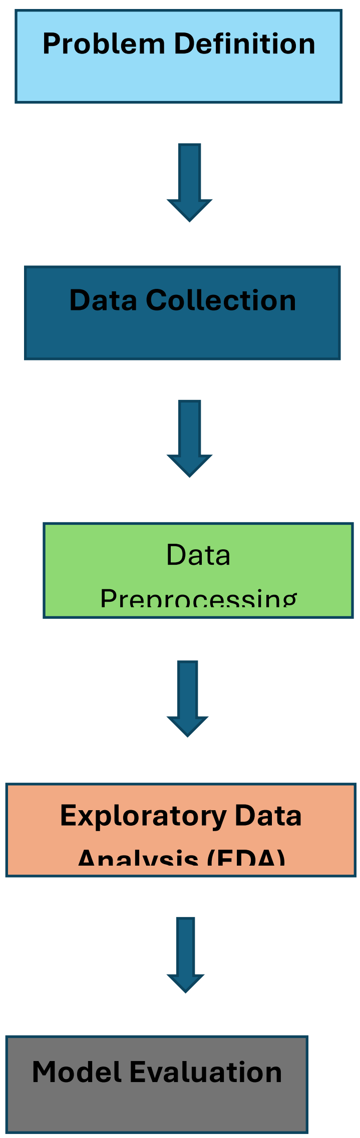
2. Project Objectives :

- **Data Cleaning & Preprocessing:**
    a. Ensure that the dataset is free of inconsistencies and missing values, and all features are ready for machine learning models.
- **Feature Selection & Engineering:**
    b. Identify which features significantly contribute to house price prediction and create new, meaningful features that can enhance model performance.
- **Hyperparameter Tuning:**
    c. Use methods like **Grid Search** or **Randomized Search** to find the best model configuration, including regularization strengths, learning rates, and tree depth (for tree-based models).
- **Model Evaluation & Error Analysis:**

d. Ensure that the model is robust, generalizing well to unseen data by evaluating it on test datasets and using error analysis to identify bias or areas for improvement.

- **Model Interpretability:**
  e. Provide stakeholders with clear explanations about how the model makes predictions, using feature importance, SHAP values, and other interpretability tools.
- **Scalability & Real-time Prediction:**
  f. Ensure the solution is scalable for real-world applications and can handle real-time house price predictions in dynamic environments.

3. Flowchart of the Project Workflow:



4. Data Description:

- **Dataset Name & Source**:
  House Prices: Advanced Regression Techniques — available on
  https://www.kaggle.com/datasets/yasserh/housing-prices-dataset
- **Data Type**:
  Structured tabular data comprising numerical and categorical variables.
- **Size & Features**:
  - **Training Set**: 1,460 records with 81 columns (including the target).
  - **Test Set**: 1,459 records with 80 features (excluding the target).
  - The dataset includes 79 explanatory variables detailing various aspects of residential homes in Ames, Iowa, such as lot size, year built, and overall quality. GitHub+2Medium+2Medium+2GitHub+5Medium+5GitHub+5
- **Static or Dynamic**:
  Static dataset — it does not update over time.
- **Target Variable**:
  SalePrice — representing the property's sale price in dollars. This is the variable the models aim to predict. DataHen+3Kaggle+3Medium+3

5. Data Preprocessing:

- **Data Cleaning**

a. Handling Missing Values

- **Numerical Features**: Impute missing values using the median or mean. For instance, if the 'LotFrontage' feature has missing values, you might fill them with the median value of that column.
- **Categorical Features**: Fill missing values with the mode (most frequent value) or a placeholder likeUnknown'. For example, if 'GarageType' is missing, you can replace it with 'Unknown'.

- **Feature Engineering**

a. Encoding Categorical Variables

- **Ordinal Encoding**: For features with an inherent order (e.g., 'ExterQual' with values like 'Poor', 'Fair', 'Good', 'Excellent'), map them to numerical values accordingly.

b. Creating New Features

- Combine existing features to create new ones. For instance, 'TotalBathrooms' can be derived by summing full and half bathrooms.

3. Feature Scaling

- **Standardization (Z-score Normalization)**: Rescale features to have a mean of 0 and a standard deviation of 1. This is especially useful for algorithms sensitive to feature scales.

- **Min-Max Scaling**: Rescale features to a specific range, typically [0, 1]. This is useful when the distribution is not Gaussian..

4.Data Splitting

- **Train-Test Split**: Divide the dataset into training and testing sets, typically using an 80-20 split. This helps in evaluating the model's performance on unseen data.
- **Cross-Validation**: Use k-fold cross-validation to ensure that the model's performance is consistent across different subsets of the data.

6. Exploratory Data Analysis (EDA):

1.Univariate Analysis

a. Numerical Features

Objective: Understand the distribution, central tendency, and dispersion of individual numerical variables.

- **Histograms**: Visualize the frequency distribution of variables like SalePrice, GrLivArea, and LotArea**Categorical Features**
- **Objective**: Assess the frequency distribution of categorical variables.

b.Categorical Features

- **Countplots**: Visualize the count of each category in variables like Neighborhood or HouseStyle.

2. Bivariate Analysis

a. Numerical vs. Numerical

Objective: Examine relationships between pairs of numerical variables.GeoDa+8The Click Reader+8Artificial Intelligence in Plain English+8

- **Scatterplots**: Identify correlations between variables like GrLivArea and SalePrice.

b. Categorical vs. Numerical

Objective: Understand how categorical variables influence numerical outcomes.

- **Boxplots**: Compare SalePrice across different categories of OverallQual

3. Multivariate Analysis

- **Objective**: Explore interactions among multiple variables simultaneously.
- **Correlation Matrix**: Identify pairs of variables with strong linear relationships.
- **Pairplots**: Visualize pairwise relationships and distributions among a set of variables.

7. Feature Engineering:

1. Creating New Features Based on Domain Knowledge and EDA Insights

a. Total Square Footage

Combine various area-related features to capture the total usable space:DEV Community.

```
df['TotalSF'] = df['TotalBsmtSF'] + df['1stFlrSF'] + df['2ndFlrSF']
```

2. Combining or Splitting Columns

a. Extracting Year and Month from Sale Date

If a 'SaleDate' column exists, decompose it:

```
df['SaleYear'] = df['SaleDate'].dt.year
df['SaleMonth'] = df['SaleDate'].dt.mont
```

b. Interaction Feature

Create features that capture interactions between variables:

```
df['OverallQual_GrLivArea'] = df['OverallQual'] * df['GrLivArea']
```

3. Applying Techniques like Binning, Polynomial Features, Ratios

a. Binning 'HouseAge'

Group houses into age categories:

```
df['AgeBin'] = pd.cut(df['HouseAge'], bins=[0, 10, 20, 50, 100, np.inf],
          labels=['0-10', '11-20', '21-50', '51-100', '100+'])
```

8. Model Building:

1. Model Selection

For predicting continuous variables like house prices, regression models are suitable. Two commonly used models are:

a. Linear Regression

- **Overview**: Assumes a linear relationship between independent variables and the target variable.
- **Pros**: Simple to implement and interpret.
- **Cons**: May underperform if the underlying relationship is non-linear or if multicollinearity exists.

b. Random Forest Regressor

- **Overview**: An ensemble learning method that builds multiple decision trees and merges their results.
- **Pros**: Handles non-linear relationships and interactions well; robust to outliers and overfitting.
- **Cons**: Less interpretable compared to linear models.

2. Data Splitting

Divide the dataset into training and testing sets to evaluate model performance on unseen data.

```
from sklearn.model_selection import train_test_split
```

3. Model Training

a. Linear Regression

```
from sklearn.linear_model import LinearRegression

lr_model = LinearRegression()
lr_model.fit(X_train, y_train)
```

b. Random Forest Regressor

```
from sklearn.ensemble import RandomForestRegressor

rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)
```

4. Model Evaluation

Evaluate model performance using regression metrics:CodeSignal+1DataHen+1

- **Mean Absolute Error (MAE)**: Average absolute difference between predicted and actual values.
- **Root Mean Squared Error (RMSE)**: Square root of the average squared differences between predicted and actual values.
- **R-squared (R^2)**: Proportion of variance in the target variable explained by the model.

```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import numpy as np
```

5. Interpretation of Results

- **MAE**: Provides a straightforward measure of average error in the same units as the target variable.
- **RMSE**: Penalizes larger errors more than MAE, useful when large errors are particularly undesirable.
- **R^2**: Indicates how well the model explains the variability of the target variable; values closer to 1 suggest better performance.

9. Visualization of Results & Model Insights:

1. Feature Importance Plot

Purpose: Identify which features most significantly influence house price predictions.

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

2. Residual Plot

Purpose: Assess the difference between actual and predicted values to identify patterns indicating model issues.

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

3. Actual vs. Predicted Plot

Purpose: Visualize how closely the model's predictions align with actual house prices.

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

4. Model Performance Metrics

Purpose: Quantify the model's predictive accuracy using standard regression metrics.

```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import numpy as np
```

10. Tools and Technologies Used:

1. Programming Language

- **Python**: Chosen for its extensive libraries and community support in data science and machine learning.

2. Development Environment

- **Jupyter Notebook**: Utilized for its interactive environment, facilitating exploratory data analysis and iterative model development.
- **Google Colab**: Employed for its cloud-based platform, allowing for collaboration and access to GPU resources.

3. Data Manipulation and Analysis Libraries

- **pandas**: Used for data manipulation and analysis, providing data structures like DataFrames.
- **NumPy**: Leveraged for numerical computations and handling arrays.

4. Data Visualization Tools

- **matplotlib**: Employed for creating static, animated, and interactive visualizations.
- **seaborn**: Built on top of matplotlib, used for making statistical graphics.
- **Plotly**: Utilized for interactive visualizations, aiding in dynamic data exploration.

5. Machine Learning Libraries

- **scikit-learn**: Used for implementing machine learning algorithms, including regression models, and for model evaluation.
- **XGBoost**: Applied for its efficient and scalable implementation of gradient boosting, enhancing model performance.

6. Optional Visualization Tools

- **Tableau**: Considered for creating dashboards and sharing insights with stakeholders.
- **Power BI**: Evaluated for its business analytics capabilities and interactive visualizations.

11. Team Members and Contributions:

A. Arthi

Data Cleaning

- **Responsibilities**:
  - Handled missing values through imputation or removal strategies.
  - Identified and removed duplicate records to ensure data integrity.
  - Standardized data formats and ensured consistency across the dataset.

Model Development & Evaluation

- **Responsibilities**:
- Built and compared multiple regression models, including Linear Regression and Random Forest.
- Split data into training and testing sets, ensuring appropriate stratification.

P. Babyshalini

Exploratory Data Analysis (EDA)

- **Responsibilities**:
  - Conducted univariate analysis using histograms, boxplots, and countplots.
  - Identified patterns, trends, and potential outliers in the data.

Feature Engineering

- **Responsibilities**:
- Created new features based on domain knowledge and insights from EDA.
- Applied techniques like binning and polynomial features to enhance model performance.

R. Akila

Documentation & Reporting

- **Responsibilities**:
- Compiled comprehensive documentation detailing each phase of the project.
- Created visual comparisons of model performance.