**Pune House Price Prediction Project Documentation**

**1. Project Overview**

The objective of this project is to build a machine learning model that can predict the price of houses in Pune based on various features such as location, size, number of rooms, amenities, and more. By accurately predicting house prices, this model can assist real estate agents, buyers, and investors in making informed decisions.

**2. Problem Statement**

House prices in Pune, like many urban cities, are influenced by a combination of factors such as the locality, area, amenities, age of the property, and market trends. Traditional methods of property valuation are often time-consuming and subjective. This project aims to automate the process by building a model that predicts house prices using historical data.

**Key questions:**

* What factors most influence house prices in Pune?
* How can we predict the price of a house based on those factors?

**3. Data Collection**

The success of the house price prediction model heavily depends on the quality and quantity of the data. For this project, relevant data was gathered from various publicly available real estate datasets, including platforms like:

* **Zillow, MagicBricks, or 99acres** (scraping real estate listings)
* **Open Data Platforms** (if available, especially city government data)
* **CSV datasets** from Kaggle or other repositories.

**Data Attributes:**

The dataset may contain the following features (columns):

1. **Location**: Area or locality of the property (e.g., Koregaon Park, Viman Nagar).
2. **Size**: Built-up area in square feet.
3. **Number of Bedrooms**: Number of bedrooms in the house.
4. **Number of Bathrooms**: Number of bathrooms.
5. **Floor Number**: Whether the house is on the ground floor, mid-floor, or top floor.
6. **Age of Property**: Years since the property was built.
7. **Amenities**: Whether the property has specific amenities like a swimming pool, gym, parking, etc.
8. **Proximity to key areas**: Distance from schools, hospitals, and transportation hubs.
9. **Price**: The dependent variable, i.e., the price of the house.

**4. Data Preprocessing**

The raw data typically needs to be cleaned and transformed before building any model. The following preprocessing steps were taken:

1. **Handling Missing Values**:
   * Imputed or removed missing values based on the data distribution and importance of the feature.
   * For categorical columns (like location), missing values were filled with a mode or dropped if too many were missing.
2. **Feature Encoding**:
   * **Categorical Encoding**: Locations or other categorical features (e.g., amenities) were encoded using techniques like one-hot encoding or label encoding.
3. **Feature Scaling**:
   * Standardization or Min-Max Scaling was applied to numerical features such as size, price, and age to ensure uniformity in the data range.
4. **Outlier Detection**:
   * Outliers in numerical features (such as price and size) were identified using statistical techniques like Z-score or IQR and treated accordingly (either by removing or transforming them).
5. **Data Split**:
   * The data was split into **training** and **testing** sets using an 80-20 split or cross-validation techniques.

**5. Exploratory Data Analysis (EDA)**

EDA is crucial for understanding the underlying patterns in the data.

1. **Distribution of House Prices**: Plotting histograms and box plots to understand the distribution of prices.
2. **Correlation Heatmap**: Visualizing the correlation between different features to understand how variables are interrelated.
3. **Feature vs Target Visualizations**: Scatter plots or bar charts comparing each feature (like size, location, number of rooms) with the target price to identify potential relationships.
4. **Location Analysis**: Visualizing house prices across different localities in Pune using geographic maps or scatter plots.

**6. Model Building**

Various machine learning models can be applied to this problem. The following models were tested for performance:

1. **Linear Regression**: As a baseline model, linear regression helps understand simple relationships between features and the target variable.
2. **Decision Trees**: A decision tree model was used for non-linear relationships between features.
3. **Random Forest**: An ensemble method that combines multiple decision trees to improve performance and reduce overfitting.
4. **Gradient Boosting Machines (GBM)**: Another ensemble method that builds trees sequentially to minimize errors.
5. **XGBoost**: A powerful, optimized version of gradient boosting that often gives superior results in regression tasks.
6. **Neural Networks**: For more complex relationships, deep learning models were also tested.

**Model Selection:**

After evaluating different models, the one with the best performance (based on **Mean Absolute Error (MAE)**, **Root Mean Squared Error (RMSE)**, or **R² score**) was chosen.

**7. Model Evaluation**

The performance of each model was evaluated using the following metrics:

1. **Mean Absolute Error (MAE)**: Measures the average magnitude of errors between predicted and actual values.
2. **Root Mean Squared Error (RMSE)**: Penalizes large errors more significantly and gives a good indication of model performance.
3. **R² Score**: Indicates how well the model explains the variance in the target variable.

Visualizations such as residual plots and error distribution graphs were used to further analyze the model’s performance.

**8. Model Tuning**

Hyperparameter tuning was performed to improve the model’s performance. Techniques such as **Grid Search** and **Random Search** were used to find the best set of hyperparameters for models like Random Forest, XGBoost, and Gradient Boosting.

**9. Deployment**

Once the model achieved satisfactory performance, the next step was to deploy it for real-time predictions.

**Steps for deployment:**

1. **API Development**: A REST API was developed using Flask or FastAPI to expose the model as a web service.
2. **Frontend**: A simple user interface was created (using HTML, CSS, and JavaScript) where users can input house features and get predictions.
3. **Integration with Databases**: For real-time predictions, the API was integrated with a database (like MySQL or MongoDB) to store historical data and manage inputs.

**10. Conclusion**

The house price prediction model provides a valuable tool for real estate professionals and prospective buyers. The model's accuracy can help make more informed decisions when buying or selling properties in Pune. Further improvements can be made by incorporating more advanced features such as market trends, inflation rates, or external economic factors.

**Future Work**:

* Integration of external datasets for improved accuracy (e.g., economic indicators, transportation data).
* Continuous model retraining with new data to ensure the model adapts to market changes.

**11. References**

* Kaggle Datasets: house Price Prediction
* "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow" by Aurélien Géron.