Home Value Prediction

A PROJECT REPORT

Submitted by

Aryan Mahajan(22MCA21063)

Sahil Rana(22MCA20783)

Sourav Singh(22MCA20249)

Aditi Dhiman(22MCA20821)

Priyanka(22MCA20765)

in partial fulfillment for the award of the degree of

Master Of Computer Application (M.C.A.*)*

IN

COMPUTER APPLICATIONS

CHANDIGARH UNIVERSITY

Discover. Learn. Empower.

Chandigarh University

January-April 2024

BONAFIDE CERTIFICATE

Certified that this project report "**HOUSE VALUE PREDICTION**" is the Bonafide work of "**ARYAN MAHAJAN**", "**SAHIL RANA**", "**SOURAV SINGH**", "**ADITI DHIMAN**", "**PRIYANKA**" who carried the work under my/our supervision.

Signature of the HoD

Signature of the Supervisor

Dr. Abdullah Khan

Associate Professor MCA

Prof. Vipin Kumar

Assistant Professor MCA

Submitted for the project viva-voce examination held on

INTERNAL EXAMINER

EXTERNAL EXAMINER

TABLE OF CONTENTS

Introduction to Home Value Prediction Model

In today's ever-evolving real estate market, accurately assessing the value of a home is crucial for both buyers and sellers. To address this need, advanced technologies such as Machine Learning, Data Science and a product-oriented approach have come together to create a powerful Home Value Prediction Model.

This model leverages the predictive capabilities of Random Forest and Linear regression i.e., machine learning algorithms known for its exceptional accuracy in regression tasks. Python, a versatile and widely-used programming language, serves as the foundation for implementing this model. It allows for seamless data manipulation, analysis, and model development, making it the ideal choice for creating a robust home value prediction system.

Furthermore, this model is not just a standalone tool but a product-oriented solution. This means that it can be accessed and utilized as a product, offering convenience and accessibility to both individuals and businesses in the real estate sector. Whether you are a home-owner looking to estimate your property's value or a real estate professional seeking valuable insights, this Home Value Prediction Model is designed to meet your needs effectively and efficiently.

Now, we will delve into the inner workings of this model, exploring the technologies, methodologies, and data sources that contribute to its accuracy. We will also provide practical examples and use cases to illustrate its real-world applications, empowering users to make informed decisions in the dynamic real estate market. Welcome to the future of home valuation, where Python and Data Science combine to redefine the way, we assess property values.

1. Client Identification and Recognition of Need

- Data-Driven Insights: The Home Value Prediction Model utilizes advanced machine learning techniques to provide accurate property valuations. This information empowers clients to understand the current market value of their property or the properties they are interested in.

- Comparative Analysis: The model can offer a comparative analysis of different properties, highlighting the strengths and weaknesses of each, helping clients narrow down their preferences.
- Market Trends: By analyzing historical data and market trends, the model can inform clients about potential future property value changes, helping them make informed decisions.

2. Identifying Suitable Properties:

- Budget Estimation: The model's predictions assist clients in estimating whether a property fits within their budget, ensuring they don't waste time considering properties that are beyond their financial means.
- Property Features: Clients can use the model to filter properties based on their desired features, such as the number of bedrooms, bathrooms, square footage, and more.
- Location Analysis: The model provides insights into the value of properties in specific neighborhood, helping clients make location-based decisions.

3. Investment Opportunities:

- Profitability Assessment: For clients interested in real estate investments, the model can estimate potential future property values, helping them identify properties with high potential for appreciation.
- Rental Income Projections: If clients are considering rental properties, the model can provide insights into expected rental income based on historical data and market conditions.

4. Timing and Negotiation:

- Market Timing: The model can indicate whether it's a buyer's or seller's market, allowing clients to time their transactions strategically.
- Negotiation Power: Armed with accurate property valuations, clients can negotiate confidently, knowing the fair market value of the property they are interested in.

5. Risk Mitigation:

- Identifying Overpriced Properties: The model can flag properties that are overpriced, helping clients avoid potentially bad investments.

6. Understanding Market Volatility:

- Clients can use the model to gauge the stability of the real estate market in their area of interest, reducing the risk of making ill-timed decisions.

1.1. Recognition & Knowledge of relevant contemporary issues

Staying informed about contemporary issues in the real estate market is essential for real estate professionals, investors, and anyone involved in property-related transactions. Here are some key areas of recognition and knowledge in the context of the real estate market:

1. Market Trends:

- Home Prices: Be aware of current trends in property prices, including fluctuations and regional variations.
- Supply and Demand: Understand the balance between housing supply and demand, as it greatly influences property values.
- Interest Rates: Stay informed about changes in mortgage interest rates, which affect affordability for buyers.

2. Technology and Innovation:

- Prop-tech: Recognize advancements in property technology (prop-tech) that can impact how real estate transactions are conducted, such as online listings, virtual tours, and blockchain based property records.
- Smart Homes: Be aware of the growing popularity of smart home features and their influence on property values.

3. Sustainability and Green Building:

- Environmental Regulations: Stay informed about regulations related to energy efficiency, emissions, and sustainable building practices, as these can affect property values.
- Green Certifications: Recognize the significance of green building certifications, like LEED, and their impact on property prices.

4. Government Policies:

- Taxation: Understand tax policies, including property taxes and capital gains taxes, which can impact real estate investments.
- Housing Policies: Keep abreast of government initiatives related to housing affordability, rent control, and zoning regulations.

5. Economic Indicators:

- GDP Growth: Monitor overall economic health, as it can influence real estate market dynamics.
- Unemployment Rates: Be aware of employment trends, as job opportunities often drive housing demand.

6. Demographic Shifts:

- Population Trends: Recognize changes in population demographics, as they can affect housing needs and preferences.
- Urbanization: Understand the ongoing trend of urbanization and its impact on property development in urban areas.

7. Health and Safety:

- Public Health: Stay informed about public health concerns and their potential impact on real estate, such as the effect of pandemics on remote work and urban exodus.
- Safety Regulations: Be aware of safety regulations and building codes that can influence property values.

8. Global Factors:

- International Investment: Recognize the impact of international investors on local real estate markets.
- Global Economic Trends: Understand how global economic trends, such as trade agreements and currency exchange rates, can affect property values.

9. Social and Cultural Shifts:

- Lifestyle Trends: Recognize changing lifestyle preferences, such as the demand for urban amenities or suburban living.
- Cultural Influences: Be aware of cultural factors that influence property choices, like the desire for diverse communities.

10. Regulatory Changes:

- Real Estate Laws: Stay updated on changes in real estate laws and regulations, including tenant rights and landlord responsibilities.
- Data Privacy: Understand data privacy regulations, especially in the context of property transactions.
- By staying informed about these contemporary issues, individuals and organizations can make well-informed decisions in the dynamic and ever-evolving real estate market.

1.2. Project Identification

In the context of the real estate market, project identification is the foundational step in property development and investment. It involves recognizing opportunities for real estate projects, assessing their feasibility, and aligning them with specific goals and market conditions. Here's a comprehensive guide to project identification in the real estate market:

1. Market Analysis:

- Market Research: Begin by conducting thorough market research to identify trends, demand, and potential niches in the real estate market.
- Location Assessment: Evaluate various locations and neighborhoods to determine their suitability for investment based on factors like proximity to amenities, transportation, and future growth prospects.

2. Project Objectives:

- Define Goals: Clearly define the objectives of the real estate project, whether it's for residential, commercial, industrial, or mixed-use purposes.
- Financial Targets: Set financial goals such as return on investment (ROI), cash flow expectations, and budget constraints.

3. Regulatory and Legal Considerations:

- Compliance Check: Ensure compliance with local zoning laws, building codes, and other regulatory requirements.
- Permitting and Approvals: Identify the necessary permits and approvals and plan for the permitting process.

4. Feasibility Assessment:

- Financial Feasibility: Assess the financial viability of the project by estimating costs, potential revenue, and profitability.
- Technical Feasibility: Evaluate the technical aspects of the project, including construction and infrastructure requirements.
- Market Feasibility: Analyze market demand and competition to determine the project's viability in the current market.

5. Risk Assessment:

- Identify Risks: Recognize potential risks related to the project, such as market fluctuations, construction delays, or unforeseen obstacles.
- Risk Mitigation: Develop strategies to mitigate identified risks and ensure the project's success.

6. Property Acquisition:

- Land Purchase: Acquire the necessary land or property for the project, considering location, size, and cost.
- Due Diligence: Perform due diligence to confirm property ownership, title history, and any legal encumbrances.

7. Financial Planning:

- Budgeting: Create a comprehensive budget that includes construction costs, financing, operating expenses, and contingencies.
- Financing: Secure financing through loans, investors, or other means to fund the project.

8. Design and Development:

- Architectural and Engineering Plans: Develop detailed architectural and engineering plans for the project.
- Construction: Begin construction activities, ensuring compliance with design specifications and safety standards.

9. Marketing and Sales Strategy:

- Marketing Plan: Create a marketing strategy to promote and sell the real estate units or properties.
- Sales Projections: Estimate sales timelines and pricing strategies to meet financial objectives.

10. Project Management:

- Project Team: Assemble a skilled project management team to oversee construction, budgets, and timelines.
- Monitoring and Control: Implement robust project monitoring and control systems to ensure the project stays on track.

11. Project Delivery:

- Completion: Ensure the project is completed to specifications and within the defined timeline.
- Occupancy or Sales: Transition to occupancy (for residential properties) or sales (for commercial properties).

12. Post-Project Evaluation:

- Performance Evaluation: Assess the project's performance against initial objectives and financial targets.
- Documentation: Maintain comprehensive records and documentation for future reference.

Project identification in the real estate market involves a holistic approach, encompassing market analysis, financial planning, risk management, and effective execution. Successful real estate projects are built on careful planning, thorough assessment, and precise execution to meet market demands and investor expectations.

1.3 Task Identification

For developing a House Sales Price Prediction Model using Machine Learning, task identification is crucial for defining and organizing the specific activities and responsibilities needed to successfully complete the project. Here's a comprehensive guide to task identification for this project:

1. Data Collection and Preparation:

- Data Gathering: Collect relevant real estate data, including property features (e.g., size, location, condition), historical sales prices, and market trends.
- Data Cleaning: Clean and preprocess the data by handling missing values, outliers, and data normalization.

2. Data Exploration and Analysis:

- Exploratory Data Analysis (EDA): Conduct EDA to gain insights into the dataset, identify correlations, and select relevant features for the model.
- Feature Engineering: Create new features or transform existing ones to improve predictive performance.

3. Model Selection and Development:

- Algorithm Selection: Choose suitable machine learning algorithms for regression, such as linear regression, decision trees, or ensemble methods like Random Forest or Gradient Boosting.
- Model Training: Train and finetune the selected model(s) using the prepared dataset.

4. Model Evaluation and Validation:

- Cross-Validation: Perform cross-validation to assess the model's performance, including metrics like Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R-squared (R**2).
- Hyperparameter Tuning: Optimize model hyperparameters to achieve the best predictive performance.

5. Model Interpretation and Visualization:

- Feature Importance: Interpret the model by analyzing feature importance scores to understand which factors most influence house sales prices.
- Visualization: Create visualizations (e.g., scatter plots, residual plots) to convey model predictions and insights effectively.

6. Model Deployment:

- Deployment Strategy: Plan how the model will be deployed, whether through a web application, API, or another method.
- Scalability: Ensure the model can handle real-time predictions and scale as needed.

7. Testing and Quality Assurance:

- Testing: Conduct rigorous testing to validate the model's accuracy and robustness.
- Quality Control: Implement quality control measures to ensure reliable predictions.

8. Documentation:

- Model Documentation: Prepare comprehensive documentation that explains the model's architecture, data sources, and usage instructions.
- Code Documentation: Document the code, including comments and explanations, for future reference.

9. User Interface Development (Optional):

- UI Design: If applicable, design a user-friendly interface for users to interact with the model.
- Integration: Integrate the model with the user interface and ensure seamless functionality.

10. User Training and Support (Optional):

- Training: Provide training sessions or documentation for users on how to use the model effectively.
- Support: Establish channels for user support and address inquiries or issues promptly.

11. Model Monitoring and Maintenance:

- Monitoring: Implement ongoing monitoring to detect model degradation or data drift.
- Maintenance: Regularly update the model, retrain it with new data, and refine its performance.

12. Project Evaluation and Feedback:

- Evaluation: Assess the project's success against initial objectives, including predictive accuracy and user satisfaction.
- Feedback: Gather feedback from users and stakeholders to identify areas for improvement.

Task identification for a House Sales Price Prediction Model project involves a series of well-defined steps, from data collection and model development to deployment and maintenance. Properly identifying and managing these tasks is essential for building an accurate and valuable predictive model for the real estate market.

CHAPTER 2.

LITERATURE REVIEW/BACKGROUND STUDY

2.1. Timeline of the reported problem

In this section of the literature, we provide a concise overview of the entire project's timeline, which spans various stages and activities.

Task	Start Date	Weeks To Complete
Project Planning and Preparation	Thursday, January 11, 2024	4
Data Preparation and Preprocessing	Thursday, February 1, 2024	5
Model Development	Tuesday, March 5, 2024	2
Evaluation and Validation	Sunday, April 21, 2024	2
Visualization and Reporting	Friday, May 10, 2024	2
Model Deployment	Thursday, May 16, 2024	1

Table 1 Timeline

Gantt Chart:

Figure 1 Gantt Chart

2.2. Existing solutions

- **Comparative Market Analysis (CMA)**:
  - **What it is**: CMAs are prepared by real estate agents to estimate a property's value based on recent sales of similar properties in the same neighbourhood or area.
  - **How it works**: Real estate agents use their local market knowledge and data on recent property sales to identify comparable properties. They make adjustments for differences in features and conditions to arrive at an estimated value.
- **Appraisal by Licensed Appraisers**:
  - **What it is**: Licensed appraisers conduct thorough property appraisals to determine its fair market value.
  - **How it works**: Appraisers inspect the property, consider its condition, location, and comparable sales data, and use industry-standard appraisal methods to arrive at a value.
- **Online Real Estate Marketplaces**:
  - **What they are**: Platforms like Zillow, Redfin, Realtor.com, and Trulia provide estimated property values based on AVMs and market data.
  - **How they work**: These platforms use their proprietary algorithms to generate property value estimates that can be a starting point for homeowners and buyers.
- **Home Inspection Reports**:
  - **What they are**: Home inspection reports may include assessments of a property's condition and any needed repairs, which can affect its value.
  - **How they work**: Home inspectors evaluate a property's structural integrity, systems, and overall condition, and their findings can influence a property's value.

- **Government Assessments**:
  - **What they are**: Local governments assess properties for tax purposes, and the assessed value can sometimes be used as an indicator of a property's worth.
  - **How they work**: Local tax assessors use a formula that considers factors like property size, location, and improvements to determine the assessed value.

2.3. Bibliometric analysis

- https://www.nar.realtor//research-and-statistics/housing-statistics/existing-home-sales
- https://financesonline.com/real-estate-statistics/
- https://www.statista.com/statistics/955598/india-real-estate-industry-market-size/
- https://www.zillow.com/research/
- https://www.99acres.com/articles/real-estate-market-price-trend-analysis-insite-report
- https://new.housing.com/research-reports
- https://docs.djangoproject.com
- https://www.mongodb.com/docs/

2.4. Problem Definition

Certainly, defining the problem statement for a housing sales price prediction project using machine learning (ML) is a crucial step to ensure clarity and alignment with project goals. Below is a breakdown of the problem statement, including what is to be done, how it is to be done, and what should not be done:

a. What Is to Be Done?

- *Problem Description:* The project aims to develop a machine learning model that predicts housing sales prices based on a set of relevant features or variables. The goal is to create a predictive tool that can assist real estate professionals, homeowners, and potential buyers in estimating the market value of residential properties accurately.

- *Data Collection:* Collect and assemble a comprehensive dataset that includes historical housing sales data. The dataset should encompass various attributes such as property size, location, features, market trends, and historical sale prices. Ensure that the data is clean, well-structured, and representative of the target housing market.

- *Data Preprocessing:* Preprocess the collected data by handling missing values, outliers, and data inconsistencies. Perform feature engineering to extract relevant features or transform existing ones that might enhance predictive accuracy.

- *Model Selection:* Choose appropriate machine learning algorithms for regression tasks. Experiment with different models such as linear regression, decision trees, random forests, gradient boosting, or neural networks to determine the most suitable model for the specific problem.

- *Model Training:* Split the dataset into training and validation sets. Train the selected machine learning model on the training data, ensuring hyperparameter tuning and optimization for optimal performance.

- *Evaluation Metrics*: Define evaluation metrics, such as Mean Absolute Error (MAE), Mean Squared Error (MSE), or Root Mean Squared Error (RMSE), to measure the accuracy and performance of the predictive model.

- *Model Deployment:* Deploy the trained model in a real-world environment where users can input property details, and the model will provide price predictions.

b. How It Is to Be Done?

- *Data Exploration:* Conduct exploratory data analysis (EDA) to gain insights into the dataset. Visualize data distributions, correlations, and identify potential patterns or outliers.

- *Feature Engineering:* Engineer relevant features by extracting information from existing variables or creating new ones that may enhance the model's predictive power.

- *Model Tuning:* Experiment with different hyperparameters, feature combinations, and model architectures to optimize the model's performance. Utilize techniques like cross-validation for robustness.

- *Validation:* Validate the model's performance on the validation dataset to ensure it generalizes well to unseen data. Use appropriate techniques like k-fold cross-validation to assess model stability.

- *Interpretability:* Attempt to interpret the model's predictions, especially for stakeholders who may need to understand how specific features influence price predictions.

c. What Not to Be Done?

- *Overfitting:* Avoid overfitting the model to the training data, as this can result in poor generalization to new data. Regularize the model and use appropriate validation techniques to detect overfitting.
- *Ignoring Data Quality:* Do not overlook data quality issues. Carefully handle missing values, outliers, and ensure data consistency.
- *Ignoring Ethical Considerations:* Avoid using discriminatory features or contributing to biases in housing pricing. Be aware of ethical considerations, fairness, and transparency in model development.
- *Ignoring Model Evaluation:* Do not rely solely on training performance. Continuously evaluate and update the model's performance as new data becomes available.
- *Neglecting Stakeholder Input:* Collaborate with real estate experts and stakeholders to ensure the model aligns with industry knowledge and user needs.

2.5. Goals/Objectives

1. User Authentication and Security:

Goal: Implement a secure and robust user authentication system.

Objectives:

- Utilize Django's built-in authentication for user login.
- Employ secure password hashing and salting techniques to protect user credentials.
- Implement measures to prevent common security vulnerabilities, such as SQL injection and cross-site scripting.
- Provide clear and informative feedback to users during login attempts.
- Include optional multifactor authentication for enhanced security.

2. User Data Input and Validation:

Goal: Enable users to input relevant data for home value prediction.

Objectives:

- Create an intuitive UI for users to input features like bedrooms, bathrooms, square footage, etc.
- Implement Django forms for input validation to ensure data integrity.
- Include error handling to guide users in case of input errors.
- Ensure adherence to privacy regulations while collecting user data.

3. Database and Data Handling:

Goal: Establish an efficient and secure database system.

Objectives:

- Design and implement models to store user data and predicted outcomes.
- Set up proper database constraints to maintain data consistency.
- Develop mechanisms for scalable data storage to accommodate increasing user and prediction data.

4. Machine Learning Model Integration:

Goal: Integrate a machine learning model for home value prediction.

Objectives:

- Train a regression model using a dataset of home values.
- Save and load the trained model within the Django backend.

- Ensure seamless communication between the input form, backend, and machine learning model.
- Implement error handling for model predictions.

5. User Dashboard:

Goal: Provide users with a dashboard to view past predictions and relevant information.

Objectives:

- Create a user-friendly dashboard interface.
- Display past predictions along with additional details.
- Include options for users to manage and view their historical data.

6. Admin Functionality:

Goal: Implement admin functionalities for system management.

Objectives:

- Create an admin login form with secure authentication.
- Implement role-based access control for varying admin privileges.
- Include features for admin data management and oversight.

7. Project Deployment:

Goal: Deploy the Django project for public or restricted access.

Objectives:

- Choose a deployment environment (e.g., local server, cloud platform).

- Configure deployment settings and environment variables.
- Ensure the project is accessible securely over HTTPS.

8. Documentation and Justification:

Goal: Document the project design decisions and justifications.

Objectives:

- Provide detailed documentation of the design flow and feature selection.
- Explain the rationale behind the chosen machine learning model and features.
- Ensure comprehensive documentation for ease of understanding and future maintenance.

CHAPTER 3.

DESIGN FLOW/PROCESS

3.1. Evaluation & Selection of Specifications/Features

1. Correlation with Target Variable (Price):

- Evaluate how each feature in your dataset correlates with the target variable, which is the home price.
- Correlation coefficients indicate the strength and direction of the relationship between features and the target variable. Features with higher correlation coefficients are more likely to be influential in predicting home prices.

2. Feature Importance from Random Forest:

- Random Forest models provide a feature importance score, which quantifies the contribution of each feature to the model's predictive performance.
- Features with higher importance scores are deemed more critical by the model in making predictions. Hence, these features should be given more weight in the selection process.

3. Domain Knowledge:

- Leverage expertise from professionals in the real estate domain to identify features that are known to impact home prices significantly.
- This could include factors such as location desirability, property size, number of bedrooms and bathrooms, proximity to amenities and schools, and neighborhood characteristics.

4. Feature Engineering:

- Explore creating new features from existing ones that may enhance the model's predictive power.

- For instance, combining 'Area' and 'No. of Bedrooms' to create a 'Price per Bedroom' feature could capture the relationship between property size and price more effectively.

5. Dimensionality Reduction Techniques:

- Implement techniques like Principal Component Analysis (PCA) or feature selection algorithms to identify the most informative subset of features.
- Dimensionality reduction helps mitigate the curse of dimensionality and reduces overfitting, leading to more robust models.

6. Multicollinearity:

- Detect and address multicollinearity issues among features, where certain features are highly correlated with each other.
- Multicollinearity can distort the model's coefficients and make interpretations unreliable. Strategies like removing redundant features or using techniques like variance inflation factor (VIF) can mitigate this issue.

7. Data Quality and Completeness:

- Ensure that selected features exhibit high data quality and completeness, as missing or inconsistent data can negatively impact model performance.
- Features with a substantial amount of missing values or inconsistencies should be carefully handled through imputation or exclusion.

8. Computational Complexity:

- Consider the computational resources required to train and deploy models with large feature sets.
- Balancing model complexity with computational efficiency is crucial, especially for real-time applications or resource-constrained environments.

9. Model Interpretability:

- Prioritize features that enhance the model's interpretability and explainability.
- Complex or overly engineered features may obscure the reasoning behind model predictions, making it challenging to convey insights to stakeholders.

10. Validation and Cross-Validation:

- Validate the performance of different feature subsets using techniques such as cross-validation.
- Assessing model performance metrics across various feature sets helps identify the most effective features and ensures model robustness and generalization.

3.2. Design Constraints

1. Feature Availability and Quality:

- ***Constraint:*** Ensure that the dataset contains essential features such as area, location, number of bedrooms, and amenities.
- ***Implementation:*** Verify that the dataset used for training and prediction includes all required features. Implement data validation and cleaning processes to handle missing values, outliers, and inconsistencies effectively.

2. Computational Resources:

- ***Constraint:*** Optimize the model for efficient memory usage, processing speed, and scalability to handle large datasets.
- ***Implementation:*** Monitor and manage computational resources during model training and deployment. Consider using scalable infrastructure such as cloud computing services to accommodate increasing data volumes and user demand.

3. Interpretability:

- ***Constraint:*** Ensure that the model is interpretable to stakeholders, including real estate professionals and end-users.
- ***Implementation:*** Select interpretable algorithms such as Random Forest and provide clear explanations of predictions. Avoid overly complex models that are difficult to interpret.

4. Performance Metrics:

- ***Constraint:*** Define performance metrics such as Mean Absolute Error (MAE) or R-squared to evaluate the model's accuracy.
- ***Implementation***: Calculate and report performance metrics on validation datasets to assess the model's effectiveness. Set thresholds for acceptable error rates based on domain expertise and stakeholder requirements.

5. Regulatory Compliance:

- ***Constraint:*** Ensure compliance with relevant regulations and ethical guidelines, such as data privacy laws and fair housing practices.
- ***Implementation:*** Implement privacy-preserving techniques to protect sensitive data. Avoid using features that may lead to discrimination or bias in predictions.

6. Model Robustness:

- ***Constraint:*** Ensure that the model is robust to variations in data distribution, input features, and external factors.
- ***Implementation:*** Test the model's performance under different scenarios, including variations in feature values and data quality. Implement techniques to handle outliers and noise effectively.

7. Scalability:

- ***Constraint:*** Ensure that the model can handle increasing data volumes and user demand.

- ***Implementation:*** Optimize algorithms and infrastructure for parallel processing and distributed computing. Consider deploying the model on scalable platforms such as cloud-based services.

8. User Experience:

- ***Constraint:*** Provide a positive user experience with an intuitive interface for inputting home features and viewing prediction results.
- ***Implementation:*** Design a user-friendly interface with clear instructions for inputting home features. Ensure responsiveness and accessibility across different devices and screen sizes.

9. Maintenance and Updates:

- ***Constraint:*** Plan for model maintenance and updates to address changes in data distributions, feature importance, and user requirements over time.
- ***Implementation:*** Establish procedures for monitoring model performance and retraining with new data. Implement version control to track changes and updates to the model code and data.

3.3. Analysis of Features and finalization subject to constraints

1. Feature Relevance and Quality:

- Assess each feature's importance in predicting home prices by analyzing correlation coefficients with the target variable (Price) and feature importance scores from the Random Forest model.
- Consider features that have a significant impact on home prices based on domain knowledge and data analysis.

2. Interpretability:

- Prefer features that are easily interpretable to stakeholders and end-users. Avoid overly complex or highly engineered features that may obscure the model's reasoning.

3. Data Quality and Compliance:

- Ensure that selected features meet data quality standards and comply with regulatory requirements such as data privacy laws and fair housing practices.
- Remove features with a high proportion of missing values, inconsistencies, or potential biases.

4. Scalability and Computational Resources:

- Consider the computational resources required to process and analyze features, especially if dealing with large datasets or complex algorithms.
- Select features that are computationally efficient and scalable for model training and deployment.

5. User Experience:

- Evaluate features based on their usability and impact on the user experience. Choose features that are intuitive for users to understand and interact with.

6. Performance Metrics:

- Assess features based on their contribution to model performance metrics such as Mean Absolute Error (MAE) or R-squared.
- Select features that improve model accuracy and meet performance thresholds set by stakeholders.

7. Model Robustness:

- Analyze features for their robustness to variations in data distribution, input features, and external factors.
- Choose features that enhance the model's stability and generalization capabilities across different scenarios.

8. Maintenance and Updates:

- Consider the long-term maintenance and update requirements for selected features. Choose features that are sustainable and adaptable to changes in data and user requirements.

3.4. Design Flow

Alternative Design/Process 1: Stratified K-Fold

Statified K-fold is known for its simplicity, scalability, and ability to handle large datasets with high dimensionality:

1. Data Collection and Preprocessing:

- Gather a dataset containing features such as area, location, number of bedrooms, amenities, and home prices.
- Preprocess the data to handle missing values, outliers, and ensure data quality.
- Encode categorical variables if necessary and scale numerical features.

2. Stratified K-Fold Splitting:

- Determine the number of folds (K) for cross-validation.
- Stratify the dataset based on the target variable to ensure that each fold maintains the same class distribution as the original dataset.
- Shuffle the data to randomize the order.
- Split the dataset into K folds, ensuring that each fold retains the same proportion of classes as the original dataset.

3. Model Training:

- Split the preprocessed data into training and testing sets.
- Train a Random Forest regression model using the selected features.
- Tune hyperparameters such as the number of trees, tree depth, and minimum samples per leaf using techniques like grid search or random search.

4. Model Evaluation:

- Evaluate the trained model using performance metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared on the testing set.
- Ensure that the model meets predefined performance thresholds and constraints.

5. Model Interpretation:

- Interpret the model's predictions and feature importance to stakeholders.
- Visualize feature importance using techniques like permutation importance or feature contribution plots.
- Provide explanations for individual predictions using techniques like SHAP (S-Hapley Additive ex-Planations).

6. Deployment:

- Deploy the trained Random Forest model to a production environment, such as a web server or cloud platform.
- Develop a user interface for users to input home features and receive predictions.
- Ensure scalability, reliability, and security of the deployed model.

7. Monitoring and Maintenance:

- Implement monitoring tools to track model performance, data drift, and user interactions.
- Establish procedures for model maintenance, including periodic retraining with new data and updates to address changing requirements.
- Monitor and optimize model performance over time to ensure continued effectiveness.

8. Documentation and Reporting:

- Document the entire process, including data preprocessing steps, feature selection rationale, model training details, and deployment procedures.

- Create reports or dashboards to communicate project findings, model performance, and insights to stakeholders and project collaborators.

Alternative Design/Process 2: Stratified K-Fold

1. Define the Problem and Data Acquisition

- Clearly define the problem you're trying to solve with your model (classification, regression, etc.).
- Gather relevant data for training and testing the model. Ensure the data is clean, consistent, and addresses the problem.
- Explore data visualization techniques to understand data distribution and identify potential issues.

2. Data Preprocessing

- Clean the data by handling missing values, outliers, and inconsistencies.
- Perform data normalization or standardization if necessary, especially for models sensitive to feature scales.
- Consider feature engineering to create new features that might improve model performance.

3. Model Selection and Base Learner

- Choose a base learner model that suits your problem type (e.g., decision tree, random forest). Bagging works well with various base learners.
- Define hyperparameters for the base learner model. These will be tuned later for optimal performance.

4. Bagging Implementation

- Split your data into training and testing sets.
- Implement the bagging algorithm:
    - Draw multiple bootstrap samples (samples with replacement) from the training data.
    - Train a separate instance of the base learner model on each bootstrap sample.
    - During prediction, make predictions on new data points using each trained base learner model and aggregate the predictions using

techniques like majority vote (classification) or averaging (regression) to get the final prediction.

5. Model Training and Evaluation

- Train the bagging ensemble model on the training data.
- Evaluate the model performance on the held-out testing data using appropriate metrics (e.g., accuracy, F1 score for classification, RMSE for regression).
- Consider using techniques like cross-validation to get a more robust estimate of model performance.

6. Hyperparameter Tuning (Optional)

- Use techniques like grid search or random search to optimize the hyperparameters of the base learner model within the bagging ensemble.
- This can significantly improve the overall performance of the model.

7. Model Selection and Deployment

- Based on evaluation metrics, choose the best performing model configuration.
- Consider deploying the model for real-world use cases. This might involve integrating it into an application or service.

8. Monitoring and Improvement

- Monitor the deployed model's performance over time.
- Re-evaluate and potentially retrain the model with new data or updated hyperparameters to maintain optimal performance.

3.5. Implementation plan/methodology

1. Data Collection and Preprocessing:

- Gather a dataset containing relevant features such as area, location, number of bedrooms, amenities, and home prices.
- Preprocess the data to handle missing values, outliers, and ensure data quality and integrity.
- Encode categorical variables and scale numerical features as necessary.

2. Feature Selection and Analysis:

- Analyze the relevance and importance of features using correlation analysis, domain knowledge, and feature importance scores from Random Forest.
- Select a subset of features that significantly impact home prices and meet interpretability and quality criteria.

3. Model Training and Validation:

- Split the preprocessed data into training and testing sets.
- Train a Random Forest regression model using the selected features.
- Tune hyperparameters such as the number of trees, tree depth, and minimum samples per leaf using techniques like grid search or random search.
- Validate the trained model using held-out test data and evaluation metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared.

4. Model Interpretation and Evaluation:

- Interpret the model's predictions and feature importance to stakeholders, providing insights into how different features influence home values.
- Visualize feature importance using techniques like permutation importance or SHAP values to enhance interpretability.
- Evaluate the model's performance and robustness across different scenarios and datasets, ensuring it meets predefined performance thresholds and constraints.

5. Deployment and User Interface:

- Deploy the trained Random Forest model to a production environment, such as a web server or cloud platform.
- Develop a user-friendly interface for users to input home features and receive predictions, adhering to usability and accessibility principles.
- Ensure scalability, reliability, and security of the deployed model and interface.

6. Monitoring and Maintenance:

- Implement monitoring mechanisms to track model performance, data drift, and user interactions in real-time.
- Establish procedures for model maintenance, including periodic retraining with new data and updates to address changing requirements.
- Monitor and optimize model performance over time to ensure continued effectiveness.

7. Documentation and Reporting:

- Document the entire project workflow, including data collection, preprocessing steps, feature selection rationale, model training details, and deployment procedures.
- Create reports or dashboards to communicate project findings, model performance, and insights to stakeholders and project collaborators for transparency and understanding.

CHAPTER 4
RESULTS ANALYSIS AND VALIDATION

4.1 Implementation of solution :

In a house value prediction model using Python and Random Forest, various analysis tools and libraries are commonly utilized for data preprocessing, model training, evaluation, and visualization. Here's an overview of the essential tools and libraries used in such a project:

1. Data Preprocessing:

Pandas: Pandas is a powerful library for data manipulation and preprocessing. It is used for tasks like loading data, handling missing values, and performing feature engineering.

2. Data Visualization:

Matplotlib and Seaborn: These libraries are employed for creating visualizations, including scatter plots, histograms, and heatmaps to explore the data's characteristics.

3. Machine Learning Libraries:

Scikit-Learn: Scikit-Learn offers a wide range of tools for machine learning tasks, including preprocessing, model selection, and evaluation. It is particularly useful for implementing the Random Forest (Ensemble Learning).

4. Model Training and Analysis:

Random Forest (Ensemble Learning): The Random Forest model itself is a crucial component of the project, implemented through Scikit-Learn.

5. Hyperparameter Tuning:

GridSearchCV or RandomizedSearchCV: These tools are part of Scikit-Learn and are used for hyperparameter tuning to optimize the Random Forest model's performance.

6. Model Evaluation:

Scikit-Learn Metrics: Scikit-Learn provides a range of metrics like Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared to evaluate the model's performance.

7. Feature Selection and Importance Analysis:

Scikit-Learn Feature Selection: Scikit-Learn offers tools for feature selection, including Select K Best and Select From Model, which can be used to assess feature importance within the Random Forest model.

8. Jupiter Notebooks:

Using Jupyter Notebooks allows you to create interactive and well-documented analysis reports. You can incorporate code, visualizations, and explanatory text to communicate your findings effectively.

Using Jupyter, we had implemented many things we don’t even know, such as:

In the figure mentioned below, we have found the homes with the most prominent “Number of Bedrooms”, we got to know that the houses with 1, 2, 3 and 4 Bedrooms have the most number of instances in the dataset.

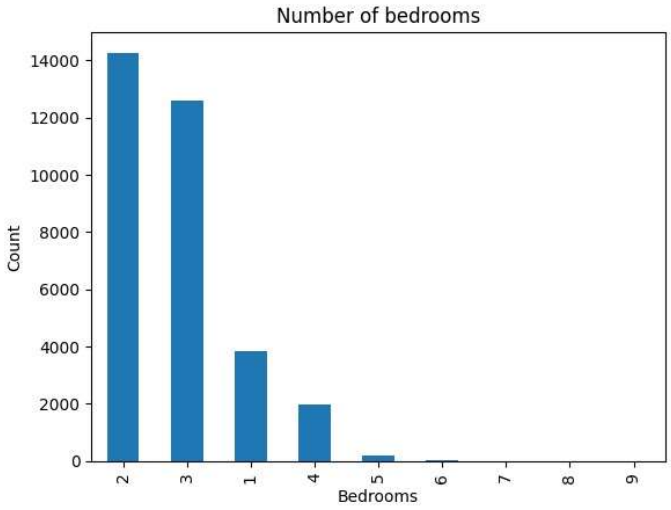


Figure 2: Distribution of Bedrooms count

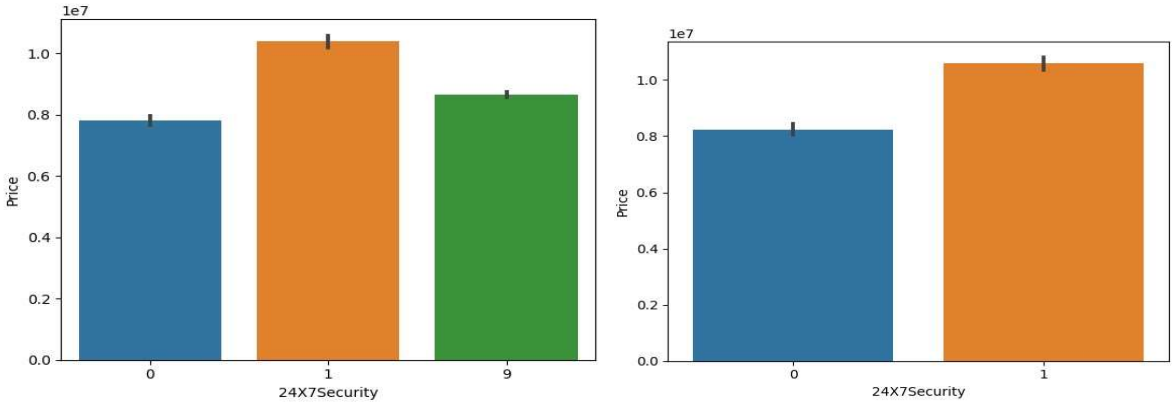


Figure 3: Converting Continuous to Discrete Data

The left figure above shows three unique values in the attribute "24X7 Security", but it is supposed to be a binary distribution as 24X7 Security must have values: Yes or No (1 or 0); the category 9 can't be explained in this regard. So we have excluded it from the column.

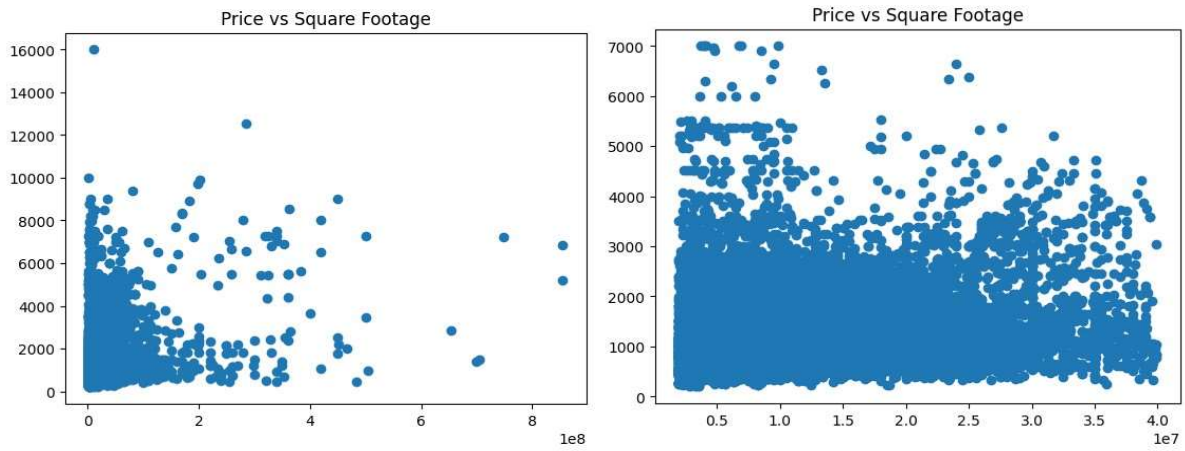


Figure 4: Removal of Outliers

Above we have the scatter plot for the attributes Price and Area (in sqft.), the distribution is spread out in a manner that has a few data points spread above 8000 sqft. and Price above 4*10^8 INR. We have performed snipping of values in both attributes in order to remove outliers which can skew our analysis.

The heatmap here shows the correlation between all the attributes in the dataset. We have to focus on the correlation coefficient of Price to all other attributes as to find the effect of each one on deciding the value of a particular property.

Figure 5: Observing Correlation of Data

The inference drawn from this diagram explains the high correlation between Price and Area, and between Price and No of Bedrooms. We can presume that these two attributes play a major role in deciding the value of a home.

4.2. Code For the Experiment:

ML Model (Random Forest):

```
# Importing Required Libraries
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

data=pd.read_csv('/content/Metropolitan.csv')


data.head()

data.describe()

# There is a house with 9 bedrooms!, would be an interesting find
# The area ranges from 200 sqft to 16,000 sqft. (The data has good distribution)

data['No. of Bedrooms'].value_counts().plot(kind='bar')
plt.title('Number of bedrooms')
plt.xlabel('Bedrooms')
plt.ylabel('Count')

# 2 and 3 bedroom homes (2BHK and 3BHK) are the most popular

data.columns
```

```
plt.scatter(data.Price, data.Area)
plt.title('Price vs Square Footage')

plt.scatter(data['Price'], data['City'])
plt.title('Price in Cities')

data.nunique()

data.drop(columns=['Resale',
'MaintenanceStaff',
'Gymnasium',
'SwimmingPool',
'LandscapedGardens',
'JoggingTrack',
'IndoorGames',
'Intercom',
'SportsFacility',
'ClubHouse',
'StaffQuarter',
'Cafeteria',
'MultipurposeRoom',
'WashingMachine','Gasconnection','AC','Wifi',
"Children'splayarea",
'LiftAvailable',
'BED','VaastuCompliant',
'Microwave', 'GolfCourse','TV',
'DiningTable','Sofa','Wardrobe','Refrigerator' ], inplace=True)
```

```
data.columns

data.head()

data.groupby('City')['City'].agg('count')

data.isnull().sum()

# No null values present

data=data[data['No. of Bedrooms'] < 5]

data['No. of Bedrooms'].unique()

# Removed homes that have bedrooms more than 4
# As they are insignificant in the dataset

data=data[data['Area'] <= 9000]

data=data[data['Price'] < 40000000]

data['Area'].min()

data['Area'].max()

data['Price'].min()
```

```
data['Price'].max()


plt.scatter(data.Price, data.Area)
plt.title('Price vs Square Footage')


sns.barplot(data=data, x='24X7Security', y='Price')


data.nunique()


data.drop_duplicates(keep='first', inplace=True, ignore_index=True)


data = data[data['ShoppingMall'] < 9]


sns.barplot(data=data, x='24X7Security', y='Price')


plt.scatter(data.Price, data.Area)
plt.title('Price vs Square Footage')


"""#### We had to cap the area distribution again to 5000 after 9000 because the individual rows with 9 as a value in a binary column were removed. Make it sound formally
#### It shows in the scatter plot wherein the max value on y-axis(namely square footage) is 5000
"""


data=data[data['Area'] <= 5000]
```

```
data.nunique()

data.describe()

data.head(100)

data.shape

from sklearn.preprocessing import OneHotEncoder, LabelEncoder

enc = LabelEncoder()
data['EncCity'] = enc.fit_transform(data['City'])

data.corr()

plt.figure(figsize=(10,6))
sns.heatmap(data.corr(),annot=True)

sns.scatterplot(x=data['Price'], y=data['Area'])
plt.xlabel('Price')
plt.ylabel('Area')
plt.title('India Metropolitan Areas: Price vs Area');

from scipy.stats import pearsonr
pearsonr(data['Price'], data['Area'])

target = "Price"
```

```
feature = ["Area"]
X_train = data[feature]
y_train = data[target]

y_mean = y_train.mean()
y_pred_baseline = [y_mean] * len(y_train)

len(y_pred_baseline) == len(y_train)

from sklearn.metrics import mean_absolute_error
mae_baseline = mean_absolute_error(y_train, y_pred_baseline)

print("Mean apt price:", round(y_mean, 2))
print("Baseline MAE:", round(mae_baseline, 2))

y_mean - mae_baseline

""" ### The MAE error is Rs 9330108.25 whilst the Baseline MAE is Rs.5048512.09. This means that by following this Baseline model, we would be off by about Rs.4281596.15587042"""

import plotly.express as px
fig=px.bar(data_frame=data, x='No. of Bedrooms', y='Price', color='City',barmode='relative')
fig.update_layout(title=dict(text='Average house price variation across Cities', xanchor='center', yanchor='top', x=0.5))

fig=px.scatter(data_frame=data, x='Area', y='Price', color='City', hover_name='City')
```

```
fig.update_layout(title=dict(text='House price Vs. Area across Cities',
xanchor='center', yanchor='top', x=0.5))

fig.show()


from sklearn import datasets, metrics

from sklearn.model_selection import train_test_split

from sklearn.ensemble import RandomForestRegressor


X = data[['Area','No. of Bedrooms', 'RainWaterHarvesting',
'School','ATM','ShoppingMall','24X7Security','PowerBackup','CarParking','Hos
pital','EncCity']]

y = data[['Price']]


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

y_train = np.ravel(y_train)


regressor = RandomForestRegressor(n_estimators=600, max_depth=8,
random_state=42)

regressor.fit(X_train, y_train)

y_pred = regressor.predict(X_test)


print(regressor.predict([[1000,1,1,1,1,1,1,1,1,1,1,1]]))


y_pred = regressor.predict(X_test)


import joblib

filename = "Forest1_model.joblib"

joblib.dump(regressor, filename)
```

Backend (Django):

```
{% load static tailwind_tags %}
<!doctype html>
<html>
   <head>
      <title>Predict</title>
      <link rel="stylesheet" href={% static 'theme\static\css\dist\stylePredict.css' %}>
      <script src="https://cdn.jsdelivr.net/npm/gsap@3.12.5/dist/gsap.min.js"></script>
      <script src="https://cdn.jsdelivr.net/npm/gsap@3.12.5/dist/ScrollTrigger.min.js"></script>
      <script src="https://cdn.tailwindcss.com"></script>
      {% tailwind_css%}
   </head>
   <body class="w-[100vw] h-[100vh] m-0 border-0 p-0 overflow-hidden">
      <div class="bg-cover z-[-1] w-[100vw] h-[100vh] absolute bg-black" style="background-image: url('{% static 'ben-duchac-96DW4Pow3qI-unsplash.jpg' %}'); opacity: 0.2;"></div>

      <form action={% url 'result' %} class="main w-[90vw] h-[98vh] flex flex-column text-white bg-black/[0.3] mb-0 rounded-[10px] hover:bg-black/[0.7]">
         {% csrf_token %}
         <div class="font-bold text-l Capitalize mt-0">Enter values</div>
            <div class="flex flex row w-[35vw] align-center justify-between items-center gap-6">
               <label for="" >City</label>
               <select name="City" id="city" for="city" class="text-black w-[20vw] text-center py-2 font-medium rounded">
```

```
                <option value="0" selected>Bangalore</option>
                <option value="1">Chennai</option>
                <option value="2">Delhi</option>
                <option value="3" >Hyderabad</option>
                <option value="4" >Kolkata</option>
                <option value="5" >Mumbai</option>
            </select>
        </div>
        <div class="flex w-[35vw] justify-between flex row align-center space-around items-center gap-10">
            <label class="w-[7vw]">Area<br /><span class="text-sm font-semibold text-white/[0.5]">(in sqft.)</span></label>
            <input type="text" class="w-[20vw] p-2 rounded font-xl text-black" placeholder="Enter the desired area(in sqft.)" name = "House_Area">
        </div>
        <div class="flex w-[35vw] justify-between flex row align-center space-around items-center gap-10">
            <label class="w-[7vw]">Bedrooms<br /><span class="text-sm font-semibold text-white/[0.5]">(1 to 4)</span></label>
            <input type = "text" class="w-[20vw] p-2 rounded font-xl text-black"  placeholder="Enter the number of bedrooms" name="No_of_Bedrooms">
        </div>
        <div>
            <div class="flex w-[35vw] justify-between flex row align-center space-around items-center gap-10">
                <label>Rain water</label>
                <div class="flex flex-row w-[21vw] justify-around cursor-pointer">
                    <div>
```

```
                    <input type = "radio" id="yes-1" name = "Rain_Water"
value="1">
                    <label for="yes-1">Yes</label>
                </div>
                <div>
                    <input type = "radio" id="no-1" name = "Rain_Water"
value="0">
                    <label for="no-1">No</label>
                </div>
            </div>
        </div>
        <div>
            <div class="flex w-[35vw] justify-between flex row align-center
space-around items-center gap-10">
                <label>School</label>
                <div class="flex flex-row w-[21vw] justify-around">
                    <div>
                        <input type = "radio" id="yes-2" name = "School"
value="1">
                        <label for="yes-2">Yes</label>
                    </div>
                    <div>
                        <input type = "radio" id="no-2" name = "School"
value="0">
                        <label for="no-2">No</label>
                    </div>
                </div>
            </div>
```

```
        </div>
        <div>
            <div class="flex w-[35vw] justify-between flex row align-center
space-around items-center gap-10">
                <label>ATM</label>
                <div class="flex flex-row w-[21vw] justify-around">
                    <div>
                        <input type = "radio" id="yes-3" name = "Atm" value="1">
                        <label for="yes-3">Yes</label>
                    </div>
                    <div>
                        <input type = "radio" id="no-3" name = "Atm" value="0">
                        <label for="no-3">No</label>
                    </div>
                </div>
            </div>
        </div>
        <div>
            <div class="flex w-[35vw] justify-between flex row align-center
space-around items-center gap-10">
                <label>Shopping_Mall</label>
                <div class="flex flex-row w-[21vw] justify-around">
                    <div>
                        <input type = "radio" id="yes-4" name = "Shopping_Mall"
value="1">
                        <label for="yes-4">Yes</label>
                    </div>
                    <div>
```

```
                    <input type = "radio" id="no-4" name = "Shopping_Mall"
value="0">
                    <label for="no-4">No</label>
                  </div>
                </div>
              </div>
              <div>
                <div class="flex w-[35vw] justify-between flex row align-center
space-around items-center gap-10">
                  <label>Security</label>
                  <div class="flex flex-row w-[21vw] justify-around">
                    <div>
                      <input type = "radio" id="yes-5" name = "Security"
value="1">
                      <label for="yes-5">Yes</label>
                    </div>
                    <div>
                      <input type = "radio" id="no-5" name = "Security"
value="0">
                      <label for="no-5">No</label>
                    </div>
                  </div>
                </div>
              </div>
              <div>
                <div class="flex w-[35vw] justify-between flex row align-center
space-around items-center gap-10">
                  <label>Power_Backup</label>
```

```
                <div class="flex flex-row w-[21vw] justify-around">
                    <div>
                        <input type = "radio" id="yes-6" name = "Power_Backup"
value="1">
                        <label for="yes-6">Yes</label>
                    </div>
                    <div>
                        <input type = "radio" id="no-6" name = "Power_Backup"
value="0">
                        <label for="no-6">No</label>
                    </div>
                </div>
            </div>
        </div>
        <div>
            <div class="flex w-[35vw] justify-between flex row align-center
space-around items-center gap-10">
                <label>Parking</label>
                <div class="flex flex-row w-[21vw] justify-around">
                    <div>
                        <input type = "radio" id="yes-7" name = "Parking"
value="1">
                        <label for="yes-7">Yes</label>
                    </div>
                    <div>
                        <input type = "radio" id="no-7" name = "Parking"
value="0">
                        <label for="no-7">No</label>
                    </div>
```

```
            </div>
          </div>
        </div>
        <div>
          <div class="flex w-[35vw] justify-between flex row align-center space-around items-center gap-10">
            <label>Hospital</label>
            <div class="flex flex-row w-[21vw] justify-around">
              <div>
                <input type = "radio" id="yes-9" name = "Hospital" value="1">
                <label for="yes-9">Yes</label>
              </div>
              <div>
                <input type = "radio" id="no-9" name = "Hospital" value="0">
                <label for="no-9">No</label>
              </div>
            </div>
          </div>
        </div>
        <input id = 'submitBtn' type="submit" class=" submit-btn w-[35vw] bg-white text-black py-1 cursor-pointer font-medium hover:bg-zinc-800 hover:text-white" value = "Predict">
        <div class = "row">
          <h2 id = "answer" >The result is {{classification_result}}</h2>
        </div>
      </div>
      </form>
```

```
</body>
</html>
```

Setting.py:

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',

    'cover',
    'history',

    'tailwind',
    'theme',

    'django_browser_reload',
]

TAILWIND_APP_NAME = 'theme'

INTERNAL_IPS = [
    "127.0.0.1",
]
NPM_BIN_PATH = "C:/Program Files/nodejs/npm.cmd"
```

```
MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
    'django_browser_reload.middleware.BrowserReloadMiddleware',
]
```

Model.py:

```
from django.db import models

class database(models.Model):
    House_Area = models.IntegerField()
    No_of_Bedrooms = models.IntegerField()
    Rain_Water = models.IntegerField()
    School = models.IntegerField()
    Atm = models.IntegerField()
    Shopping_Mall = models.IntegerField()
    Security = models.IntegerField()
    Power_Backup = models.IntegerField()
    Parking = models.IntegerField()
    Hospital = models.IntegerField()
```

```
    City = models.CharField(max_length=10)
    classification = models.IntegerField()
```

View.py:

```
from django.shortcuts import redirect, render
from cover.models import database
import joblib

def coverPage(request):
    return render(request,"base.html")

def predict2(request):
    return render(request,"predict2.html")

def result(request):
    model = joblib.load('F1_model.joblib')

    House_Area = request.GET['House_Area']
    No_of_Bedrooms = request.GET['No_of_Bedrooms']
    Rain_Water = request.GET['Rain_Water']
    School = request.GET['School']
    Atm = request.GET['Atm']
    Shopping_Mall = request.GET['Shopping_Mall']
    Security = request.GET['Security']
    Power_Backup = request.GET['Power_Backup']
    Parking = request.GET['Parking']
    Hospital = request.GET['Hospital']
```

```
City = request.GET['City']

lis = []

lis.append(House_Area)
lis.append(No_of_Bedrooms)
lis.append(Rain_Water)
lis.append(School)
lis.append(Atm)
lis.append(Shopping_Mall)
lis.append(Security)
lis.append(Power_Backup)
lis.append(Parking)
lis.append(Hospital)
lis.append(City)

classification = model.predict([lis])

database.objects.create(
    House_Area=House_Area,
    No_of_Bedrooms=No_of_Bedrooms,
    Rain_Water=Rain_Water,
    School=School,
    Atm=Atm,
    Shopping_Mall=Shopping_Mall,
    Security=Security,
    Power_Backup=Power_Backup,
```

```
        Parking=Parking,
        Hospital=Hospital,
        City=City,
        classification=classification[0]
    )
    return render(request, 'predict2.html', {'classification_result':
classification[0]})
```

Urls.py:

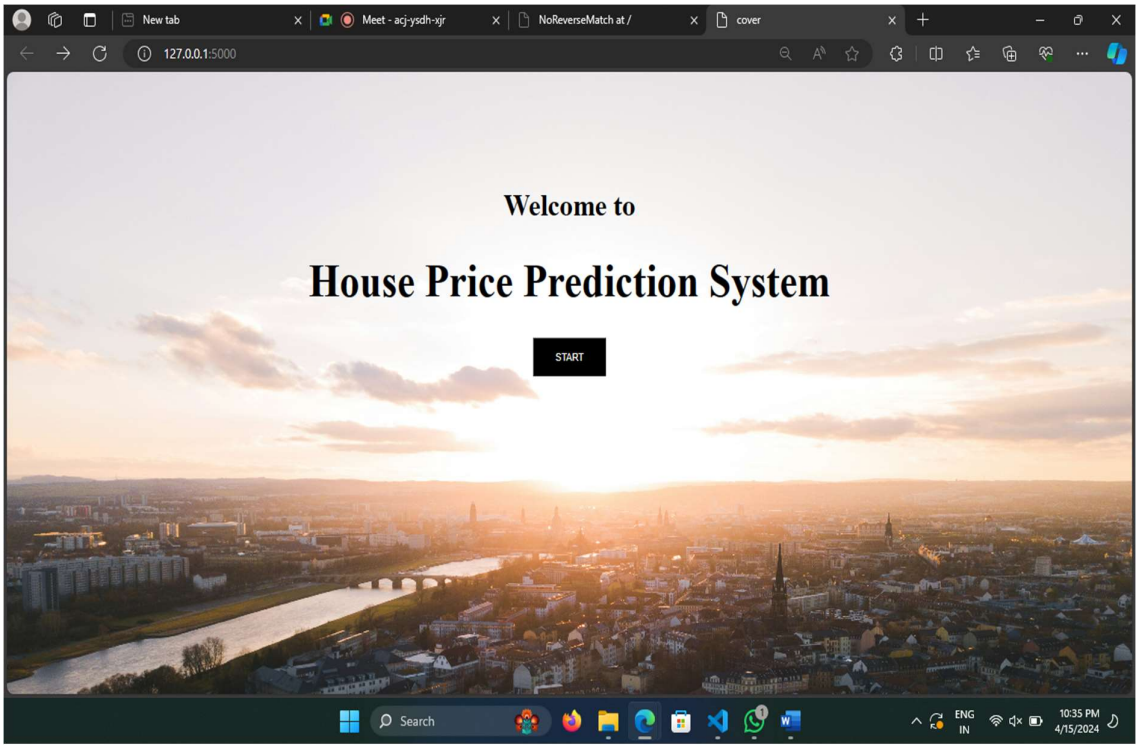
```
from django.contrib import admin
from django.urls import path, include
from cover import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('',views.coverPage),
    path('predict/', views.predict2, name="predict"),
    path('result/', views.result, name="result"),

    path("__reload__/",
include("django_browser_reload.urls")),
 ]
```

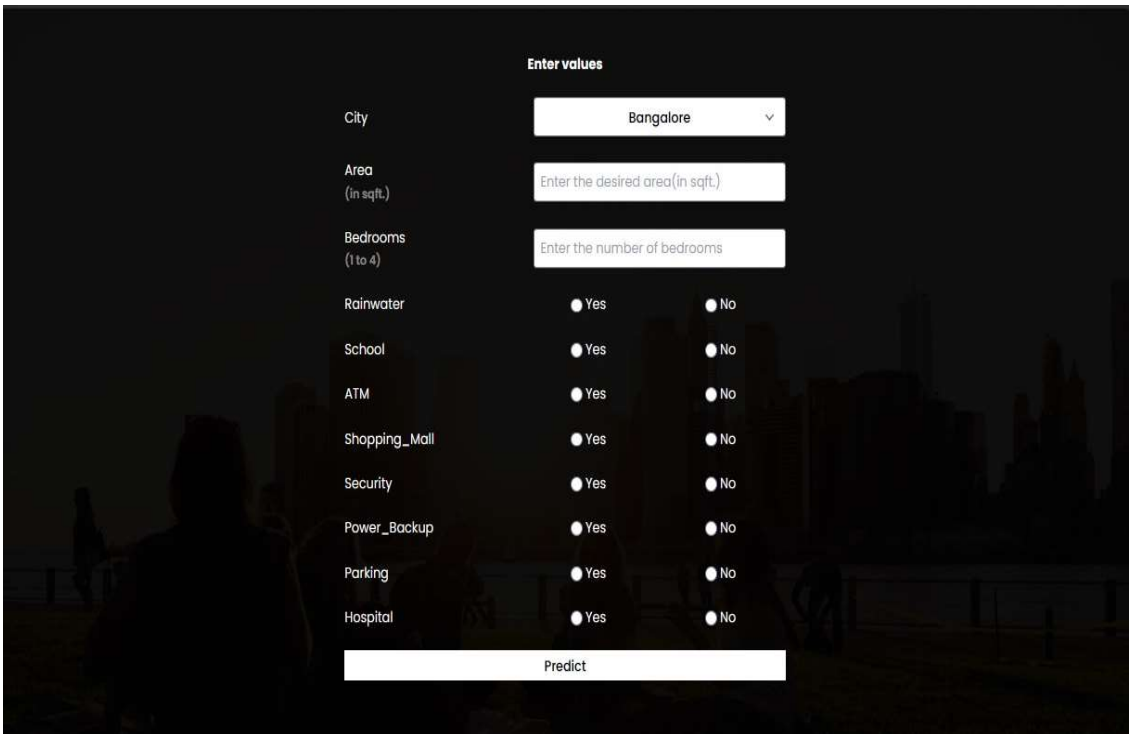
Admin.py:

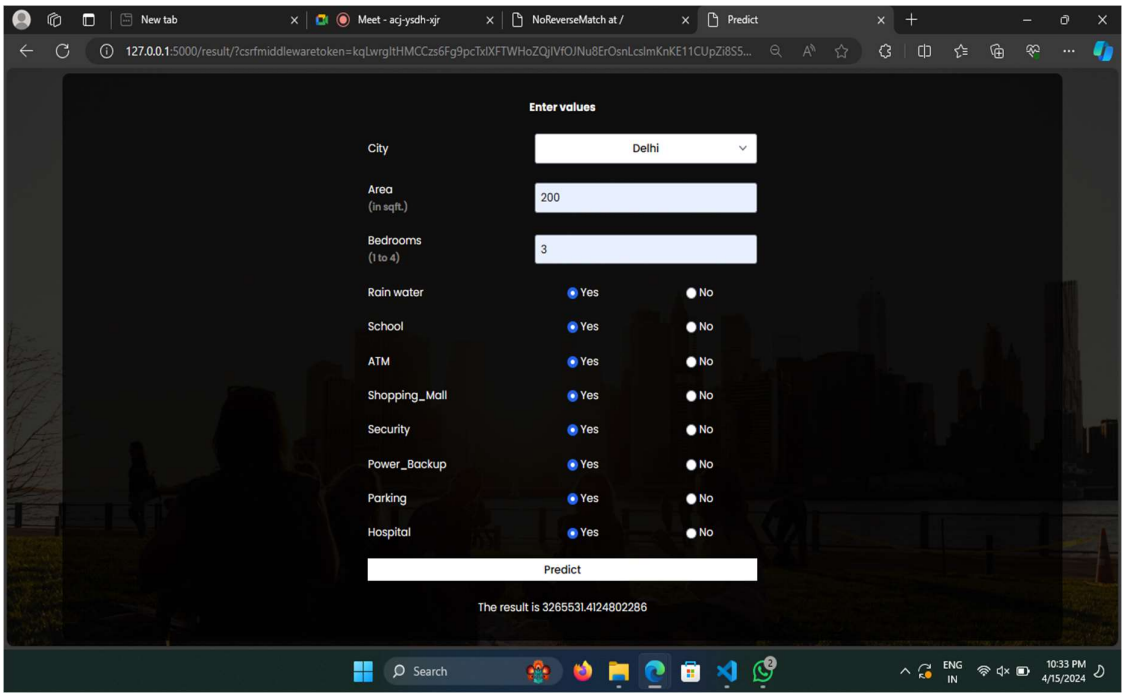
```
from django.contrib import admin
from cover.models import database
admin.site.register(database)
```

OUTPUT:

Frontend (Prediction):






New tab
Meet - acj-ysdh-xjr
NoReverseMatch at /
Predict
127.0.0.1:5000/result/?csrfmiddlewaretoken=kqLwrgItHMCCzs6Fg9pcTxIXFTWHoZQjIVfOJNu8ErOsnLcslmKnKE11CUpZi8S5...
Enter values
City
Delhi
Area
(in sqft.)
200
Bedrooms
(1 to 4)
3
Rain water
Yes
No
School
Yes
No
ATM
Yes
No
Shopping_Mall
Yes
No
Security
Yes
No
Power_Backup
Yes
No
Parking
Yes
No
Hospital
Yes
No
Predict
The result is 3265531.4124802286
Search
ENG
IN
10:33 PM
4/15/2024

5.1. Conclusion

The house value prediction model we discussed has a potential to be really useful. Here's what we can conclude in simpler terms:

1. Considering Everything:

The model looks at a bunch of different things about a house, not just its size or location. It thinks about everything from where it is to how nice the neighborhood is.

2. Making it Easy for People:

The goal is to make it easy for people to use. The model can be adjusted to fit what each person cares about, like schools nearby or how close it is to work.

3. Keeping Up with Changes:

The model doesn't stay the same forever. It can change and adapt to what's happening in the housing market. So, it stays useful and accurate over time.

4. Being Fair and Responsible:

The people working on this model care about being fair. They want to make sure it doesn't favor one group of people over another. They're also paying attention to any new rules or standards to make sure everything is done responsibly.

5. Working with Others:

They're open to working with other groups, like real estate companies and the government.
This way, they can get more information and make the model even better.

5.2. Future work

The future scope of home value prediction lies in making the process even more accurate and useful for homeowners, buyers, and real estate professionals. In simpler terms, here are some potential advancements and areas of growth:

1. Precision in Predictions:

We can expect improvements in the accuracy of home value predictions. Advanced algorithms and access to more comprehensive datasets may lead to more precise estimates of a property's worth.

2. Incorporating More Factors:

Future models may consider an even broader range of factors influencing home values. This could include emerging trends in urban development, environmental factors, and new amenities in the neighborhood.

3. Realtime Market Insights:

The use of real-time data and continuous updates could provide more dynamic insights into the everchanging real estate market. This would be especially beneficial for both sellers and buyers in making timely decisions.

4. Integration with Emerging Technologies:

Integration with technologies like augmented reality (AR) could offer immersive experiences for potential buyers. They might be able to virtually tour homes and assess their value more comprehensively.

5. Enhanced User Interfaces:

The interfaces used to interact with home value predictions could become more user-friendly and accessible (deployment using flask, Django, etc.), allowing a broader audience to utilize and understand the information provided.

6. Integration with Financial Planning:

Home value predictions might become more closely integrated with financial planning tools, helping individuals make informed decisions about their property investments as part of their overall financial strategy.

7. Greater Transparency:

Future models may prioritize transparency, explaining in simpler terms how predictions are calculated. This can enhance trust and understanding among users.

8. Market Trend Analysis:

Predictive models could offer more sophisticated trend analysis, helping both buyers and sellers anticipate and respond to market shifts.

9. Reinforcement Learning:

We could offer user feedback form after the prediction has been made. The minute changes to the values can be kept into consideration for further model improvement (accuracy). After a certain amount of instances that will be collected, would later be used for reinforcement learning.