DIC Project Phase 3

Model Deployment and Building Data Product

Mahammad Thufail - mahamma2
Sai Shirini Prathigadapa - saishiri
Rekha Anvitha Inturi - rekhaanv

Title: Airbnb Analysis Tool - price prediction and visualisation.

Problem Statement:

The project aims to predict the prices of Airbnb listings based on relevant features of such properties. These features include the neighbourhood group, type of room, number of minimum nights, the number of reviews, host listings count, availability throughout the year, geographical coordinates (latitude and longitude), and other relevant details that influence pricing dynamics in the hospitality industry.

a. Code Documentation:

Based on the implementation and evaluation of several predictive models, the best-performing model was selected for its high R-squared value and low Mean Squared Error (MSE). The chosen model for this project is the Random Forest, which was observed to provide the most accurate predictions for Airbnb listing prices.

The model is seamlessly integrated into a Flask-based web application, providing an intuitive user interface that allows users to input features of Airbnb listings and receive price predictions. The web application includes features such as form inputs for various listing characteristics and dynamic display of the prediction results. Below are the technologies used:

Flask:

The technology chosen for model deployment in this project is Flask, a lightweight web server gateway interface (WSGI) web application framework. Flask is renowned for its minimal dependency on external libraries and its straightforwardness, thanks to its Python-based development environment. In this project, Flask serves as the web interface for predicting Airbnb listing prices. It integrates the Random Forest machine learning model, providing a user-friendly platform where users can input property features and receive immediate price predictions. This setup not only facilitates easy user interaction but also ensures quick and efficient handling of predictive requests, making it an ideal choice for deploying interactive data-driven web applications.

HTML, CSS, Bootstrap:

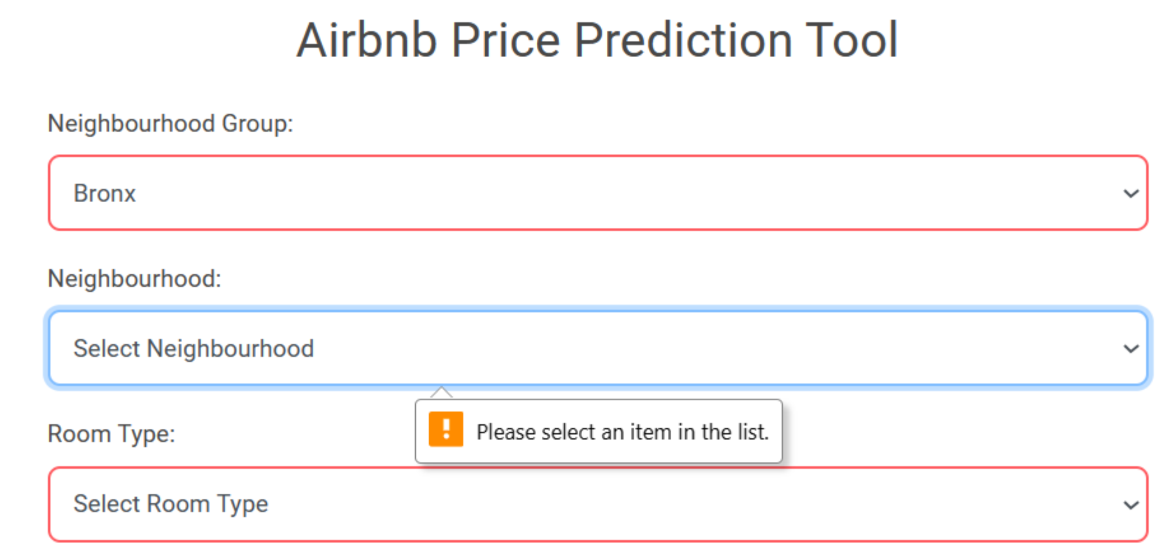
The user interface of the Airbnb Price Prediction Tool requires users to provide various details about Airbnb listings to generate price predictions. To facilitate this, a form-based interface is designed using HTML, which effectively organizes input fields for features like neighbourhood, room type, minimum nights, and more. The frontend is enhanced with CSS and Bootstrap, improving the overall user experience. Bootstrap, in particular, is employed for its responsive design capabilities, ensuring that the application is accessible and functional across different devices and browsers. This combination of technologies not only simplifies the development process but also enhances the aesthetic and functional qualities of the application, making it more engaging and easier to use for end-users.

Front-end Code:
The developed front-end for the Airbnb Price Prediction Tool is a web-based application that includes forms where users can input relevant attributes of Airbnb listings to predict prices. The user interface consists of select fields for attributes like neighbourhood group, room type, and the borough where the listing is located. Additionally, input fields are provided for entering the neighbourhood, minimum nights, number of reviews, host listings count, availability days, and geographical coordinates (latitude and longitude).

Upon submission, the form data is sent to the Flask backend, where the Random Forest model processes the inputs and predicts the price. The predicted price is then displayed in a modal on the same page, leveraging HTML template literals for dynamic content insertion (e.g., {{ prediction_text }}). This setup not only provides immediate feedback to the user but also enables a seamless integration of backend computations with frontend display, enhancing user interaction and understanding of how different features impact the predicted prices.

Visualization Tool:
In addition to the price prediction form, the front-end includes a visualization tool that allows users to select different features for graphical analysis. Users can choose two features to compare and select the type of graph they wish to view (e.g., scatter plot, bar chart, histogram, line graph). This component uses Chart.js (or another JavaScript charting library, if specified) to render the selected data dynamically. The visualizations help users to discern patterns and correlations between different property features and their effects on pricing. This interactive tool is crucial for deeper data exploration and helps in making informed decisions about property pricing strategies.

Error Handling:
In the Airbnb Price Prediction Tool, robust form error handling is implemented to ensure all necessary inputs are provided by the user before submission. If a user attempts to submit the form without selecting or entering a value in any of the required fields, the application will display an error message stating, “Please fill out this field/ Please select item in the list.” This is crucial as the Random Forest model requires complete data to make accurate predictions. The error checking is performed by validating the values of all input fields in the form. If any field is found empty, an error message is prominently displayed at the top of the form. Once the user fills in all the required fields, the error message is automatically cleared, and the form can be successfully submitted. This feature not only prevents incomplete data submission but also guides users to provide the necessary information, ensuring a smooth and efficient user experience.



Back-end Code Explanation:

Flask Application Setup:
The Flask application acts as the back-end server for our Airbnb price prediction web tool. It's configured to read HTML templates from a specified directory, facilitating the dynamic rendering of web pages based on user input.

Routing in Flask:
Default Home Route (/):
Method: GET
Function: This route serves the main page of the web application, where users can enter the details of an Airbnb listing into a form. The form collects inputs like neighbourhood group, room type, number of nights, reviews, and other pertinent details required for price prediction.

Prediction Route (/predict):
Methods: GET (to load the form) and POST (to submit the form and receive predictions).
Form Processing and Prediction:
Upon receiving a POST request, the server extracts and preprocesses the data from the form to align with the input requirements of the Random Forest model. This preprocessing might include validation checks, normalization, or encoding of categorical variables.
The application includes error handling to ensure that all fields are filled out before submission. If any field is missing, an error message is displayed, prompting the user to complete all required inputs.
The model then uses these inputs to predict the price of the Airbnb listing.

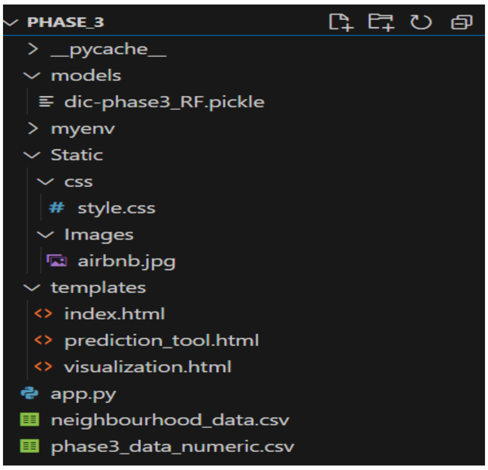
When it comes to processing data for our model, we ensure that all the variables, whether they represent neighborhoods, neighborhood groups, or room types, are encoded numerically. This ensures that the model can understand and analyze the information effectively. For categorical variables, like neighborhood names or room types, we typically employ techniques such as one-hot encoding or label encoding. This means that when a user provides information in the form of strings, like the name of a neighborhood or the type of room they're interested in, we convert these strings into numerical values before feeding them into our model for prediction. This numeric representation enables the model to make accurate predictions based on the input provided by the user. Once the model processes the data and generates predictions, we can then convert these numeric results back into a format that's meaningful and interpretable for the user. This ensures a seamless interaction between the user and the predictive capabilities of our model.

Dynamic Predictions and Visualization:

The back-end might also handle requests for dynamic predictions where certain features (e.g., room type, number of reviews) are varied to show how different configurations could influence the predicted price. This could be particularly useful for users looking to understand how different features impact pricing.
A code, perhaps named request.form.values(), used to generate sets of feature values that alter some while keeping others constant. These scenarios are then passed to the model, which computes prices for each scenario, providing a richer, data-driven visualization back to the user.

Working Instructions for the Code:

Initial Setup:
Ensure that all HTML files are placed in the designated templates folder within our Flask application directory. This setup is necessary because Flask automatically looks for HTML files in this folder when rendering templates.



Load the Model:
The Random Forest model, presumably saved in a pickle file, needs to be loaded into the Flask application. Ensure the pickle file is correctly placed in a known directory and loaded at the start of our Flask app to be available for making predictions upon request.

```python
# Load trained model
model = pickle.load(open('models\\dic-phase3_RF.pickle', 'rb'))
```

Server Configuration and Execution:

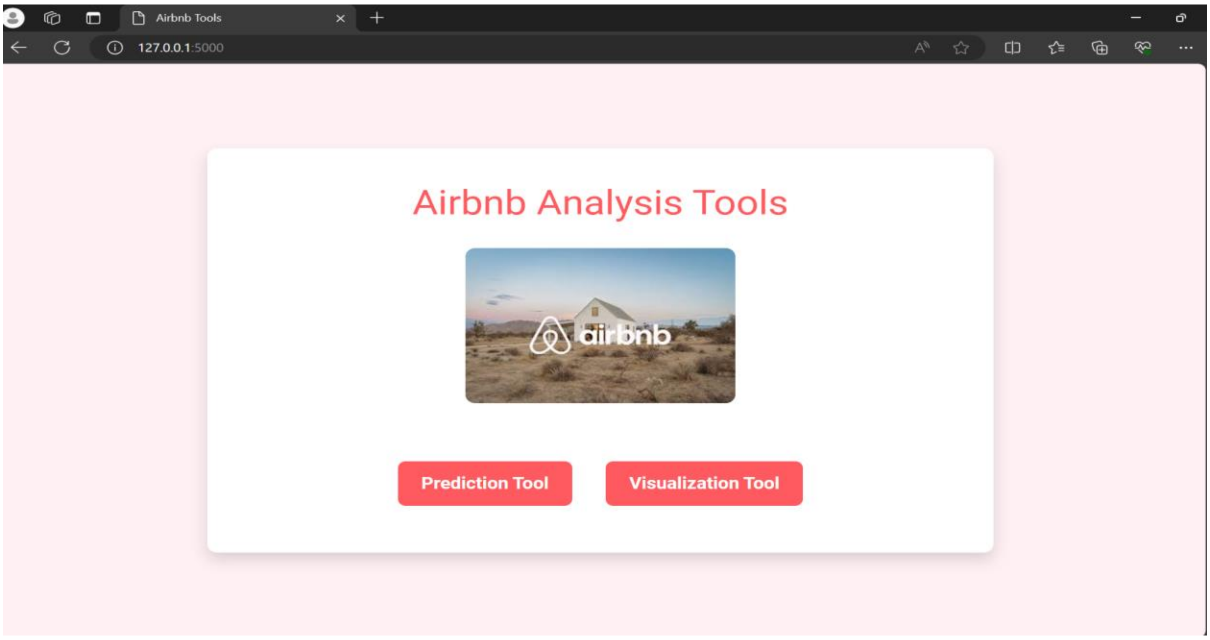
Run Flask application using a command like flask run, which starts the server on the default port (usually 5000). This command activates the Flask server, making application accessible from a web browser.

```
(myenv) PS C:\Users\saish\OneDrive\Desktop\Phase_3> flask run
>>
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
```

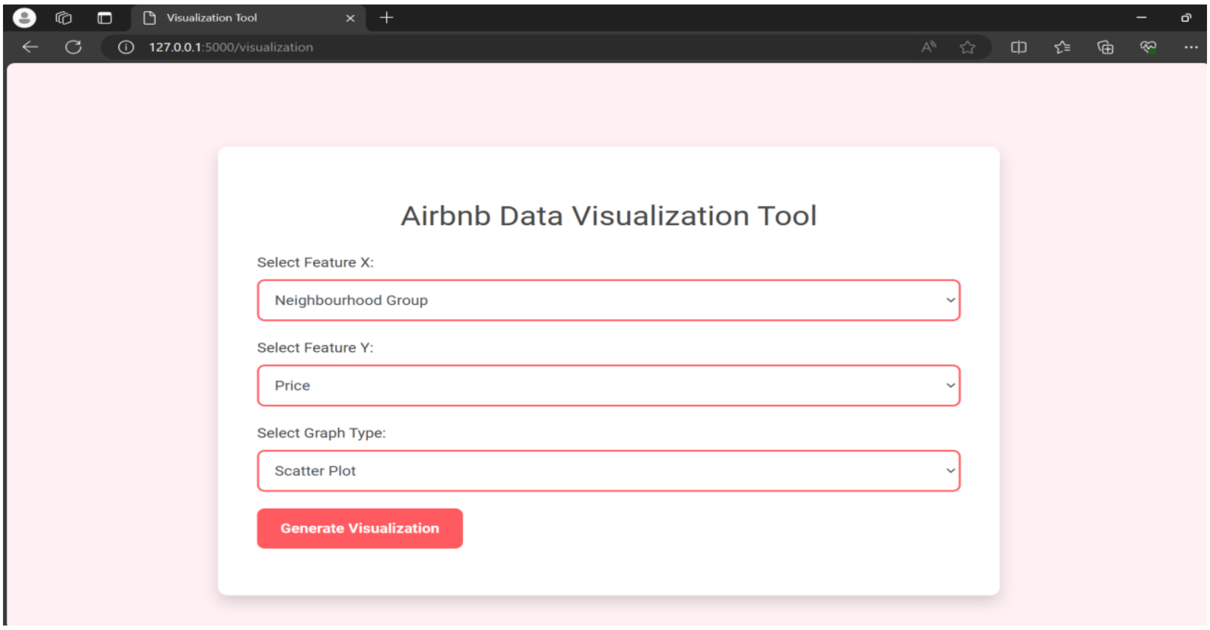
Using the Web Application:

Access the application via the public URL provided by the proxy setup or directly through the local host address if running on our own machine.

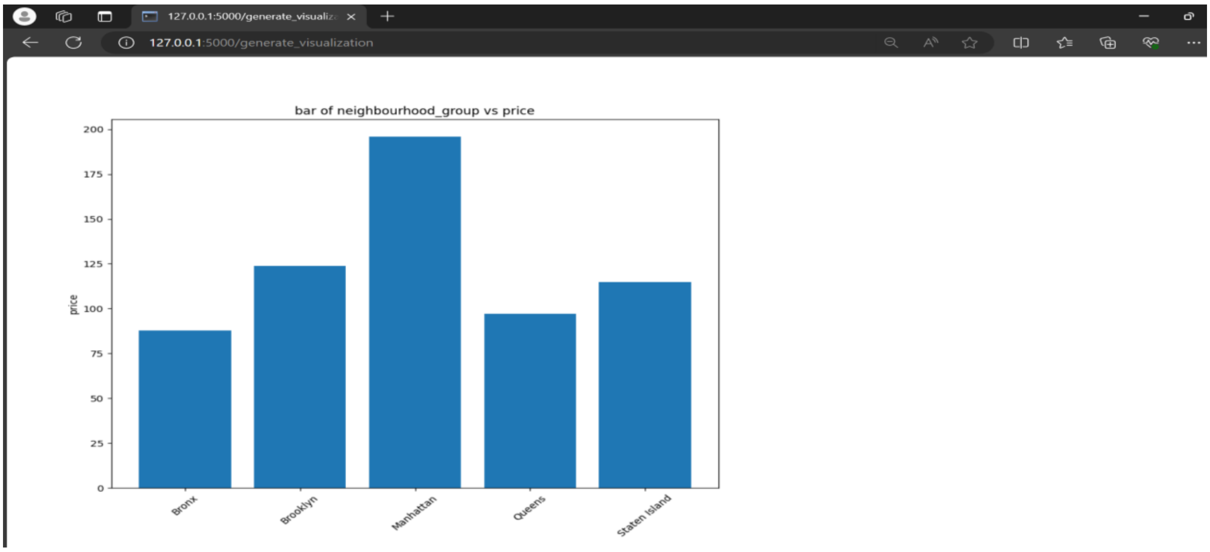
Navigate to the main page (home route /) to find the prediction and visualization tools.



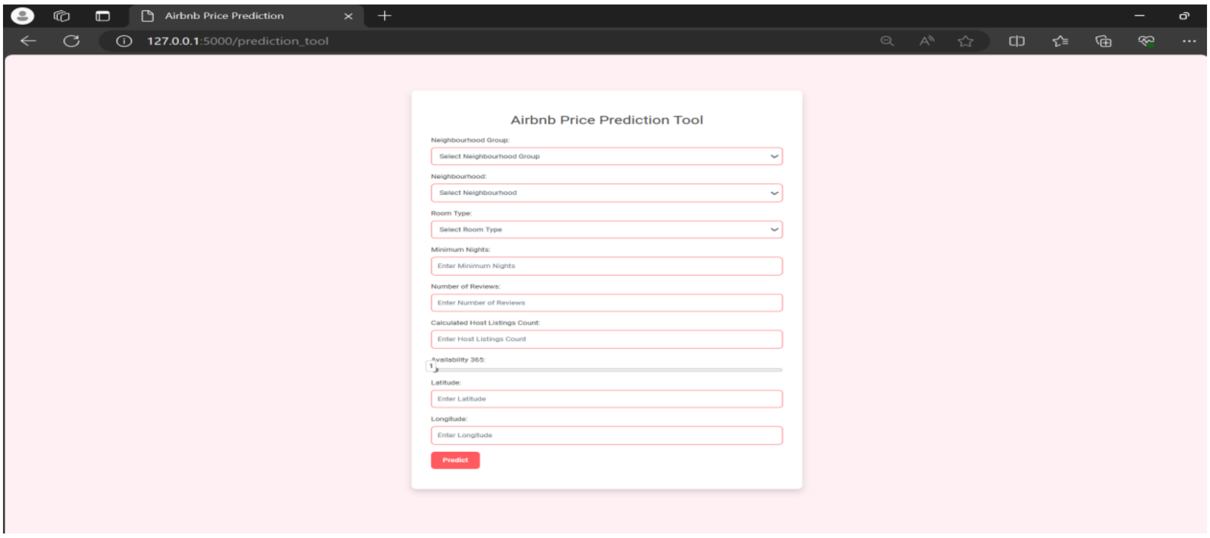
By clicking visualisation tools in home page it will redirect to visulization page with form to take feature as input and select the type of plot want to display .



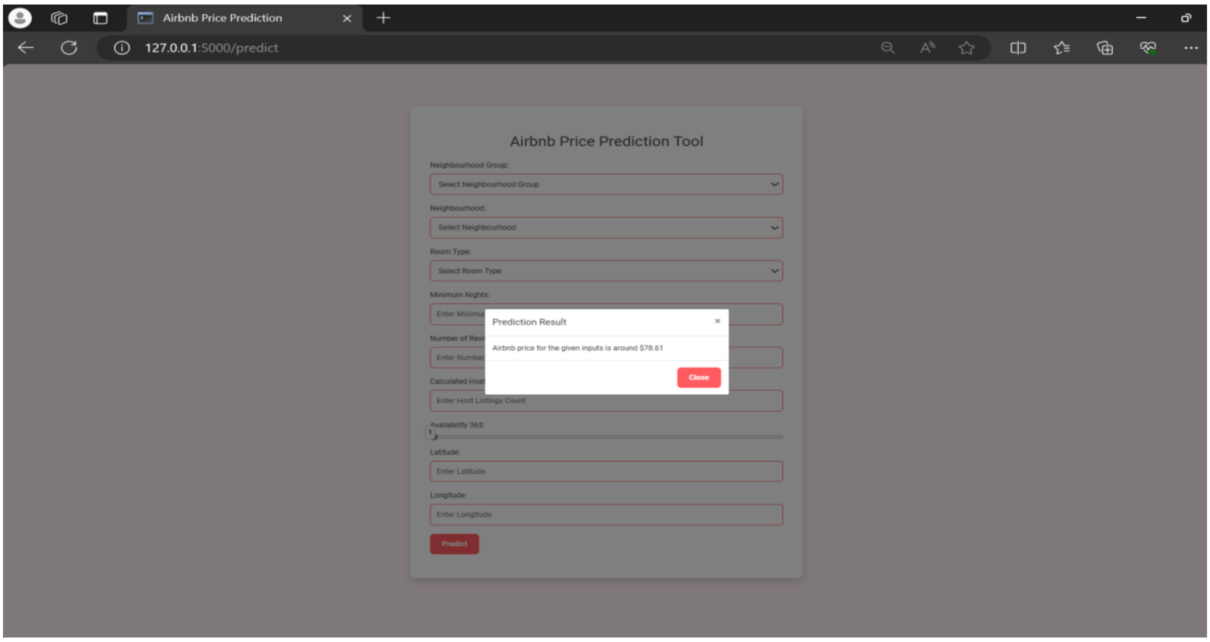
After filling out the form, submit it to trigger a POST request to the /generate_visualization route based on user selection.



Then after clicking the prediction tool in the home page, get the form for entering Airbnb listing details. The form fields should include neighbourhood group, room type, minimum nights, number of reviews, etc.



The Flask backend will process this request, using the loaded Random Forest model to predict the price based on the input values.



b. Model Selection:

In phase 2 of the project, we explored several models for predicting Airbnb listing prices: Decision Tree, Polynomial Regression, K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Linear Regression, Random Forest, and Ridge Regression. After comprehensive training and evaluation, the Random Forest algorithm was selected as our primary model due to its superior effectiveness compared to other models. The Random Forest model was chosen because it excels in handling the intricate relationships within datasets and is particularly adept at capturing non-linear patterns. As an ensemble learning method, it combines the strengths of multiple decision trees, enhancing its predictive capabilities.

The dataset used for predicting Airbnb prices contains a mixture of categorical and numerical features, and the Random Forest model is well-suited to handle these diverse data types. This allows for seamless integration and interpretation of varied information. Additionally, Random Forest is robust against outliers, which helps maintain model performance even in the presence of anomalies in features like the number of reviews or availability days.

To prevent the risk of overfitting, we implemented techniques like max depth limitation and the setting of the minimum number of samples required at a leaf node during the training process. These techniques help prevent the model from excessively learning noise in the training data, thereby enhancing its generalization capabilities.

To optimize the Random Forest model, we performed parameter tuning. The selected parameters and their values were carefully adjusted to balance complexity and learning ability, including:

Number of Trees: 100
Max Depth: None (allowing unlimited growth of trees where necessary)
Min Samples Split: 2
Min Samples Leaf: 1
The Random Forest model demonstrated high performance metrics, achieving an R-squared score significantly above other models and maintaining competitive error rates, indicative of its robust predictive power and reliability in this application scenario.

c. Recommendations for User Empowerment and Project Enhancement:

From the Airbnb price prediction form, users can input the specific details of their choice, and the model provides valuable insights into how specific features influence Airbnb listing prices. For example, understanding the correlation between a listing's location (e.g., neighbourhood group) and its price provides insights into how attributes like room type and minimum nights affect the price. This enables users to more effectively strategize their listing prices or choose where to book based on budget constraints.

The product serves as a decision-support tool that helps users make informed choices. Hosts equipped with insights into reasonable pricing based on various features can check whether their listing prices are competitive, while guests can verify if the listed prices are reasonable. Understanding each feature's impact helps hosts strategically set competitive prices based on the predicted values, increasing their chances of attracting bookings and maintaining profitability.

To further enhance the project, we could incorporate real-time market data, which holds the potential to significantly improve the accuracy of the model. Incorporating factors like current hospitality trends, peak and off-peak seasons, and even local events can improve the model's usability and relevance in dynamic market conditions. Additionally, integrating user reviews and ratings for specific properties could provide a comprehensive perspective, adding a qualitative dimension to the quantitative predictions. This qualitative aspect is instrumental in addressing user concerns related to satisfaction and property quality, contributing to a more user-centric experience.

Furthermore, the expansion of the model could involve the consideration of external economic factors like local economic conditions, tourism rates, and regional regulatory changes. Including these factors in the predictive model can provide users with valuable insights into the broader economic context and how it influences fluctuations in Airbnb prices. By providing more visualizations in the user interface on how predicted prices are impacted by various features, users can gain a deeper understanding of the factors impacting pricing dynamics in the Airbnb market. This visual approach aims to empower users with actionable insights, enabling them to make more informed decisions when navigating the complexities of the hospitality industry.

This adaptation emphasizes the potential benefits and enhancements for the Airbnb price prediction tool, aligning it with our project's specific needs and goals.