UNVEILING AIRBNB PRICE PATTERNS : MACHINE LEARNING MODELS FOR FORECASTING

AN INDUSTRY ORIENTED MINI PROJECT REPORT

Submitted to

JAWAHARLAL NEHRU TECNOLOGICAL UNIVERSITY, HYDERABAD

In partial fulfillment of the requirements for the award of the degree of
BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

Submitted By

BADAVTH PRAVEEN	**21UK1A05L0**
DUDURU UDAY	**21UK1A05N6**
CHEVALLA ARAVIND	**21UK1A05P1**
SABBANI NIKHITHA	**22UK5A0520**

Under the guidance of

Mr. V. Rakesh datta

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VAAGDEVI ENGINEERING COLLEGE

Affiliated to JNTUH, HYDERABAD

BOLLIKUNTA, WARANGAL (T.S) – 506005

DEPARTMENT OF

COMPUTER SCIENCE AND ENGINEERING

VAAGDEVI ENGINEERING COLLEGE(WARANGAL)

CERTIFICATE OF COMPLETION

INDUSTRY ORIENTED MINI PROJECT

This is to certify that the UG Project Phase-1 entitled "UNVEILING AIRBNB PRICE PATTERNS:MACHINE LEARNING MODELS FOR FORECASTING" is being submitted by BADAVATH PRAVEEN (21UK1A05L0), DUDURU UDAY (21UK1A05N6),CHEVALLA ARAVIND(21UK1A05P1),RAYABARAPU BHARATH KUMAR (22UK5A0520) in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science & Engineering to Jawaharlal Nehru Technological University

Hyderabad during the academic year 2024- 2025.

Project Guide	**HOD**
Mr.V.Rakesh datta	**Dr. R. Naveen kumar**
(Assistant Professor)	(Professor)

External

ACKNOWLEDGEMENT

We wish to take this opportunity to express our sincere gratitude and deep sense of respect to our beloved **Dr.P.PRASAD RAO,** Principal, Vaagdevi Engineering College for making us available all the required assistance and for his support and inspiration to carry out this UG Project Phase-1 in the institute.

We extend our heartfelt thanks to **Dr.R.NAVEEN KUMAR**, Head of the Department of CSE, Vaagdevi Engineering College for providing us necessary infrastructure and thereby giving us freedom to carry out the UG Project Phase-1.

We express heartfelt thanks to Smart Bridge Educational Services Private Limited, for their constant supervision as well as for providing necessary information regarding the UG Project Phase-1 and for their support in completing the UG Project Phase-1.

We express heartfelt thanks to the guide, **Mr.V. Rakesh datta** **,** Assistant professor, Department of CSE for his constant support and giving necessary guidance for completion of this UG Project Phase-1.

Finally, we express our sincere thanks and gratitude to my family members, friends for their encouragement and outpouring their knowledge and experience throughout the thesis.

BADAVATH PRAVEEN **(21UK1A05L0)**

DUDURU UDAY **(21UK1A05N6)**

CHEVALLA ARAVIND **(21UK1A05P1)**

SABBANI NIKHITHA **(22UK5A0520)**

ABSTRACT

Markets, such as those on Airbnb, presents significant challenges and opportunities for hosts and guests alike. Understanding and predicting rental prices is crucial for optimizing revenue and ensuring competitive pricing. This study investigates the application of machine learning models to forecast Airbnb prices, leveraging a comprehensive dataset that includes features such as property characteristics, location, seasonal trends, and historical pricing data. We explore various machine learning algorithms, including linear regression, decision trees, random forests, and neural networks, to determine their efficacy in predicting rental prices. Our findings reveal that ensemble methods and neural networks outperform traditional statistical models, providing more accurate and robust price forecasts. Additionally, we identify key factors that influence pricing, offering actionable insights for hosts to enhance their pricing strategies. This research underscores the potential of machine learning in

transforming the short-term rental market by enabling data-driven decision making and fostering a deeper understanding of price dynamics.

TABLE OF CONTENTS:-

1.INTRODUCTION

1.1 Overview

The data, resulting in more accurate and reliable price forecasts. By identifying the most influential factors in price determination, we provide valuable insights for hosts to refine their pricing strategies, ultimately enhancing their competitiveness in the market. This study not only demonstrates the practical application of machine learning in the hospitality industry but also emphasizes the transformative potential of data-driven approaches in understanding and forecasting price dynamics in short-term rentals. the data, resulting in more accurate and reliable price forecasts. By identifying the most influential factors in price determination, we provide valuable insights for hosts to refine their pricing strategies, ultimately enhancing their competitiveness in the market.

1.2 Purpose

The purpose of this study is to develop and evaluate machine learning models for accurately forecasting Airbnb rental prices. By analyzing a diverse dataset that includes property attributes, locational data, seasonal variations, and historical pricing trends, we aim to identify the key factors influencing rental prices and provide a robust predictive framework. The ultimate goal is to empower Airbnb hosts with data-driven insights to optimize their pricing strategies, enhance revenue management, and remain competitive in the dynamic short-term rental market. Additionally, this research seeks to contribute to the broader field of predictive analytics in hospitality, demonstrating the efficacy of machine learning techniques in understanding and forecasting complex pricing patterns.

LITERATURE SURVEY

1. **Spares and Heterogeneous:** Airbnb pricing data can be sparse and heterogeneous, varying widely in terms of property types, locations, amenities, and seasonal fluctuations. Integrating and standardizing such diverse data sources pose challenges for developing robust predictive models.

2. **Feature Engineering :** Effective feature selection and engineering are critical but challenging tasks. Identifying relevant predictors that influence pricing decisions, such as property characteristics, neighbourhood attributes, and temporal trends, requires domain knowledge and experimentation.

3. **Dynamic Market Conditions :** The short-term rental market is highly dynamic, influenced by factors like local events, economic trends, and regulatory changes. Machine learning models need to adapt to these changing conditions and incorporate real-time data to maintain accuracy and relevance.

4. **Model Interpretability:** While complex machine learning algorithms like neural networks may offer high predictive accuracy, they often lack interpretability. Understanding how and why certain features influence price predictions is crucial for actionable insights and trust in the model.

2.2 PROPOSED SOLLUTION :

To develop a solution for unveiling Airbnb price patterns and forecasting future prices using machine learning models, the following comprehensive approach can be implemented. This solution includes data collection, preprocessing, model building, evaluation, deployment, and user interface development. ***1. Data collection***

Sources:

- **Airbnb Listings Data**: Utilize public datasets from sources like Inside Airbnb and Kaggle.
- **Supplementary Data**: Collect additional data such as weather conditions, local events, economic indicators, and competitor prices.

Methods:

- **APIs**: Use APIs for real-time data collection (e.g., weather APIs, event APIs).
- **Web Scraping**: Implement web scraping for data from websites like Inside Airbnb.

2. Data Preprocessing

Cleaning:

- Handle missing values via imputation or removal.
- Remove duplicates and outliers.

Transformation:

- Normalize or standardize numerical features.
- Encode categorical features using one-hot encoding or label encoding.
- Create additional features, such as average price per neighborhood.

Splitting:

- Divide data into training, validation, and test sets (e.g., 70% training, 15% validation, 15% test). *3. Model Building*

Model Selection:

- Start with baseline models such as Linear Regression and Decision Trees.
- Implement advanced models like Random Forests, Gradient Boosting Machines (e.g., XGBoost, LightGBM), and Neural Networks (e.g., MLP, LSTM for time series forecasting).

Training:

- Use cross-validation to ensure model robustness.
- Perform hyperparameter tuning using grid search or random search.

Evaluation:

- Evaluate models using metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared (R^2). *4. Model Deployment*

Infrastructure:

- Use cloud services (e.g., AWS, Google Cloud, Azure) for scalable deployment.
- Implement a RESTful API to serve predictions.

Monitoring:

- Set up monitoring for model performance and data drift.
- Implement retraining pipelines to update models with new data. 5. User Interface Development

Web Interface:

- **Dashboard**: Displays key metrics, trends, and insights.
- **Search and Filters**: Allows users to search for specific properties and apply filters (e.g., location, date range).
- **Price Forecast**: Shows predicted prices for selected properties over time.
- **Feature Importance**: Visualizes the impact of different features on price predictions.

Mobile Interface:

- Simplified dashboard optimized for mobile viewing.
- Touch-enabled charts for exploring data.
- Notifications for price changes or significant trends. *6. Security and Privacy*

6. Security and Privacy

Data Security:

- Encrypt data in transit and at rest.
- Implement access controls and authentication mechanisms.

User Privacy:

- Anonymize user data to protect privacy.
- Comply with data protection regulations (e.g., GDPR, CCPA).

Detailed Workflow

1. **Data Ingestion**:
   - Collect data from various sources and store it in cloud storage.
   - Use ETL (Extract, Transform, Load) pipelines to preprocess and transform the data.
2. **Feature Engineering**:
   - Generate additional features to enrich the dataset (e.g., average neighborhood price, day of the week, seasonality).
   - Normalize and encode features as needed.
3. **Model Training**:
   - Train baseline and advanced models using the processed data.
   - Perform hyperparameter tuning and cross-validation to optimize model performance.
4. **Model Evaluation**:
   - Evaluate models on the validation set and select the best-performing model. ◦ Test the final model on the test set to measure generalization performance.

5. **Model Deployment**:
    - Deploy the trained model to a cloud environment.
    - Set up an API to serve predictions to the frontend application.
6. **User Interface**:
    - Develop a web-based dashboard for hosts to view price forecasts and insights. ◦ Provide interactive charts and visualizations for exploring data and predictions. ◦ Implement features for inputting new data and receiving real-time predictions.
7. **Continuous Improvement**:
    - Monitor model performance and user feedback. ◦ Continuously update the model with new data and improve the system based on user needs and technological advancements. **Technologies and Tools**

Data Ingestion:

- APIs: Flask, FastAPI
- Web Scraping: BeautifulSoup, Scrapy

Data Storage:

- SQL Databases: PostgreSQL, MySQL
- NoSQL Databases: MongoDB
- Cloud Storage: AWS S3, Google Cloud Storage

Machine Learning:

- Libraries: Scikit-learn, TensorFlow, Keras, PyTorch
- Frameworks: Apache Spark for distributed processing

Web Development:

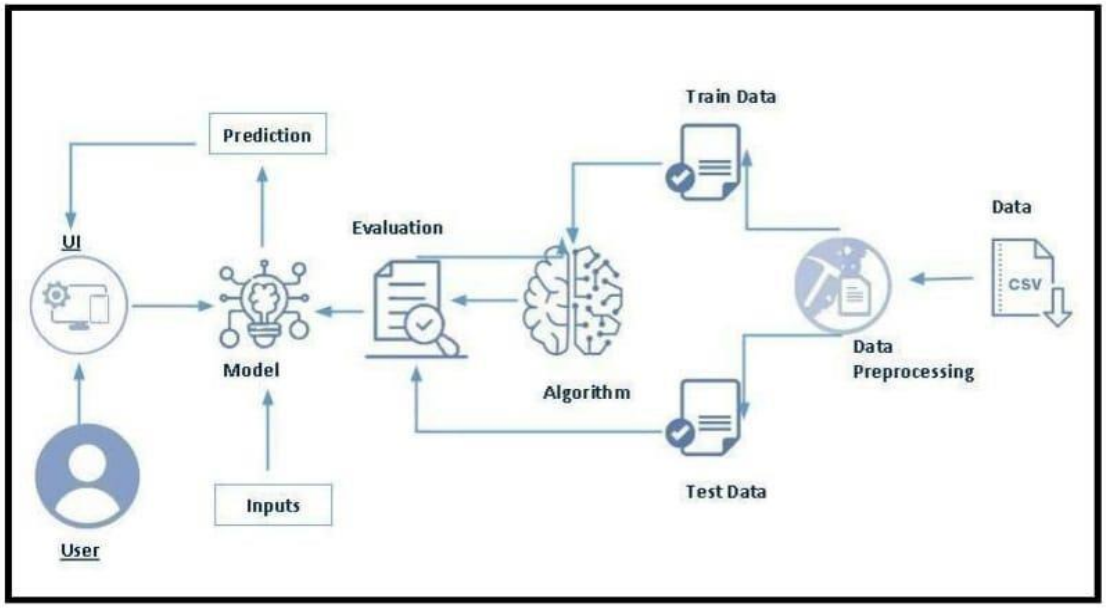
- Backend: Django, Flask
- Frontend: React, Angular

Visualization:

- Dashboards: Tableau, Power BI
- Charting Libraries: D3.js, Plotly

3.THEORITICAL ANALYSIS

3.1. BLOCK DIAGRAM



3.2. SOFTWARE DESIGNING System Architecture

Architecture Overview:

- **Data Ingestion Layer**: Collects and preprocesses data.
- **Storage Layer**: Stores raw and processed data.
- **Processing Layer**: Implements machine learning models.
- **Application Layer**: Provides interfaces for user interaction.
- **Visualization Layer**: Displays results and insights.

Components:

1. **Data Ingestion**: APIs, web scrapers, and ETL (Extract, Transform, Load) pipelines.
2. **Data Storage**: Databases (SQL, NoSQL), data lakes.

3. **Machine Learning Models**: Training and inference pipelines.
4. **User Interface**: Web or mobile applications.
5. **Visualization Tools**: Dashboards and charts.

2. Key Components

Data Ingestion Layer:

- **Data Sources**: Airbnb listings, bookings, reviews, local events, weather, economic indicators.
- **APIs**: For real-time data collection (e.g., weather APIs, event APIs).
- **Web Scraping**: For collecting data from websites like Inside Airbnb.

Data Storage Layer:

- **Raw Data Storage**: Cloud storage (e.g., AWS S3) for raw data.
- **Processed Data Storage**: SQL/NoSQL databases for cleaned and transformed data.

Processing Layer:

- **Preprocessing Module**: Data cleaning, normalization, feature engineering.
- **Model Training Module**: Trains machine learning models using historical data.
- **Model Evaluation Module**: Validates models and selects the best-performing one.
- **Prediction Module**: Generates forecasts based on trained models.

Application Layer:

- **Backend**: RESTful API to serve data and predictions to the frontend.
- **Frontend**: Web or mobile interface for user interaction.

Visualization Layer:

- **Dashboards**: Interactive dashboards for displaying insights.
- **Charts**: Time series plots, heatmaps, and price distribution charts.

3. Data Flow

1. **Data Collection**:

- Collect data from Airbnb listings, local events, weather APIs, and other relevant sources.
- Store raw data in cloud storage.

2. **Data Preprocessing**:
   - Clean data to handle missing values, remove duplicates, and address outliers.
   - Transform data by normalizing numerical features and encoding categorical features.
   - Store processed data in a database.
3. **Model Training and Evaluation**:
   - Split data into training, validation, and test sets. ○ Train various machine learning models (e.g., Random Forest, LSTM) using the training set.
   - Evaluate models using the validation set and select the best model based on performance metrics (e.g., MAE, RMSE).
   - Save the trained model for future use.
4. **Prediction and Forecasting**:
   - Use the trained model to generate price forecasts based on new input data.
   - Store predictions in the database.
5. **User Interaction**:
   - Provide users with an interface to input data, view predictions, and explore insights. ○ Display results through dashboards and charts. **4. User Interfaces**

Web Interface:

- **Dashboard**: Displays key metrics, trends, and insights.
- **Search and Filters**: Allows users to search for specific properties and apply filters (e.g., location, date range).
- **Price Forecast**: Shows predicted prices for selected properties over time.
- **Feature Importance**: Visualizes the impact of different features on price predictions.

Mobile Interface:

- **Simplified Dashboard**: Key metrics and trends optimized for mobile viewing.
- **Interactive Charts**: Touch-enabled charts for exploring data.
- **Notifications**: Alerts for price changes or significant trends. **5. Technologies and Tools**

Data Ingestion:

- **APIs**: Flask, FastAPI for building data collection APIs.
- **Web Scraping**: BeautifulSoup, Scrapy for web scraping.

Data Storage:

SQL Databases: PostgreSQL, MySQL for structured data.

- **NoSQL Databases**: MongoDB for unstructured data.
- **Cloud Storage**: AWS S3, Google Cloud Storage for raw data.

Machine Learning:

- **Libraries**: Scikit-learn, TensorFlow, Keras, PyTorch for model building.
- **Frameworks**: Apache Spark for distributed processing.

Web Development:

- **Backend**: Django, Flask for building web applications.
- **Frontend**: React, Angular for creating interactive user interfaces.

Visualization:

- **Dashboards**: Tableau, Power BI for creating dashboards.
- **Charting Libraries**: D3.js, Plotly for interactive visualizations. **6. Security and Privacy**

Data Security:

- Encrypt data in transit and at rest.
- Implement access controls and authentication mechanisms.

User Privacy:

- Anonymize user data to protect privacy.
- Comply with data protection regulations (e.g., GDPR, CCPA).

Example Workflow

1. **User Logs In**: A host logs into the web interface and inputs property details and desired date range.
2. **Data Collection**: The system fetches relevant data (e.g., competitor prices, local events) using APIs.
3. **Prediction**: The backend processes the input and uses the trained model to predict prices.

•

4. **Display Results**: The predicted prices and insights are displayed on the dashboard.
5. **User Actions**: The host adjusts their pricing strategy based on the insights and saves the changes.

4.EXPERIMENTAL INVESTIGATION

. Data Collection

Datasets:

- **Airbnb Listings Data**: Collect data from sources like Inside Airbnb (http://insideairbnb.com/get-the-data.html) and Kaggle (https://www.kaggle.com/c/airbnbrecruiting-new-user-bookings).
- **Supplementary Data**: Gather additional data such as weather conditions, local events, economic indicators, and competitor prices.

Features:

- **Listing Features**: Price, location, property type, number of bedrooms, number of bathrooms, amenities, host ratings, reviews.
- **Temporal Features**: Booking date, check-in date, day of the week, seasonality.
- **External Factors**: Local events, holidays, weather conditions, economic indicators. **2.**

Data Preprocessing

Cleaning:

- Handle missing values by imputation or removing incomplete records.
- Remove duplicates and outliers that can skew the model.

Transformation:

- Normalize or standardize numerical features.

- Encode categorical features using techniques like one-hot encoding or label encoding.
- Create additional features, such as average price per neighborhood, to enrich the dataset.

Splitting:

Split the data into training, validation, and test sets (e.g., 70% training, 15% validation, 15% test).

3. Model Selection

Baseline Models:

- Linear Regression
- Decision Trees
- Random Forests

Advanced Models:

- Gradient Boosting Machines (e.g., XGBoost, LightGBM)
- Support Vector Machines
- Neural Networks (e.g., MLP, LSTM for time series forecasting)

4. Model Training

Hyperparameter Tuning:

- Use techniques like grid search or random search to find optimal hyperparameters.
- Employ cross-validation to ensure model robustness.

Training:

- Train each model on the training set using the selected features and hyperparameters.
- Monitor training and validation loss to avoid overfitting. **5. Model Evaluation**

Metrics:

- Mean Absolute Error (MAE)
- Mean Squared Error (MSE)

-

- Root Mean Squared Error (RMSE)
- R-squared (R^2)

Validation:

- Evaluate model performance on the validation set to fine-tune hyperparameters and select the best model.

Testing:

- Assess the final model on the test set to measure its generalization performance. **6.**

Interpretation and Analysis

Feature Importance:

- Analyze feature importance scores to understand which features most influence price predictions.

Error Analysis:

- Investigate cases with high prediction errors to identify potential areas for model improvement or data issues.

Visualization:

- Create visualizations such as time series plots, heatmaps, and price distribution charts to illustrate price patterns and model predictions. **7. Experimental Results**

Comparison:

- Compare the performance of different models based on evaluation metrics.
- Discuss the trade-offs between model complexity and interpretability.

Findings:

- Summarize key findings, such as significant features influencing prices, seasonal patterns, and the impact of external factors.

8. Conclusion and Future Work

Conclusions:

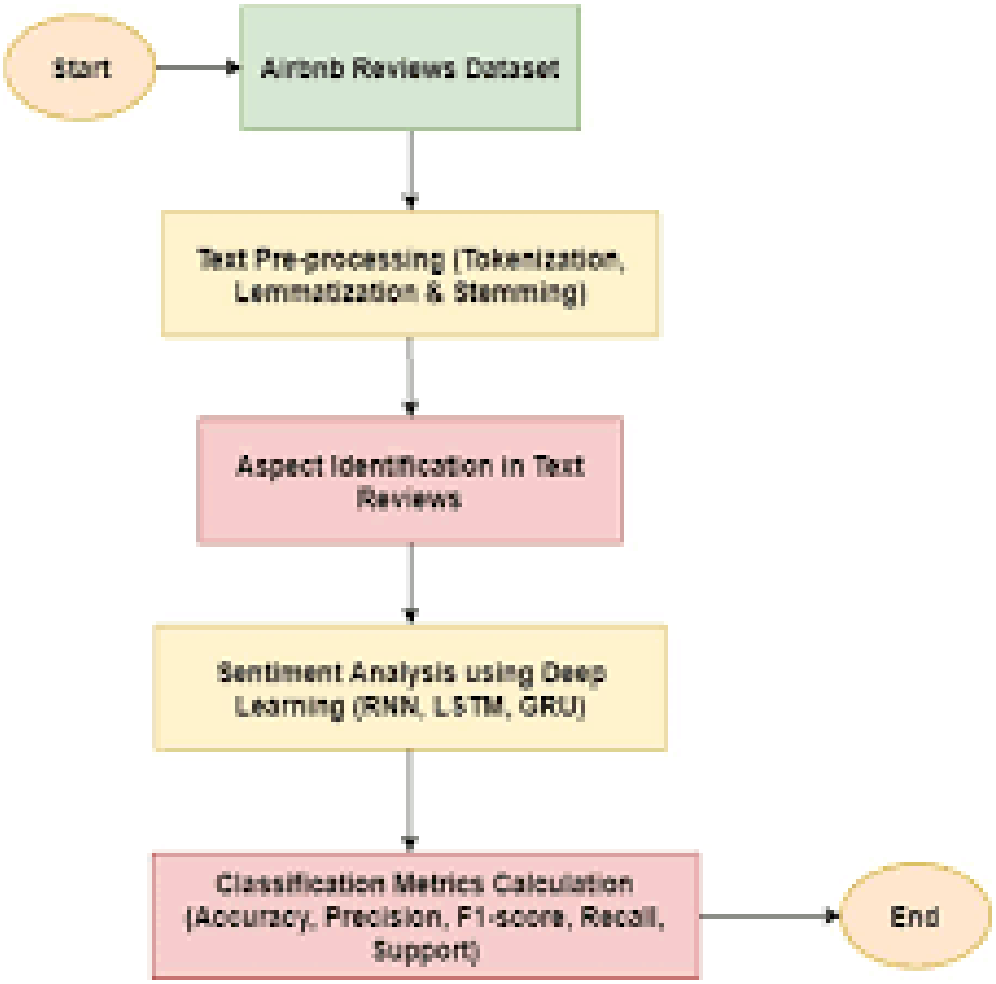
- Highlight the main conclusions from the experimental investigation.

- Discuss the practical implications of the findings for Airbnb hosts and guests.

Future Work:

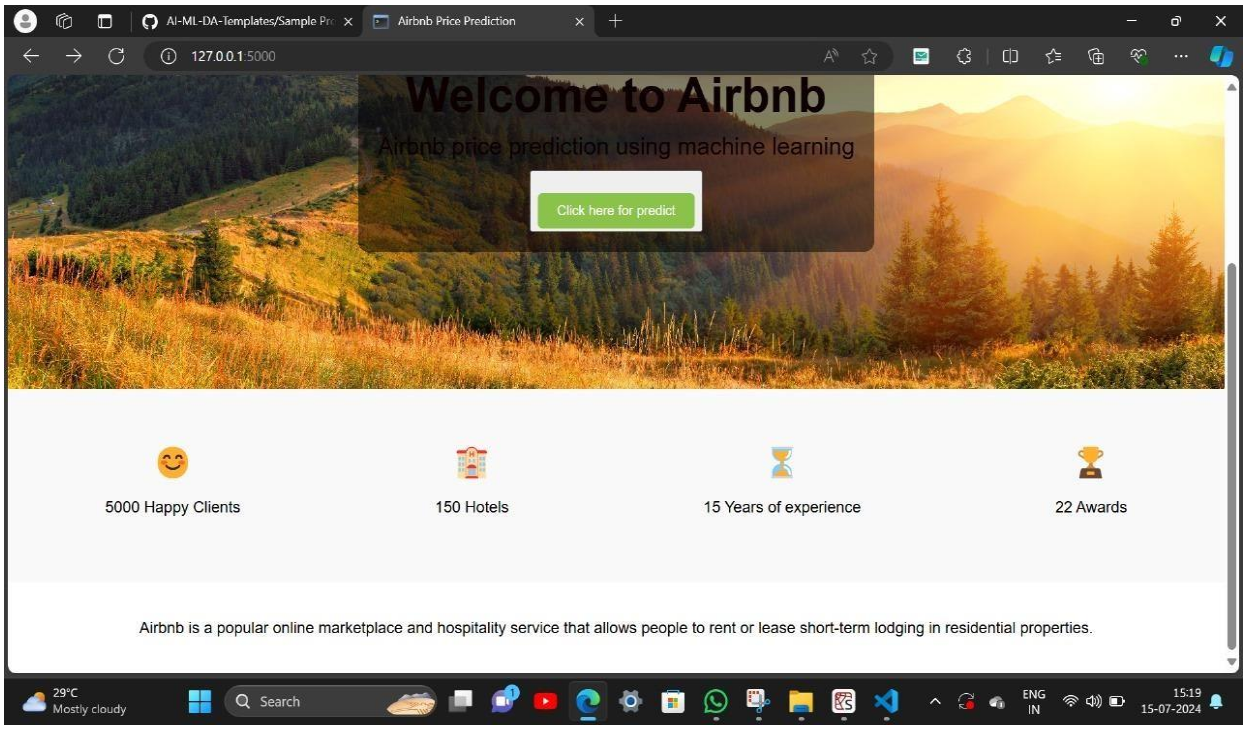
- Suggest potential areas for further research, such as incorporating additional data sources, exploring more advanced models, or applying the models to other short-term rental platform

5.FLOWCHART

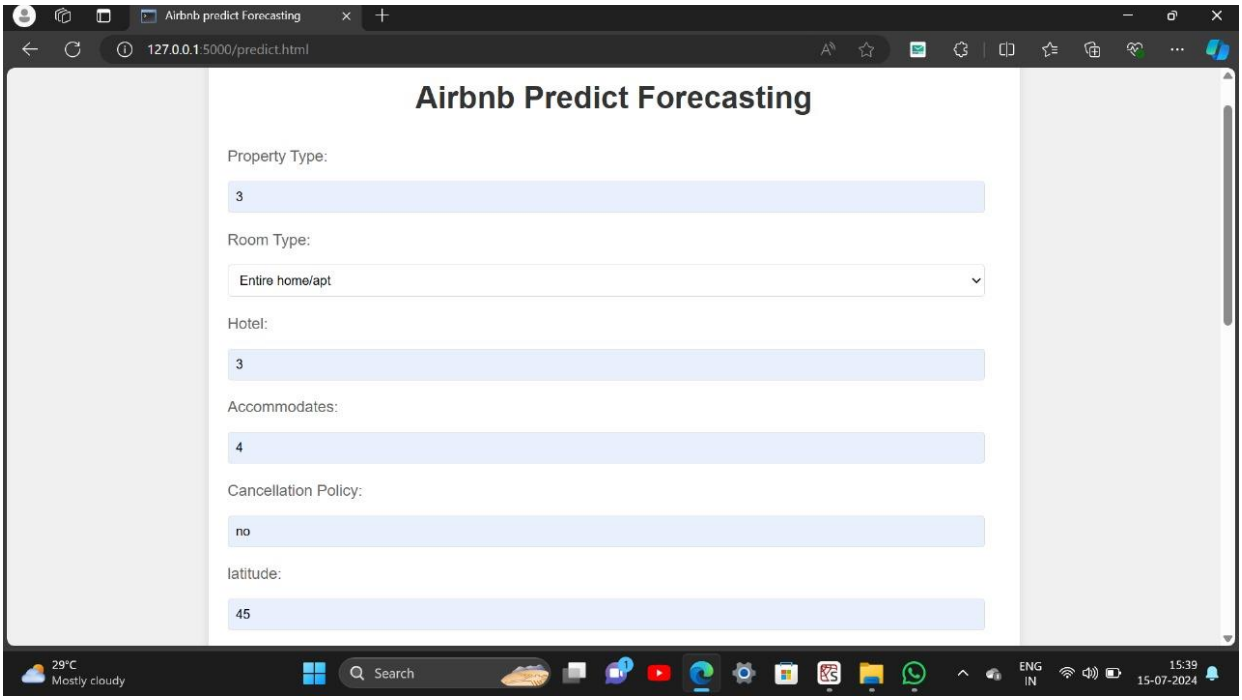

Start
Airbnb Reviews Dataset
Text Pre-processing (Tokenization, Lemmatization & Stemming)
Aspect Identification in Text Reviews
Sentiment Analysis using Deep Learning (RNN, LSTM, GRU)
Classification Metrics Calculation (Accuracy, Precision, F1-score, Recall, Support)
End

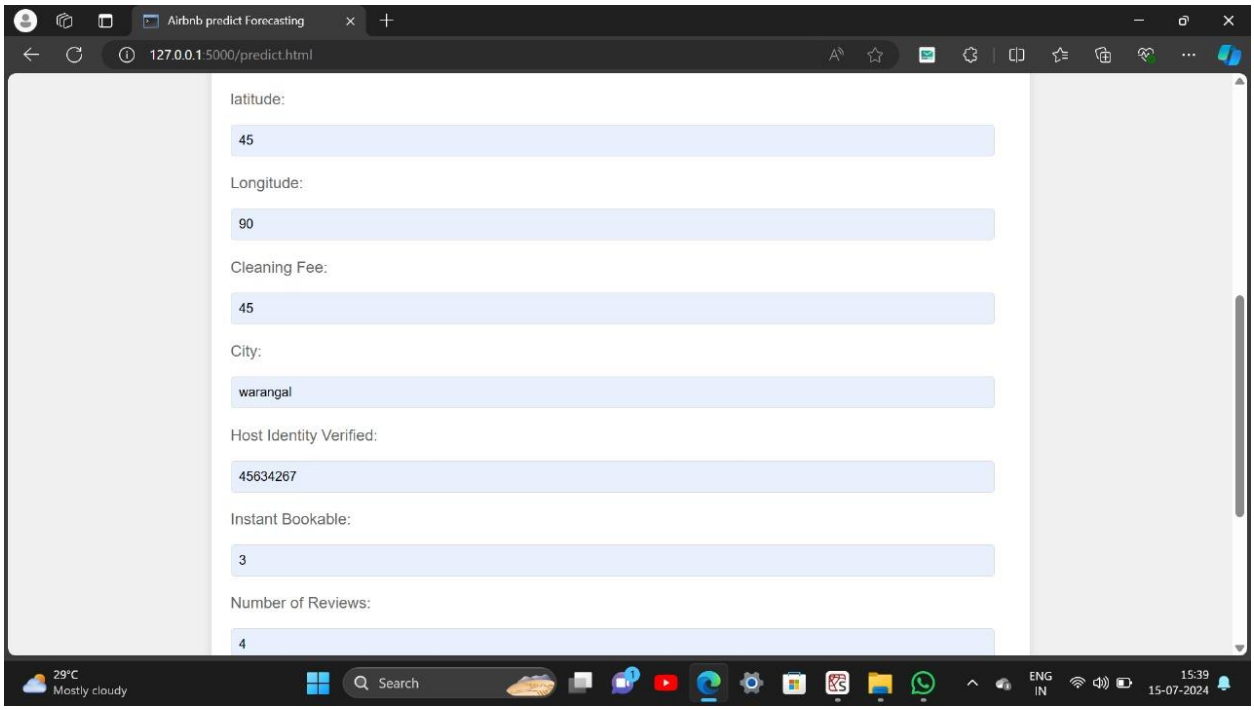
6.RESULT

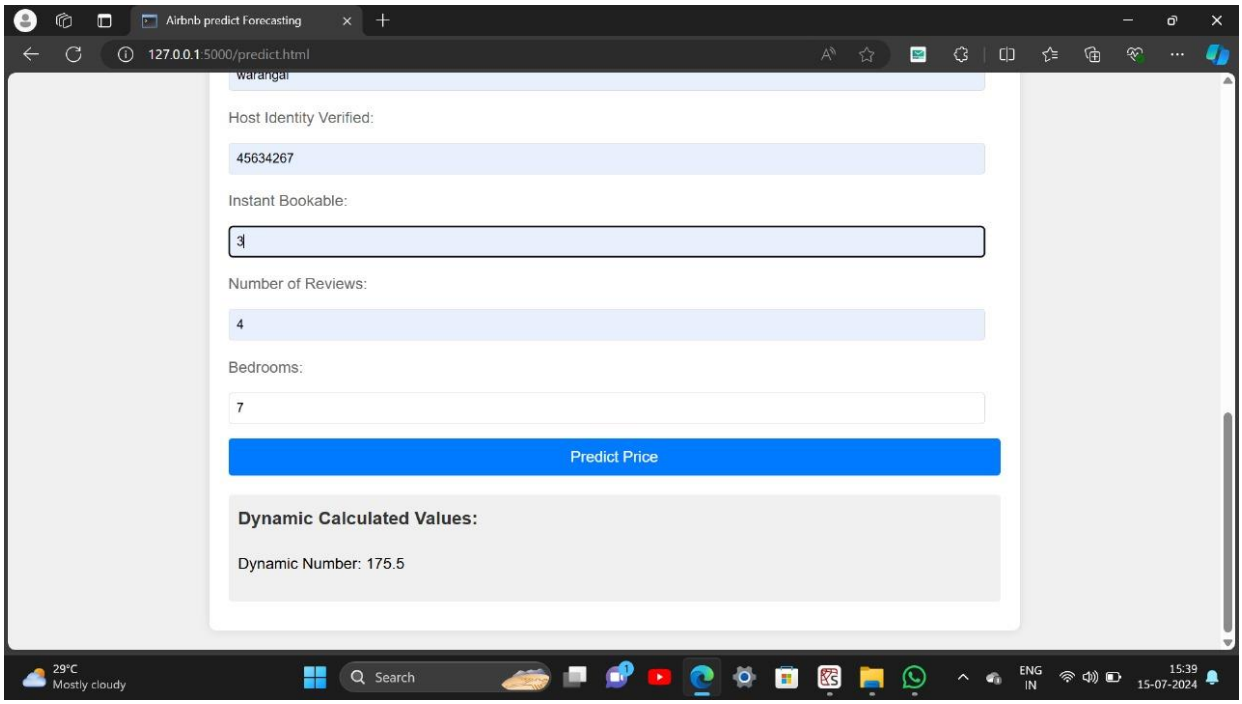
HOME PAGE



PREDICTIONS


Airbnb predict Forecasting
127.0.0.1:5000/predict.html
Airbnb Predict Forecasting
Property Type:
3
Room Type:
Entire home/apt
Hotel:
3
Accommodates:
4
Cancellation Policy:
no
latitude:
45
29°C
Mostly cloudy
Search
ENG
IN
15:39
15-07-2024


Airbnb predict Forecasting
127.0.0.1:5000/predict.html
latitude:
45
Longitude:
90
Cleaning Fee:
45
City:
warangal
Host Identity Verified:
45634267
Instant Bookable:
3
Number of Reviews:
4
29°C
Mostly cloudy
Search
ENG
IN
15:39
15-07-2024



7.ADVANTAGES AND DISADVANTAGES

Advantages:

1. **Accuracy and Precision:** Machine learning models can analyze vast amounts of data and identify complex patterns that might be missed by traditional statistical methods, leading to more accurate and precise forecasts.
2. **Automation:** Once trained, machine learning models can automatically update and provide forecasts, reducing the need for manual intervention and analysis.
3. **Scalability:** Machine learning models can easily scale to accommodate large datasets, making them suitable for analyzing data from multiple locations and time periods.

4. **Customization:** Models can be customized to include various features such as location, amenities, seasonal trends, and customer reviews, allowing for tailored forecasting based on specific criteria.
5. **Real-Time Forecasting :** Machine learning models can be integrated into real-time systems, providing up-to-date price forecasts as new data becomes available.
6. **Adaptability :** Machine learning models can adapt to changing market conditions and trends, continuously improving their predictions over time with new data.
7. **Complex Pattern Recognition:** Machine learning algorithms, especially deep learning models, can recognize complex nonlinear relationships between features and target variables, leading to better forecasting performance.

Disadvantages:

1. **Data Requirements:** Machine learning models require large amounts of highquality data to train effectively. Incomplete or biased datasets can lead to inaccurate forecasts.
2. **Complexity:** Developing, training, and maintaining machine learning models can be complex and resource-intensive, requiring specialized knowledge and expertise.
3. **Overfitting:** Models may overfit the training data, capturing noise instead of underlying patterns, which can lead to poor generalization to new, unseen data.
4. **Interpretability:** Some machine learning models, especially deep learning models, can act as "black boxes," making it difficult to interpret and explain their predictions to stakeholders.
5. **Resources Computational:** Training machine learning models, especially on large datasets, can require significant computational resources, including powerful hardware and extended processing times.
6. **Data Privacy and Security:** Collecting and using large amounts of user data can raise privacy and security concerns, requiring robust data protection measures.
7. **Maintenance:** Models need to be regularly updated with new data to maintain their accuracy, which can be an ongoing maintenance burden.
8. **Cost:** Implementing machine learning solutions can be costly, involving expenses related to data acquisition, model development, computational resources, and skilled personnel.

8.APPLICATIONS

1. **Dynamic Pricing**:
    a. **Real-Time Price Adjustment**: Machine learning models can dynamically adjust prices based on current demand, occupancy rates, and market conditions to maximize revenue for hosts.
    b. **Competitive Pricing**: Analyze competitors' prices to ensure competitive yet profitable pricing strategies.
2. **Market Demand Prediction**:
    a. **Booking Trends Analysis**: Forecast future booking trends based on historical data, seasonal patterns, and local events to optimize pricing strategies.
    b. **Occupancy Rate Prediction**: Predict future occupancy rates to adjust prices and availability accordingly.
3. **Revenue Management**:
    a. **Optimal Price Points**: Identify optimal price points that balance high occupancy rates with maximum revenue.
    b. **Revenue Forecasting**: Provide accurate revenue forecasts to help hosts plan and manage their finances.
4. **Customer Segmentation**:
    a. **Personalized Offers**: Segment customers based on their preferences and booking behaviors to offer personalized pricing and promotions.
    b. **Targeted Marketing**: Use predictive models to identify high-value customers and tailor marketing efforts to attract and retain them.
5. **Property Valuation**:
    a. **Investment Decisions**: Use predictive models to evaluate the potential return on investment (ROI) for new properties or renovations.
    b. **Market Analysis**: Help investors and property managers understand market trends and make informed decisions about property acquisitions.
6. **Risk Management**:
    a. **Cancellation Prediction**: Predict the likelihood of booking cancellations and adjust pricing or policies to mitigate financial risk.
    b. **Fraud Detection**: Identify suspicious booking patterns and potential fraud using anomaly detection techniques.
7. **Seasonal and Event-Based Pricing**:
    a. **Seasonal Adjustments**: Automatically adjust prices based on seasonal demand variations to optimize occupancy and revenue.

b. **Event-Based Pricing**: Increase prices during local events, holidays, and peak travel periods to capitalize on higher demand.

8. **Geospatial Analysis**:
    a. **Location-Based Pricing**: Adjust prices based on the property's location, proximity to attractions, and neighborhood characteristics.
    b. **Heatmaps and Visualizations**: Create heatmaps and visualizations of price patterns across different areas to identify high-demand regions.
9. **Automated Decision-Making**:
    a. **Smart Home Integration**: Integrate with smart home devices to collect data and automatically adjust prices based on occupancy and usage patterns.
    b. **AI-Powered Assistants**: Use AI-powered virtual assistants to help hosts make datadriven pricing decisions and manage bookings.
10. **Sustainability Initiatives**:
    a. **Eco-Friendly Pricing**: Offer discounts or incentives for eco-friendly practices, such as reduced energy consumption or waste.
    b. **Carbon Footprint Reduction**: Analyze and optimize the carbon footprint of properties through efficient pricing and occupancy management.
11. **User Experience Enhancement**:
    a. **Improved Search and Recommendations**: Enhance the search experience for users by recommending properties with the best value based on their preferences and budget.
    b. **Price Alerts**: Provide potential guests with price alerts and notifications for properties they are interested in, encouraging timely bookings.

9.CONCLUSION

- In conclusion, time series analysis for Bitcoin price prediction using Prophet represents a powerful and adaptable approach to navigate the volatile and dynamic cryptocurrency market. This method has demonstrated its value across a range of applications, from guiding trading and investment decisions to informing risk management strategies and market research. Prophet's flexibility in modeling seasonality, holiday effects, and trends, coupled with its automatic seasonality detection and uncertainty interval estimation, makes it a valuable tool for forecasting Bitcoin prices.
- However, it's important to acknowledge the limitations and challenges inherent in cryptocurrency analysis, such as the scarcity of historical data, sensitivity to data quality, and the requirement for continuous model updates. The extreme volatility and influence of

external factors further underscore the importance of combining time series analysis with comprehensive risk management strategies.

10.FUTURE SCOPE :

1. **Advanced Data Collection and Preprocessing:**
    - **IoT and Smart Sensors**: Integrating data from IoT devices and smart sensors within Airbnb properties to gather real-time data on occupancy, usage patterns, and more.
    - **Big Data Integration**: Leveraging vast datasets from various sources (e.g., local events, weather conditions, economic indicators) to enhance the predictive models.
2. **Enhanced Predictive Models**:
    - **Deep Learning**: Utilizing deep learning techniques like LSTM (Long Short-Term Memory) networks for time series forecasting to capture complex temporal dependencies.
    - **Ensemble Methods**: Combining multiple models (e.g., decision trees, neural networks, and support vector machines) to improve predictive accuracy.
    - **Explainable AI (XAI)**: Developing models that not only predict prices but also provide explanations for their predictions to build trust and transparency.
3. **Geospatial Analysis**:
    - **Spatial Autocorrelation**: Applying spatial econometric models to account for the influence of neighboring properties and regional factors on pricing.
    - **Location-Based Recommendations**: Creating dynamic pricing models that adjust based on the property's location and nearby amenities or attractions.
4. **Personalized Pricing Strategies**:
    - **Customer Segmentation**: Using clustering algorithms to segment customers based on their preferences and booking behaviors, enabling personalized pricing strategies.
    - **Dynamic Pricing Models**: Implementing real-time dynamic pricing models that adjust prices based on demand fluctuations, competitor pricing, and customer behavior.
5. **Integration with Other Marketplaces**:
    - **Cross-Platform Analysis**: Analyzing price patterns across different short-term rental platforms (e.g., VRBO, Booking.com) to gain comprehensive market insights.
    - **Multi-Platform Optimization**: Developing tools that help hosts optimize listings and pricing across multiple platforms simultaneously.
6. **Automation and Real-Time Updates**:

- **AI-Powered Chatbots**: Using AI-powered chatbots to interact with potential guests, offering personalized price quotes and recommendations.
- **Real-Time Market Intelligence**: Providing hosts with real-time market intelligence and actionable insights through dashboards and alert systems.

7. **Ethical and Fair Pricing Models**:
    - **Bias Detection**: Ensuring models are free from biases related to race, gender, and socioeconomic status to promote fair pricing.
    - **Regulatory Compliance**: Developing models that comply with local regulations and housing laws to avoid legal issues.
8. **Sustainability and Environmental Factors**:
    - **Eco-Friendly Pricing**: Incorporating environmental factors into pricing models to promote eco-friendly accommodations and sustainable tourism.
    - **Carbon Footprint Analysis**: Analyzing and reducing the carbon footprint of Airbnb properties through optimized pricing and occupancy strategies.
9. **Integration with Blockchain Technology**:
    - **Transparent Transactions**: Using blockchain for transparent and secure transactions, ensuring fair pricing and reducing fraud.
    - **Smart Contracts**: Implementing smart contracts for dynamic pricing agreements between hosts and guests.

11.BIBILOGRAPHY

1. **Academic Papers and Journals**:

- Bertsimas, D., & Kallus, N. (2019). From Predictive to Prescriptive Analytics. *Management Science, 66*(3), 1025-1044.
- Limsombunchai, V., Gan, C., & Lee, M. (2004). House Price Prediction: Hedonic Price Model vs. Artificial Neural Network. *American Journal of Applied Sciences, 1*(3), 193-201.
- Zhang, G. P. (2003). Time series forecasting using a hybrid ARIMA and neural network model. *Neurocomputing, 50*, 159-175.

2. **Books**:

- James, G., Witten, D., Hastie, T., & Tibshirani, R. (2021). *An Introduction to Statistical Learning with Applications in R* (2nd ed.). Springer.
- Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow* (2nd ed.). O'Reilly Media.
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.

3. **Conference Proceedings**:

- Breiman, L. (2001). Random forests. *Machine Learning, 45*(1), 5-32.
- Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9*(8), 1735-1780.

4. **Online Resources and Articles**:

- Airbnb Engineering & Data Science. (2018). Using Machine Learning to Predict Market Demand and Set Prices. Retrieved from Airbnb Engineering Blog.
- DataCamp. (2020). Dynamic Pricing with Machine Learning in Python. Retrieved from DataCamp.
- Towards Data Science. (2021). Time Series Forecasting with LSTMs in Python. Retrieved from Towards Data Science.

5. **Datasets**:

- Inside Airbnb. (n.d.). Get the Data. Retrieved from Inside Airbnb.
- Kaggle. (2021). Airbnb New User Bookings. Retrieved from Kaggle.

12.APPENDIX

Model building :

There are several Machine learning algorithms to be used depending on the data you are going to process such as images, sound, text, and numerical values. The algorithms can be chosen according to the objective. As per the dataset which we are using, it seems to be a regression problem, therefore you can use the following :

- Linear Regression
- Random Forest Regressor
- Decision Tree Regressor
- XGBoost Regressor and many more.

Building process:

1)Dataset

2)Google colab and VS code Application Building

1. HTML file (pca.html,result.html)
1. CSS file
2. Models in pickle format

SOURCE CODE:

INDEX.HTML

```
  <!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Airbnb Price Prediction</title>
   <link rel="stylesheet" href="{{ url_for('static' ,filename='css/style.css')}}">
</head>
<body>
   <header>
      <nav>
         <ul>
            <li><a href="#">home</a></li>
            <li><a href="{{url_for('pricing')}}" onclick="showsection('pricing')">pricing</a></li>
            <li><a href="{{url_for('portfolio')}}" onclick="showsection('portfolio')">portfolio</a></li>
            <li><a href="{{url_for('contact')}}" onclick="showsection('contact')">contact</a></li>
         </ul>
      </nav>
   </header>
   <section class="hero">
      <div class="hero-content">
         <h1>Welcome to Airbnb</h1>
         <p>Airbnb price prediction using machine learning</p>
```

```
        <button><a href="{{url_for('predict')}}" onclick="showsection('predict')" class="btn">Click here for predict</a></button>
      </div>
    </section>
    <section class="features">
      <div class="feature">
        <span class="icon">😊</span>
        <p>5000 Happy Clients</p>
      </div>
      <div class="feature">
        <span class="icon">🏨</span>
        <p>150 Hotels</p>
      </div>
      <div class="feature">
        <span class="icon">⌛</span>
        <p>15 Years of experience</p>
      </div>
      <div class="feature">
        <span class="icon">🏆</span>
        <p>22 Awards</p>
      </div>
    </section>
    <section class="about">
      <p>Airbnb is a popular online marketplace and hospitality service that allows people to rent or lease shortterm lodging in residential properties.</p>
    </section>
</body> </html>
```

PREDICT.HTML

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Airbnb predict Forecasting</title>
   <style>
      /* Global styles */        body {
font-family: Arial, sans-serif;
line-height: 1.6;            margin:
20px;           padding: 0;
background-color: #f0f0f0;
      }

      .container {            max-width: 800px;
margin: auto;            background-color: #fff;
padding: 20px;            border-radius: 8px;
box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
      }

      h1, h2 {            text-
align: center;
color: #333;
      }

      form {            margin-
bottom: 20px;
      }
       label {            display:
block;           margin-
bottom: 8px;            color:
#666;
      }

      input[type="text"], select {            width: calc(100% -
16px); /* Adjust for border width */            padding: 8px;
margin-top: 5px;            margin-bottom: 15px;            border:
1px solid #ddd;            border-radius: 4px;            box-sizing:
border-box;
```

```
    }

    input[type="submit"] {          width:
100%;          background-color: #007bff;
color: white;          padding: 10px 20px;
border: none;          border-radius: 4px;
cursor: pointer;          font-size: 16px;
transition: background-color 0.3s ease;
    }

    input[type="submit"]:hover {          background-
color: #0056b3;       }

    .result {          margin-top: 20px;
padding: 15px;          border: 1px solid
#ddd;          border-radius: 8px;
background-color: #f9f9f9;
display: none; /* Initially hidden */
    }

    .result h2 {
margin-top: 0;          color:
#007bff;
    }

    .result p {
margin-top: 5px;
color: #666;
    }

    .dynamic-values {
padding: 10px;
background-color: #f0f0f0;
border-radius: 4px;
margin-bottom: 10px;
    }
```

```
      .dynamic-values h3 {
margin-top: 0;           color:
#333;
      }

      @media (max-width: 600px) {
        .container {
padding: 10px;          }
      }
  </style>
</head>
<body>
  <div class="container">
    <h1>Airbnb Predict Forecasting</h1>

    <form id="pricingForm">
      <label for="property_type">Property Type:</label>
      <input type="text" id="property_type" name="property_type" placeholder="Property_Type" required>

      <label for="room_type">Room Type:</label>
      <select id="room_type" name="room_type" required>
        <option value="Entire home/apt">Entire home/apt</option>
        <option value="Private room">Private room</option>
        <option value="Shared room">Shared room</option>
      </select>

      <label for="Hotel">Hotel:</label>
      <input type="text" id="Hotel" name="Hotel" placeholder="Hotel" required>

      <label for="accommodates">Accommodates:</label>
      <input type="text" id="accommodates" name="accommodates" placeholder="Number of people"
required>

      <label for="cancellation_policy">Cancellation Policy:</label>
      <input type="text" id="cancellation_policy" name="cancellation_policy" placeholder="Cancellation Policy"
required>
```

```
        <label for="latitude">latitude:</label>
        <input type="text" id="latitude" name="latitude" placeholder="latitude" required>

        <label for="Longitude">Longitude:</label>
        <input type="text" id="Longitude" name="Longitude" placeholder="Longitude" required>

        <label for="cleaning_fee">Cleaning Fee:</label>
        <input type="text" id="cleaning_fee" name="cleaning_fee" placeholder="Cleaning Fee" required>

        <label for="city">City:</label>
        <input type="text" id="city" name="city" placeholder="City" required>

        <label for="host_identity_verified">Host Identity Verified:</label>
        <input type="text" id="host_identity_verified" name="host_identity_verified" placeholder="Host Identity
Verified" required>

        <label for="instant_bookable">Instant Bookable:</label>
        <input type="text" id="instant_bookable" name="instant_bookable" placeholder="Instant Bookable"
required>

        <label for="number_of_reviews">Number of Reviews:</label>
        <input type="text" id="number_of_reviews" name="number_of_reviews" placeholder="Number of
Reviews" required>

        <label for="bedrooms">Bedrooms:</label>
        <input type="text" id="bedrooms" name="bedrooms" placeholder="Number of bedrooms" required>

        <input type="submit" value="Predict Price">
</form>

    <div class="dynamic-values">
        <h3>Dynamic Calculated Values:</h3>
        <p id="dynamicNumber">-</p>
    </div>
```

```
      <div class="result" id="predictionResult">
        <h2>Predicted Price</h2>
        <p id="predictedPrice">-</p>
      </div>

    </div>

    <script>        document.getElementById('pricingForm').addEventListener('submit',
function(event) {            event.preventDefault();

          // Gather form data            const formData = new
FormData(document.getElementById('pricingForm'));

          // Convert FormData to JSON object
const jsonData = {};
formData.forEach((value, key) => {
jsonData[key] = value;
          });

          // Calculate dynamic number based on form input
const dynamicNumber = calculateDynamicNumber(jsonData);

          // Display dynamic number to user            const dynamicNumberElement =
document.getElementById('dynamicNumber');
dynamicNumberElement.textContent = Dynamic Number: ${dynamicNumber};

          // Add dynamic number to JSON data
jsonData['dynamic_number'] = dynamicNumber;

          // Send POST request to backend API
fetch('/predict', {              method: 'POST',
headers: {
              'Content-Type': 'application/json',
            },
            body: JSON.stringify(jsonData),
```

```
        })
        .then(response => {              if (!response.ok) {
throw new Error('Network response was not ok');
          }
          return response.json();
        })
        .then(data => {
          // Display predicted price              const resultContainer =
document.getElementById('predictionResult');              const
predictedPriceElement = document.getElementById('predictedPrice');
predictedPriceElement.textContent = $${data.predicted_price.toFixed(2)};
resultContainer.style.display = 'block';
        })
        .catch(error => {
console.error('Error:', error);
          // Handle error scenario (e.g., show error message to user)
        });
      });

      // Function to calculate dynamic number based on form data
function calculateDynamicNumber(formData) {
        // Example: Calculate a dynamic number based on form data
const accommodates = parseInt(formData['accommodates']);          const
cleaningFee = parseFloat(formData['cleaning_fee']);          const
numberOfReviews = parseInt(formData['number_of_reviews']);          const
bedrooms = parseInt(formData['bedrooms']);

        // Example calculation (replace with actual ML model integration)          const dynamicNumber =
accommodates * 10 + cleaningFee * 0.5 + numberOfReviews * 2 + bedrooms * 15;

        return dynamicNumber;
      }
  </script>
</body>
</html>
```

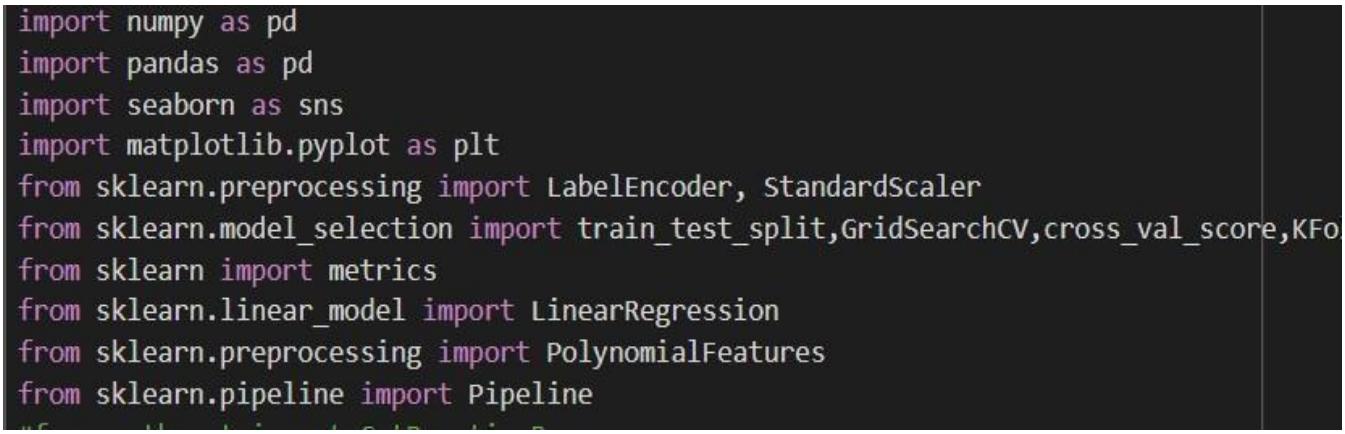
APP.PY

```
import pickle
from flask import Flask, render_template, request import
pandas as pd
import numpy as np

# Load the pickled model
model = pickle.load(open('airnbnb model.pkl','rb'))

# Create the Flask application
app = Flask(_name_)

# Define routes @app.route('/') def
index():     return
render_template('index.html')

@app.route('/contact.html') def
contact():     return
render_template('contact.html')

@app.route('/portfolio.html') def
portfolio():     return
render_template('portfolio.html')

@app.route('/pricing.html') def
pricing():     return
render_template('pricing.html')

@app.route('/predict.html', methods=['POST', 'GET'])
def predict():     return render_template('predict.html')
if request.method == 'POST':
        Hotelid = int(request.form['Hotelid'])
        Property_type = int(request.form['property_type'])
        Room_type = int(request.form['room_type'])
        Accommodates = int(request.form['accommodates'])
        Cancellation_policy = int(request.form['cancellation_policy'])
        Cleaning_fee = int(request.form['cleaning_fee'])  # Corrected variable name
```

```
    City = int(request.form['city'])
    Host_identity_verified = int(request.form['host_identity_verified'])
    Instant_bookable = int(request.form['instant_bookable'])
    Latitude = float(request.form['latitude'])
    Longitude = float(request.form['longitude'])
    Number_of_reviews = int(request.form['number_of_reviews'])
    Bedrooms = int(request.form['bedrooms'])

    # Make prediction
    prediction = model.predict(pd.DataFrame([[Hotelid, Property_type, Room_type, Accommodates,
Cancellation_policy,
                                        Cleaning_fee, City, Host_identity_verified, Instant_bookable,
Latitude, Longitude, Number_of_reviews, Bedrooms]]))        prediction = np.round(prediction, 4)


    return render_template('predict.html', prediction_text="is {}".format(prediction))
return render_template('predict.html')


if _name_ == '_main_':
app.run(debug=True)
```

CODE SNIPPETS

MODEL BUILDING



```
import numpy as pd
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split,GridSearchCV,cross_val_score,KFo
from sklearn import metrics
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import Pipeline
```

```
df=pd.read_csv('/content/train.csv')
df.head()
```

	id	log_price	property_type	room_type	amenities	accommodates	bathrooms	bed_type	cancellation_policy	cleaning_fee	...	latitude	longitude	name
0	6901257	5.010635	Apartment	Entire home/apt	{"Wireless Internet","Air conditioning",Kitche...	3.0	1.0	Real Bed	strict	True	...	40.696524	-73.991617	Beautiful brownstone 1-bedroom
1	6304928	5.129899	Apartment	Entire home/apt	{"Wireless Internet","Air conditioning",Kitche...	7.0	1.0	Real Bed	strict	True	...	40.766115	-73.989040	Superb 3BR Apt Located Near Times Square
2	7919400	4.976734	Apartment	Entire home/apt	{TV,"Cable TV","Wireless Internet","Air condit...	5.0	1.0	Real Bed	moderate	True	...	40.808110	-73.943756	The Garden Oasis
3	13418779	6.620073	House	Entire home/apt	{TV,"Cable TV",Internet,"Wireless Internet",Ki...	4.0	1.0	Real Bed	flexible	True	...	37.772004	-122.431619	Beautiful Flat in the Heart of SF!
4	3808709	4.744932	Apartment	Entire home/apt	{TV,Internet,"Wireless Internet","Air conditio...	2.0	1.0	Real Bed	moderate	True	...	38.925627	-77.034596	Great studio in midtown DC

```
df.shape
```

```
(1543, 29)
```

```
df=df.rename(columns={'id':'hotel_id'})
df
```

	hotel_id	log_price	property_type	room_type	amenities	accommodates	bathrooms	bed_type	cancellation_policy	cleaning_fee	.
0	6901257	5.010635	Apartment	Entire home/apt	{"Wireless Internet","Air conditioning",Kitche...	3.0	1.0	Real Bed	strict	True	
1	6304928	5.129899	Apartment	Entire home/apt	{"Wireless Internet","Air conditioning",Kitche...	7.0	1.0	Real Bed	strict	True	
2	7919400	4.976734	Apartment	Entire home/apt	{TV,"Cable TV","Wireless Internet","Air condit...	5.0	1.0	Real Bed	moderate	True	
3	13418779	6.620073	House	Entire home/apt	{TV,"Cable TV",Internet,"Wireless Internet",Ki...	4.0	1.0	Real Bed	flexible	True	
4	3808709	4.744932	Apartment	Entire home/apt	{TV,Internet,"Wireless Internet","Air conditio...	2.0	1.0	Real Bed	moderate	True	

```
df.tail()
```

	hotel_id	log_price	property_type	room_type	amenities	accommodates	bathrooms	bed_type	cancellation_policy	cleaning_fee	...	latitude	l
1538	18120849	4.382027	Apartment	Private room	{TV,Internet,"Wireless Internet","Air conditio...	2.0	1.0	Real Bed	flexible	False	...	37.770892	-12
1539	4720091	3.912023	Apartment	Private room	{Internet,"Wireless Internet",Kitchen,Heating,...	2.0	1.0	Futon	flexible	False	...	40.661779	-7
1540	15497573	4.442651	Apartment	Private room	{TV,"Wireless Internet","Air conditioning",Kit...	1.0	1.0	Real Bed	flexible	True	...	34.102151	-11
1541	3375448	4.787492	Guesthouse	Entire home/apt	{TV,Internet,"Wireless Internet","Air conditio...	2.0	1.0	Real Bed	moderate	True	...	34.030401	-11

```
df=df.drop(['amenities','host_has_profile_pic','first_review','last_review','host_response_rate','review_scores_rating','zipcode'],
df.head()
```

	hotel_id	log_price	property_type	room_type	accommodates	bathrooms	bed_type	cancellation_policy	cleaning_fee	city	...
0	6901257	5.010635	Apartment	Entire home/apt	3.0	1.0	Real Bed	strict	True	NYC	...
1	6304928	5.129899	Apartment	Entire home/apt	7.0	1.0	Real Bed	strict	True	NYC	...
2	7919400	4.976734	Apartment	Entire home/apt	5.0	1.0	Real Bed	moderate	True	NYC	...
3	13418779	6.620073	House	Entire home/apt	4.0	1.0	Real Bed	flexible	True	SF	...
4	3808709	4.744932	Apartment	Entire home/apt	2.0	1.0	Real Bed	moderate	True	DC	...

5 rows × 22 columns

```
df.isnull().sum()
```

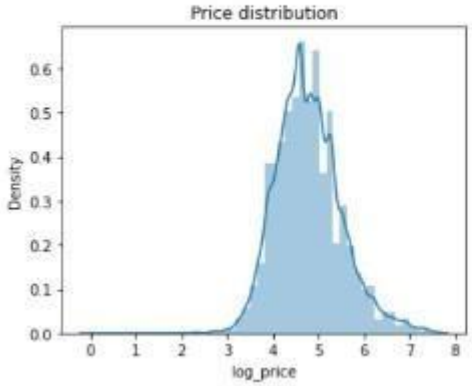
```
hotel_id                  0
log_price                 0
property_type             0
room_type                 0
accommodates              1
bathrooms                 6
bed_type                  1
cancellation_policy       1
cleaning_fee              1
city                      1
description               1
host_identity_verified    7
host_since                7
instant_bookable          1
latitude                  1
longitude                 1
name                      1
neighbourhood           126
number_of_reviews         1
bedrooms                  3
beds                      3
dtype: int64
```

```
df.host_identity_verified.fillna(method='ffill', inplace=True)
df.neighbourhood.fillna(method='ffill', inplace=True)
df.bathrooms.fillna(method='ffill', inplace=True)
df.beds.fillna(method='ffill', inplace=True)
df['bedrooms'].fillna(df['bathrooms'].median(), inplace=True)
```

```
df.describe()
```

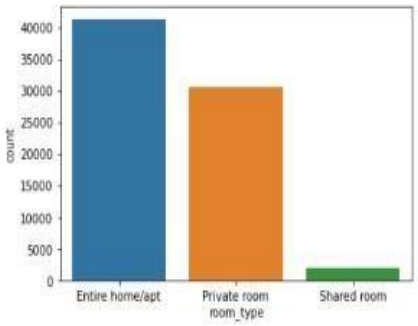
	hotel_id	log_price	accommodates	bathrooms	latitude	longitude	number_of_reviews	bedrooms	beds
count	1.543000e+03	1543.000000	1542.000000	1543.000000	1542.000000	1542.000000	1542.000000	1543.000000	1543.000000
mean	1.126008e+07	4.790316	3.114137	1.208360	38.503836	-92.037717	19.418936	1.237848	1.683085
std	6.215381e+06	0.717216	2.121842	0.501465	3.036840	21.767771	35.635651	0.817175	1.222367
min	1.206800e+04	2.484907	1.000000	0.000000	33.722757	-122.508521	0.000000	0.000000	1.000000
25%	6.016475e+06	4.317488	2.000000	1.000000	34.151089	-118.350804	1.000000	1.000000	1.000000
50%	1.238868e+07	4.744932	2.000000	1.000000	40.662595	-74.103087	5.000000	1.000000	1.000000
75%	1.657608e+07	5.192957	4.000000	1.000000	40.740870	-73.953838	21.750000	1.000000	2.000000
max	2.118810e+07	7.569412	16.000000	5.000000	42.382883	-70.991861	318.000000	7.000000	16.000000

```
plt.figure(figsize = (5, 4))
sns.distplot(df["log_price"])
plt.title('Price distribution')
plt.show()
```



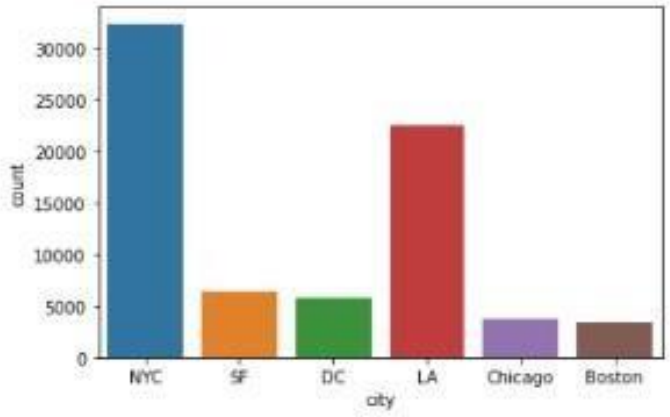
```
sns.countplot(x='room_type', data=df)
```

```
<AxesSubplot: xlabel='room_type', ylabel='count'>
```

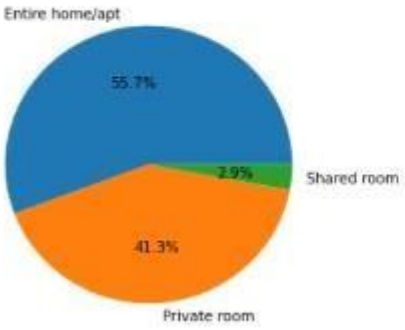



```
sns.countplot(x='city', data=df)
```

```
<AxesSubplot: xlabel='city', ylabel='count'>
```

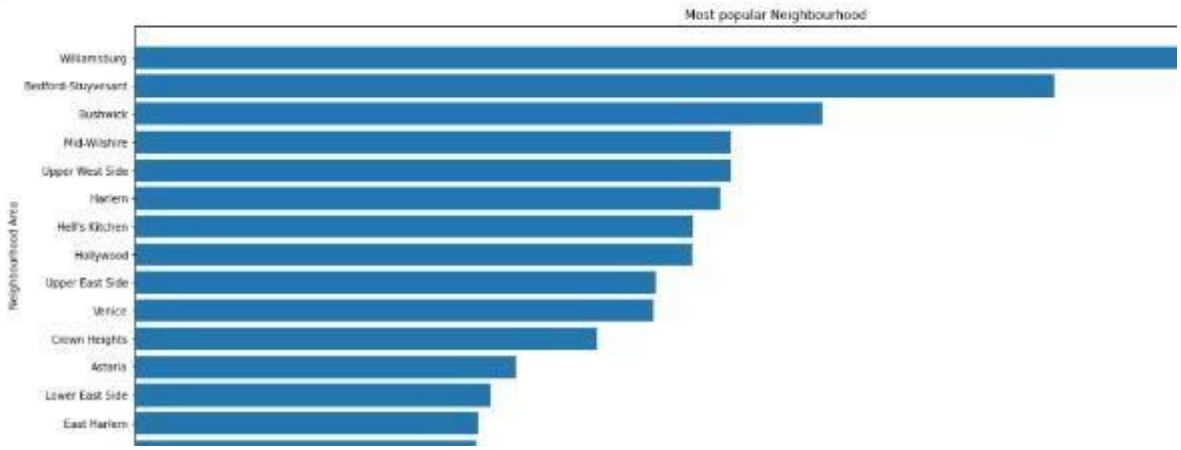



```
room_type_counts = df['room_type'].value_counts()

plt.pie(room_type_counts, labels=room_type_counts.index, autopct='%1.1f%%')
plt.axis('equal')
plt.show()
```



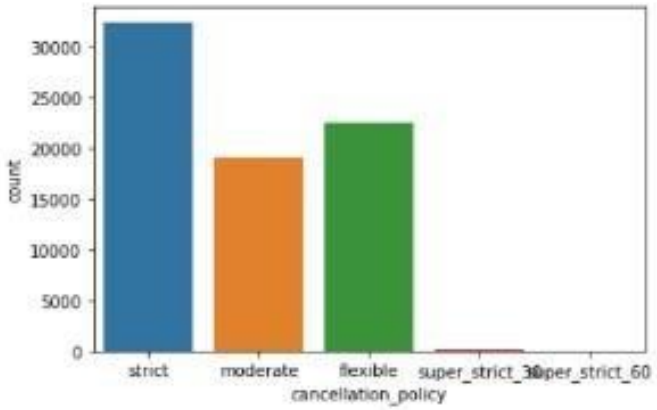
```
fig = plt.figure(figsize=(3,3))
ax = fig.add_axes([0,0,1,1])
ax.axis('equal')
langs = list(df.city.unique())
students =list(df.city.value_counts())
ax.pie(students, labels = langs,autopct='%1.2f%%')
plt.show()
```


```
data = df.neighbourhood.value_counts()[:15]
plt.figure(figsize=(22,8))
x = list(data.index)
y = list(data.values)
x.reverse()
y.reverse()

plt.title("Most popular Neighbourhood")
plt.ylabel("Neighbourhood Area")
plt.xlabel("Number of guest who host in this area")

plt.barh(x,y)
plt.show()
```



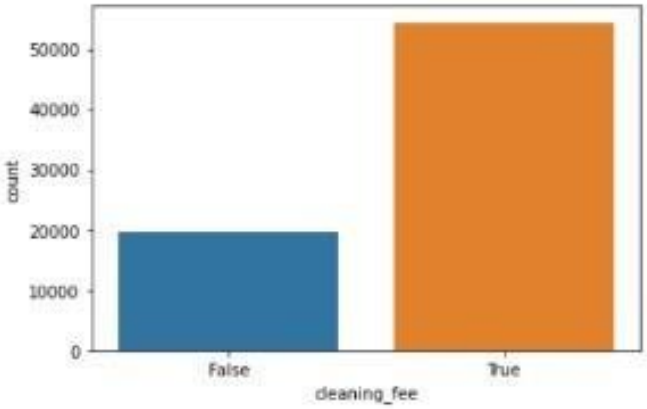
```
sns.countplot(x='cancellation_policy', data=df)
```

```
<AxesSubplot: xlabel='cancellation_policy', ylabel='count'>
```



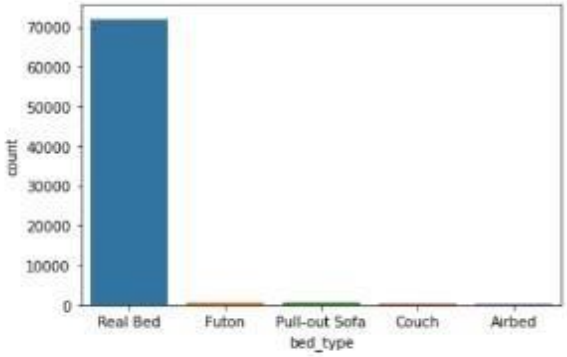

```
sns.countplot(x='cleaning_fee', data=df)
```

```
<AxesSubplot: xlabel='cleaning_fee', ylabel='count'>
```



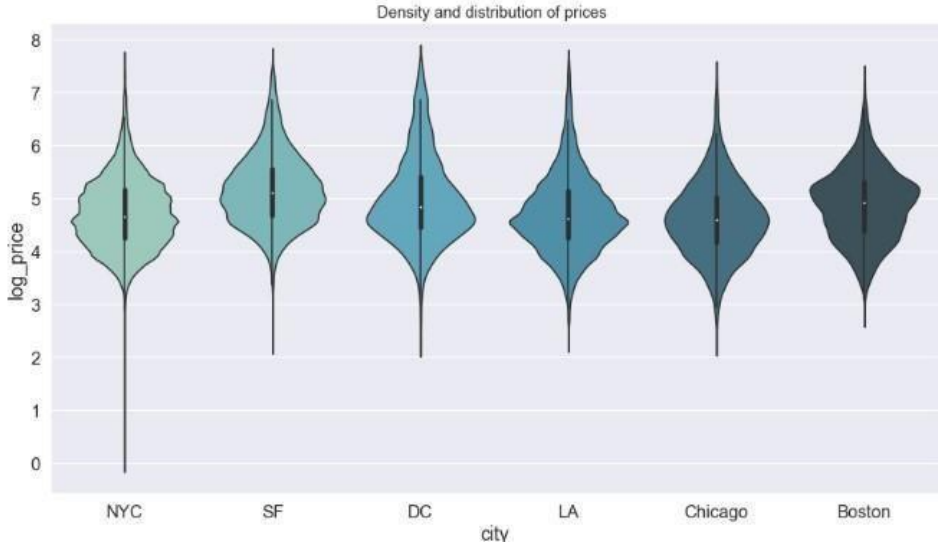

```
sns.countplot(x='bed_type', data=df)
```

```
<AxesSubplot: xlabel='bed_type', ylabel='count'>
```

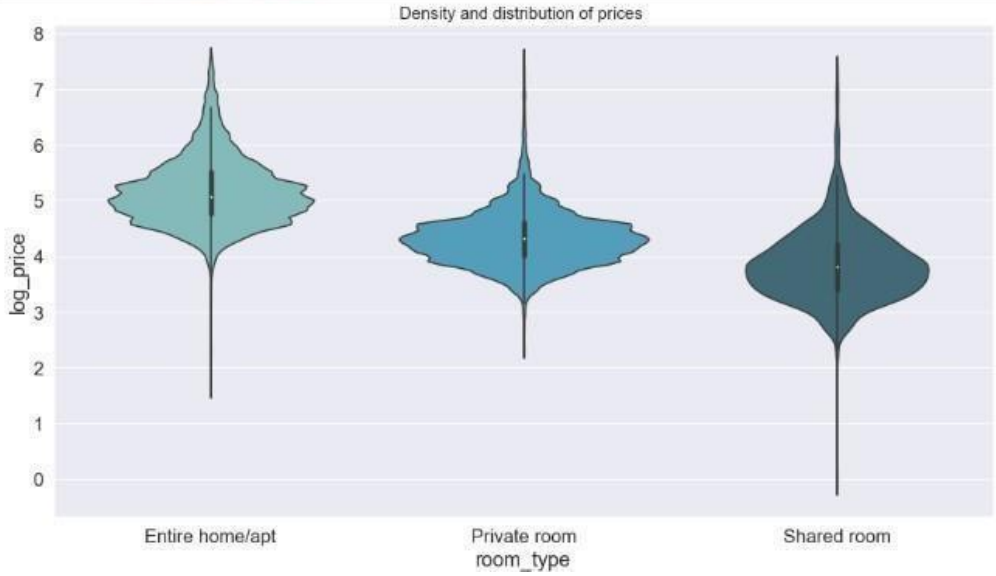


```
def plot_violinplot(h,v):
    plt.figure(figsize=(15,8))
    sns.set(font_scale=1.5)
    sns.violinplot(data=df, x=h, y=v, palette='GnBu_d')
    plt.title('Density and distribution of prices ', fontsize=15)
    plt.xlabel(h)
    plt.ylabel(v)
```

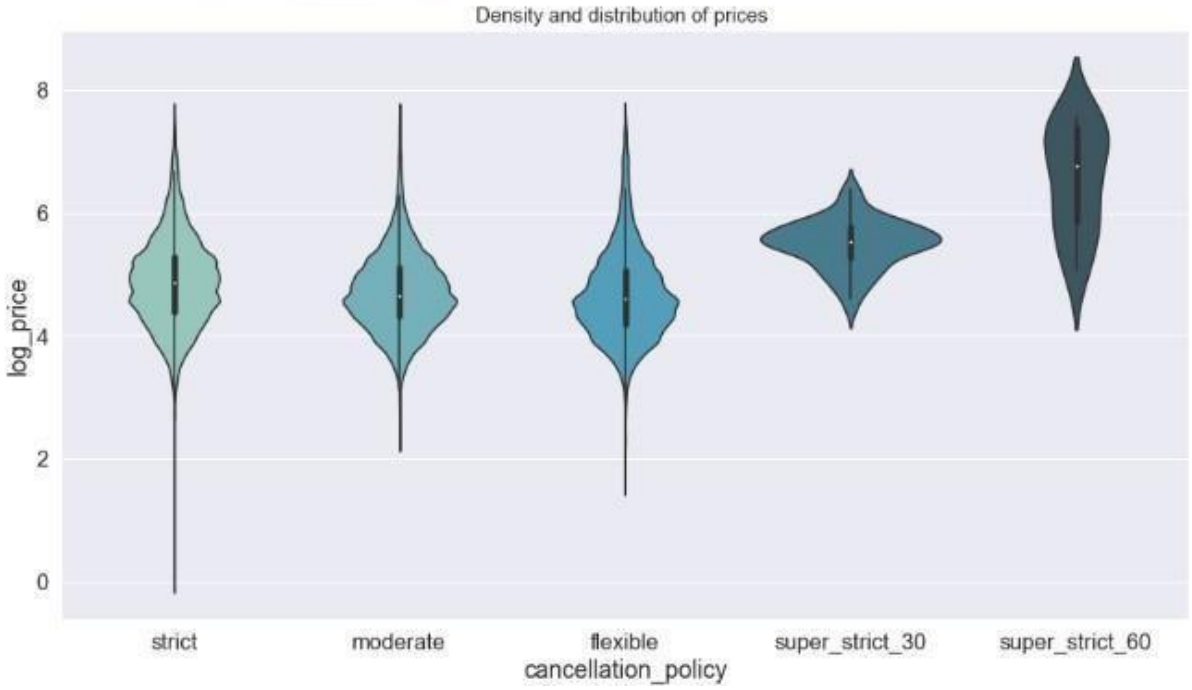
```
plot_violinplot("city","log_price")
```



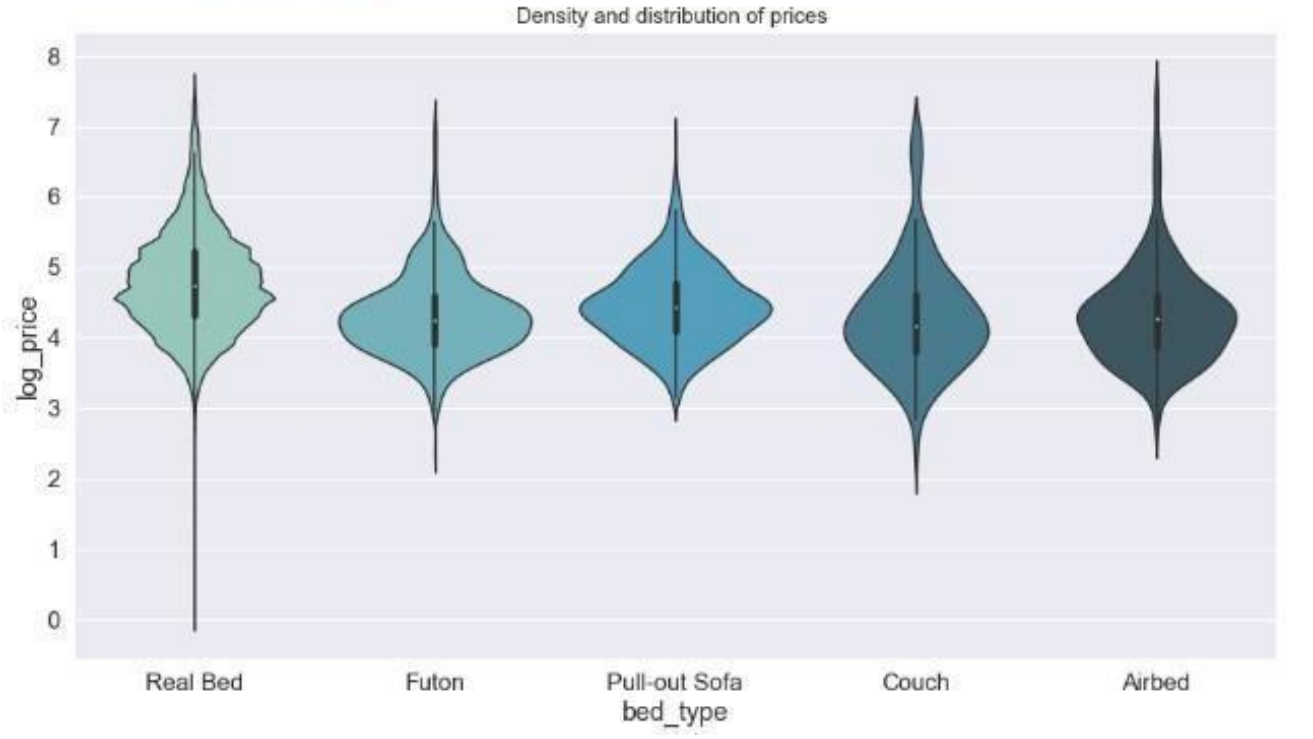

```
plot_violinplot("room_type","log_price")
```



```
plot_violinplot("cancellation_policy","log_price")
```

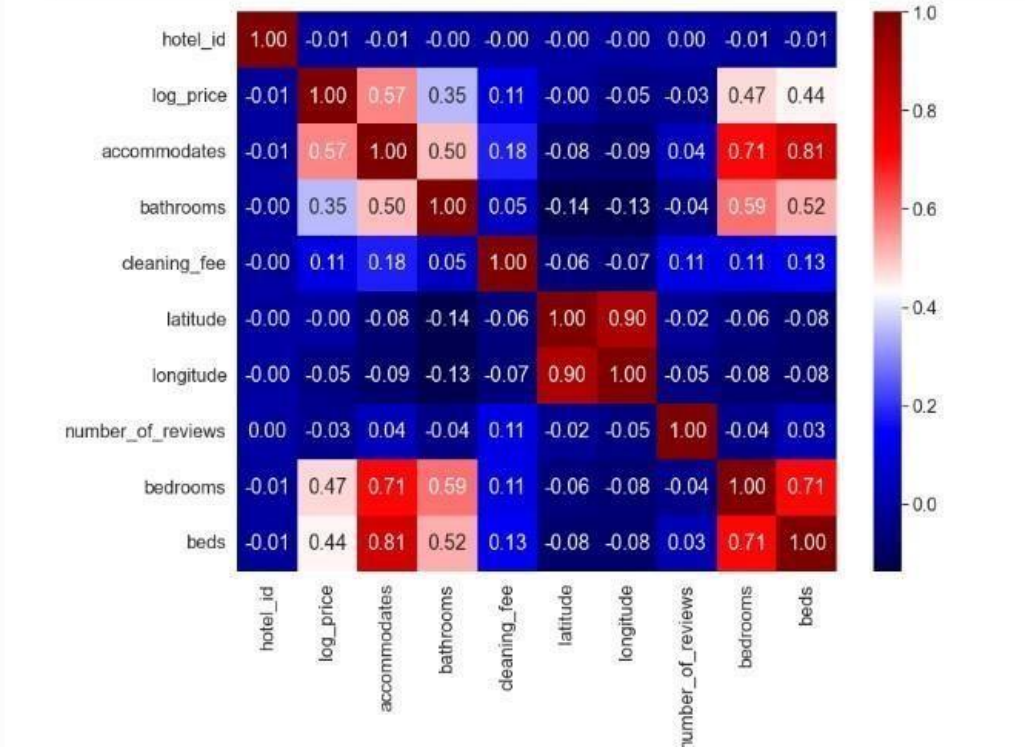



```
plot_violinplot("bed_type","log_price")
```



HEAT MAP

```
plt.figure(figsize = (12,9))
sns.heatmap(df.corr(), annot=True, fmt=".2f", cmap="seismic")
plt.show()
```




```
categorical_col = []
for column in df.columns:
   if df[column].dtypes !=  "float64" and df[column].dtypes != "int64":
     categorical_col.append(column)
```

```
categorical_col
```

```
['property_type',
 'room_type',
 'bed_type',
 'cancellation_policy',
 'cleaning_fee',
 'city',
 'description',
 'host_identity_verified',
 'host_since',
 'instant_bookable',
 'name',
 'neighbourhood']
```

```
from sklearn.preprocessing import LabelEncoder
le= LabelEncoder()
```

```
print('Before encoding:', df['property_type'].unique())
df['property_type']= le.fit_transform(df['property_type'])
print('After encoding: ', df['property_type'].unique())
```

```
Before encoding: ['Apartment' 'House' 'Condominium' 'Loft' 'Townhouse' 'Hostel'
 'Guest suite' 'Bed & Breakfast' 'Bungalow' 'Guesthouse' 'Dorm' 'Other'
 'Camper/RV' 'Villa' 'Boutique hotel' 'Timeshare']
After encoding:  [ 0 10  5 11 14  9  7  1  3  8  6 12  4 15  2 13]
```

```
print('Before encoding:', df['cancellation_policy'].unique())
df['cancellation_policy'] =le.fit_transform(df['cancellation_policy'])
print('After encoding:', df ['cancellation_policy'].unique())
```

```
Before encoding: ['strict' 'moderate' 'flexible' 'super_strict_30' 'super_strict_60' nan]
After encoding: [2 1 0 3 4 5]
```

```
print('Before encoding:', df['cleaning_fee'].unique())
df['cleaning_fee'] =le.fit_transform(df['cleaning_fee'])
print('After encoding:', df ['cleaning_fee'].unique())
```

```
Before encoding: [True False nan]
After encoding: [1 0 2]
```

```
print('Before encoding:', df['city'].unique())
df['city'] =le.fit_transform(df['city'])
print('After encoding:', df ['city'].unique())
```

```
Before encoding: ['NYC' 'SF' 'DC' 'LA' 'Chicago' 'Boston' nan]
After encoding: [4 5 2 3 1 0 6]
```

```
print('Before encoding:', df['host_identity_verified'].unique())
df['host_identity_verified'] = le.fit_transform(df['host_identity_verified'])
print('After encoding:', df['host_identity_verified'].unique())
```

```
Before encoding: ['t' 'f']
After encoding: [1 0]
```

```
df.host_identity_verified.fillna(method='ffill', inplace=True)
df.neighbourhood.fillna(method='ffill', inplace=True)
df.bathrooms.fillna(method='ffill', inplace=True)
df.beds.fillna(method='ffill', inplace=True)
df['bedrooms'].fillna(df['bathrooms'].median(), inplace=True)
```

```
df.isnull().sum()
```

```
hotel_id                0
log_price               0
property_type           0
room_type               0
accommodates            1
bathrooms               0
bed_type                1
cancellation_policy     0
cleaning_fee            0
city                    0
description             1
host_identity_verified  0
host_since              7
instant_bookable        1
latitude                1
longitude               1
name                    1
neighbourhood           0
number_of_reviews       1
bedrooms                0
beds                    0
dtype: int64
```

```
x=df.iloc[:,[0,2,3,4,7,8,9,11,13,14,15,18,19]]
x.head(20)
```

	hotel_id	property_type	room_type	accommodates	cancellation_policy	cleaning_fee	city	host_identity_verified	instant_bookable	latitude
0	6901257	0	0	3.0	2	1	4	1	f	40.696524
1	6304928	0	0	7.0	2	1	4	0	t	40.766115
2	7919400	0	0	5.0	1	1	4	1	t	40.808110
3	13418779	10	0	4.0	0	1	5	1	f	37.772004
4	3808709	0	0	2.0	1	1	2	1	t	38.925627
5	12422935	0	1	2.0	2	1	5	1	t	37.753164
6	11825529	0	0	3.0	1	1	3	0	t	33.980454
7	13971273	5	0	2.0	1	1	3	1	f	34.046737
8	180792	10	1	2.0	1	1	5	0	f	37.781128
9	5385260	10	1	2.0	1	1	3	0	t	33.992563
10	5578513	0	1	2.0	2	1	4	1	f	40.723883
11	17423675	10	0	4.0	2	1	3	1	f	33.875862
12	14066228	0	1	2.0	0	1	3	0	f	33.813228
13	2658946	0	0	6.0	2	1	2	1	t	38.919630
14	583490	0	0	2.0	2	1	3	1	f	33.778526

```
y=df.iloc[:,1]
y.head(20)
```

```
0     5.010635
1     5.129899
2     4.976734
3     6.620073
4     4.744932
5     4.442651
6     4.418841
7     4.787492
8     4.787492
9     3.583519
```

```
from sklearn.model_selection import train_test_split
x=df.iloc[:,[0,2,3,4,7,8,9,11,13,14,15,18,19]]
y=df.iloc[:,1]
x_train,X_test,y_train,y_test=train_test_split(x,y,test_size=0.25,random_state=101)
print(x_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(1157, 13)
(386, 13)
```

```
from sklearn.preprocessing import StandardScaler
```

```
scaler = StandardScaler()
x_train_scaled = scaler.fit_transform(x_train)
x_test_scaled = scaler.transform(x_test)
```

1.LinearRegression

```
# Initialize the Linear Regression model
lr = LinearRegression()

# Fit the model to the training data
lr.fit(x_train_scaled, y_train)

# Predict using the test data
y_pred_lr = lr.predict(x_test_scaled)

# Calculate evaluation metrics
mae_lr = metrics.mean_absolute_error(y_test, y_pred_lr)
mse_lr = metrics.mean_squared_error(y_test, y_pred_lr)
rmse_lr = np.sqrt(metrics.mean_squared_error(y_test, y_pred_lr))
r2_lr = metrics.r2_score(y_test, y_pred_lr)

# Print evaluation metrics
print('\nMean Absolute Error of Linear Regression:', mae_lr)
print('\nMean Squared Error of Linear Regression:', mse_lr)
print('\nRoot Mean Squared Error of Linear Regression:', rmse_lr)
print('\nR2 Score of Linear Regression:', r2_lr)
```

```
Mean Absolute Error of Linear Regression: 0.24874665538693447

Mean Squared Error of Linear Regression: 0.08264846267447648

Root Mean Squared Error of Linear Regression: 0.2874864565061744

R2 Score of Linear Regression: -0.0006869779839682177
```

2.RandomForestRegression

```
rf=RandomForestRegressor()
rf.fit(x_train_scaled,y_train)
y_pred_rf= rf.predict(x_test_scaled)
mae_rf= metrics.mean_absolute_error(y_test, y_pred_rf)
mse_rf= metrics.mean_squared_error(y_test, y_pred_rf)
rmse_rf =np.sqrt(metrics.mean_squared_error(y_test, y_pred_rf))
r2_rf= metrics.r2_score(y_test, y_pred_rf)
print('\nMean Absolute Error of Random Forest Regressor:',mae_rf)
print('\nMean Squarred Error of Random Forest Regressor :', mse_rf)
print('\nRoot Mean Squarred Error of Random Forest Regressor:', rmse_rf)
print('\nR2 Score of Random Forest Regressor :',r2_rf)
```

```
Mean Absolute Error of Random Forest Regressor: 0.2498636525834891

Mean Squarred Error of Random Forest Regressor : 0.08386865392715212

Root Mean Squarred Error of Random Forest Regressor: 0.28960085277352365

R2 Score of Random Forest Regressor : -0.015460749421335018
```

3.Polynomial Regression

```
from sklearn.linear_model import Ridge
model=Pipeline([
('poly', PolynomialFeatures()),
('ridge', Ridge(fit_intercept=True))
])
param_grid ={
'poly__degree': [1, 2, 3],
'ridge__alpha': [0.1,0.5,1.0,2.0]
}
#Perform grid search with 5-fold cross-validation
poly_tuned=GridSearchCV(model, param_grid, cv=5)
#Traning and Testing
poly_tuned.fit(x_train_scaled, y_train)
y_pred_poly= poly_tuned.predict(x_test_scaled)
mae_poly= metrics.mean_absolute_error(y_test, y_pred_poly)
mse_poly= metrics.mean_squared_error(y_test, y_pred_poly)
rmse_poly= np.sqrt(metrics.mean_squared_error(y_test, y_pred_poly))
r2_poly= metrics.r2_score(y_test, y_pred_poly)
print('\nMean Absolute Error of Polynomial Regression:',mae_poly)
print('\nMean Squarred Error of Polynomial Regression:',mse_poly)
print('\nRoot Mean Squarred Error of Polynomial Regression:',rmse_poly)
print('\nR2 Score of Polynomial Regression :',r2_poly)
```

```
Mean Absolute Error of Polynomial Regression: 0.24874665259347883

Mean Squarred Error of Polynomial Regression: 0.08264845965849907

Root Mean Squarred Error of Polynomial Regression: 0.2874864512607491

R2 Score of Polynomial Regression : -0.0006869414672656671
```

4.Gradient Boosting

```
gb = GradientBoostingRegressor(n_estimators=200, learning_rate=0.2, max_depth=10)

# Fit the model
gb.fit(x_train_scaled, y_train)

# Predict using the model
y_pred_gb = gb.predict(x_test_scaled)

# Calculate evaluation metrics
mae_gb = metrics.mean_absolute_error(y_test, y_pred_gb)
mse_gb = metrics.mean_squared_error(y_test, y_pred_gb)  # Corrected variable name
rmse_gb = np.sqrt(metrics.mean_squared_error(y_test, y_pred_gb))
r2_gb = metrics.r2_score(y_test, y_pred_gb)

# Print the results
print('\nMean Absolute Error of Gradient Boosting:', mae_gb)
print('\nMean Squared Error of Gradient Boosting:', mse_gb)  # Corrected print statement
print('\nRoot Mean Squared Error of Gradient Boosting:', rmse_gb)
print('\nR2 Score of Gradient Boosting:', r2_gb)
```

```
Mean Absolute Error of Gradient Boosting: 0.2561680197799005

Mean Squared Error of Gradient Boosting: 0.09002655771730858

Root Mean Squared Error of Gradient Boosting: 0.3000442595973277

R2 Score of Gradient Boosting: -0.09371798203234927
```

5.catboost

```
from catboost import CatBoostRegressor
from sklearn.model_selection import cross_val_score, KFold
from sklearn import metrics
import numpy as np
model_CBR = CatBoostRegressor()
model_CBR.fit(x_train_scaled, y_train)
cross_val_score(model_CBR,x_train_scaled, y_train,
scoring='r2',
    cv=KFold(n_splits=5,
    shuffle=True,
    random_state=2022,
))
y_pred_cbr= model_CBR.predict(x_test_scaled)
mae_cbr =metrics.mean_absolute_error(y_test, y_pred_cbr)
mse_cbr =metrics.mean_squared_error(y_test, y_pred_cbr)
rmse_cbr = np.sqrt(metrics.mean_squared_error(y_test, y_pred_cbr))
r2_cbr =metrics.r2_score(y_test, y_pred_cbr)
print('\nMean Absolute Error of CatBoost Regressor :',mae_cbr)
print('\nMean Squarred Error of CatBoost Regressor:',mse_cbr)
print('\nRoot Mean Squarred Error of CatBoost Regressor:',rmse_cbr)
print('\nR2 Score of CatBoost Regressor:',r2_cbr)
```

```
streaming output truncated to the last 5000 lines.
12:     learn: 0.2872384        total: 231ms    remaining: 17.5s
13:     learn: 0.2871869        total: 257ms    remaining: 18.1s
14:     learn: 0.2871500        total: 277ms    remaining: 18.2s
15:     learn: 0.2871268        total: 303ms    remaining: 18.6s
16:     learn: 0.2870934        total: 330ms    remaining: 19.1s
17:     learn: 0.2870527        total: 353ms    remaining: 19.3s
18:     learn: 0.2870221        total: 367ms    remaining: 19s
```

6.XGBoost

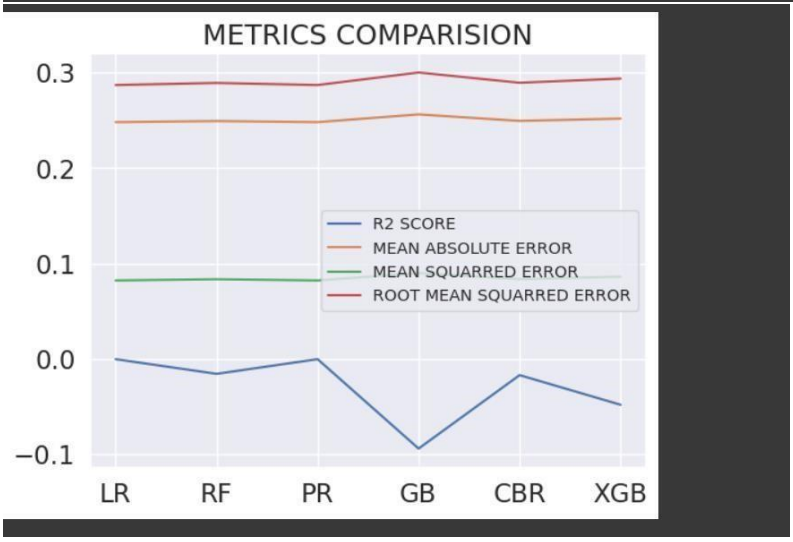
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn import metrics
from xgboost import XGBRegressor
xgb = XGBRegressor(objective='reg:squarederror')
xgb.fit(x_train_scaled, y_train)
y_pred_xgb = xgb.predict(x_test_scaled)
mae_xgb = metrics.mean_absolute_error(y_test, y_pred_xgb)
mse_xgb = metrics.mean_squared_error(y_test, y_pred_xgb)
rmse_xgb = np.sqrt(mse_xgb)
r2_xgb = metrics.r2_score(y_test, y_pred_xgb)
print('\nMean Absolute Error of XGBoost Regressor:', mae_xgb)
print('\nMean Squared Error of XGBoost Regressor:', mse_xgb)
print('\nRoot Mean Squared Error of XGBoost Regressor:', rmse_xgb)
print('\nR2 Score of XGBoost Regressor:', r2_xgb)
```

```
Mean Absolute Error of XGBoost Regressor: 0.2517812900427125

Mean Squared Error of XGBoost Regressor: 0.08625071704707236

Root Mean Squared Error of XGBoost Regressor: 0.29368472389123745

R2 Score of XGBoost Regressor: -0.04784590891261442
```

```
r2_list={"Linear Regression": r2_lr,
"Random Forest": r2_rf,
"Polynomial Regression":r2_poly,
"Gradient Boosting":r2_gb, "CatBoost":r2_cbr,"XGBoost": r2_xgb}
mae_list={"Linear Regression": mae_lr,
"Random Forest": mae_rf, "Polynomial Regression": mae_poly,
"Gradient Boosting":mae_gb ,"CatBoost": mae_cbr,
"XGBoost" :mae_xgb}
mse_list = {"Linear Regression": mse_lr,
"Random Forest": mse_rf ,"Polynomial Regression":mse_poly,
"Gradient Boosting":mse_gb,
"CatBoost": mse_cbr,
"XGBoost":mse_xgb}
rmse_list ={"Linear Regression":rmse_lr,
"Random Forest": rmse_rf ,"Polynomial Regression":rmse_poly,
"Gradient Boosting":rmse_gb,
"CatBoost": rmse_cbr, "XGBoost":rmse_xgb}
a1 =pd.DataFrame.from_dict(r2_list, orient='index',columns=["R2_SCORE"])
a2= pd.DataFrame.from_dict(mae_list, orient ='index', columns  = ["MEAN ABSOLUTE ERROR"])
a3 =pd.DataFrame.from_dict(mse_list, orient ='index', columns = ["MEAN SQUARRED ERROR"])
a4 =pd.DataFrame.from_dict(rmse_list, orient ='index', columns = ["ROOT MEAN SQUARRED ERROR"])
org=pd.concat([a1,a2,a3,a4],axis=1)
org
```

	R2_SCORE	MEAN ABSOLUTE ERROR	MEAN SQUARRED ERROR	ROOT MEAN SQUARRED ERROR
Linear Regression	-0.000100	0.248069	0.082321	0.286916
Random Forest	-0.015474	0.249214	0.083586	0.289113
Polynomial Regression	-0.000100	0.248069	0.082321	0.286916
Gradient Boosting	-0.093718	0.256168	0.090027	0.300044
CatBoost	-0.016833	0.249363	0.083698	0.289306
XGBoost	-0.047846	0.251781	0.086251	0.293685

```
alg= ['LR', 'RF', 'PR', 'GB', 'CBR', 'XGB']
plt.plot(alg,a1)
plt.plot(alg,a2)
plt.plot(alg,a3)
plt.plot(alg,a4)
legend =["R2 SCORE", "MEAN ABSOLUTE ERROR", "MEAN SQUARRED ERROR", "ROOT MEAN SQUARRED ERROR"]
plt.title("METRICS COMPARISION")
plt.legend(legend, loc ='right', fontsize='xx-small')
plt.show()
```



```
import pickle
# Use r prefix to create a raw string
pickle.dump(model_CBR, open(r'C:\Users\akula\OneDrive\Desktop\cbr.pkl', 'wb'))
```

```
import pickle
import numpy as np  # Import numpy for array manipulation (if not already imported)

# Load the model from the pickle file
file_path = r'C:\Users\akula\OneDrive\Desktop\cbr.pkl'
model = pickle.load(open(file_path, 'rb'))  # Use 'rb' for read binary mode

# Example prediction using the loaded model
input_data = np.array([[180792, 17, 1, 2, 1, 1, 5, 0, 0, 37.781128, -122.501095, 159, 1.0]])  # Convert to numpy array if not already
prediction = model.predict(input_data)

# Print the prediction
print(prediction)
```

```
[0.60562949]
```

Run The Web Application

```
[Running] python -u "c:\Users\Admin\Desktop\SmartBridge\ML project\Airbnb price prediction\flask\app.py"
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server inste
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
```