airbnb

Guest Satisfaction Prediction

TEAM SC_3

نوران احمد سمير احمد	2022170472
محمد خالد جمال عرابي	2022170365
رانا ناصر محمد جودة	2022170145
عبدالرحمن احمد طاهر	2022170227
سلمي عماد احمد علي	2022170189
شهد سامح عبد العزيز	2022170206

Introduction

"Our Guest Satisfaction prediction project advanced machine learning techniques to analyze and predict guest satisfaction levels for Airbnb stays. By examining features such as property characteristics, host behavior, and guest feedback, our model delivers accurate and insightful predictions. Focused on enhancing the guest experience and supporting hosts in improving their services, our work empowers the Airbnb community with data-driven insights. Join us in our mission to create more enjoyable, personalized, and memorable travel experiences for guests around the world."

Objective

Build a regression model to accurately predict acquisition prices
Identify key predictors, including founder/board member attributes Compare multiple regression algorithms and their performance

Scope

The project implements supervised regression on datasets, focusing on preprocessing, feature engineering, and model comparison. The algorithms include Linear Regression, Polynomial Regression, Random Forest, XGBoost, and SVR, lasso , ridge, DT , RF ,Elastic, Hyper, Gradiant Boosting, Log tranform

1. Preprocessing Techniques

The provided dataset contained various types of features, including numerical, categorical, and textual data. To ensure consistency and prepare the data for modeling, the following preprocessing steps were performed:

1.1 Handling Missing Values and Duplicates

- **Numerical Features:** Missing values in numerical features such as reviews_per_month were imputed using the KNN imputer
- **Categorical Features:** Features such as host_response_rate and host_is_superhost had missing values filled with the mode (most frequent value).
- For features with too many missing values (>70% missing), such as thumbnail_url, we **dropped** the column because imputing would have introduced too much noise.
- Duplicates dropped
- Drop host_listings_count after calculating the match between it and host_total_listings_count, and it was above 99%

1.2 Encoding Categorical Variables

- **Technique:**
    - **One-Hot Encoding** was applied to nominal categories (e.g., bed_type).
    - **Label Encoding** was used for ordinal features if ordering made sense (e.g., amenities).
    - **Target Encoding**
    -

1.3 Outlier Detection and Removal

- **Technique:**

Outliers in fields like price, number_of_reviews were detected via boxplots and removed using the IQR method.

2. Feature Analysis

We explored feature relationships using correlation matrices, scatter plots, and pair plots.

Key Observations:

- **Overall, most features show a very weak correlation with review_scores_rating.**
- No feature has a moderate (>|0.3|) or strong correlation.
- number_of_reviews and number_of_stays show a small positive correlation (~0.066) with review_scores_rating.
- **Price-related features** (nightly_price, price_per_stay) have **extremely weak** positive correlation (~0.01 to 0.02).
- **Property size features** (like bedrooms, bathrooms, accommodates, beds) show strong mutual correlations (0.6-0.8) but have **very weak** direct **correlation** to review_scores_rating.
- **Location features** (latitude, longitude) have **negligible correlation** with the target.

3. Regression Techniques Used

We implemented 15 very different regression models to predict guest satisfaction:

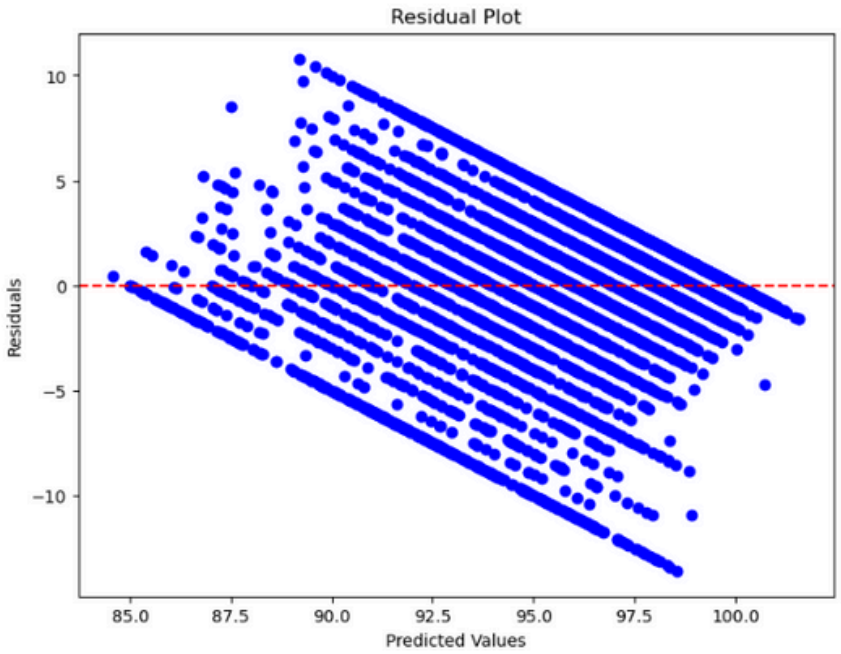
3.1) Linear Regression

Linear regression is a foundational machine learning algorithm used to predict a target variable based on the linear relationship between input features and the target. The model assumes that the relationship between the independent variables (features) and the dependent variable (target) is linear.

Key Metrics:

- **R^2 (R-squared)**: 0.4292 – Represents the proportion of the variance in the target variable that is explained by the model. The R^2 score indicates that about 42.92% of the variance is explained by the model.
- **MAE (Mean Absolute Error)**: 2.4394 – Indicates the average absolute difference between the predicted and actual values.
- **MSE (Mean Squared Error):** 11.7083 – Measures the average squared difference between predicted and actual values.
- **RMSE (Root Mean Squared Error):** 3.4217 – The square root of MSE, giving a sense of the magnitude of errors in the same units as the target variable.

.



3.2) Huber Regression

Huber regression is a robust regression model that combines the properties of linear regression and absolute error minimization, making it less sensitive to outliers. It uses a combination of squared error for smaller residuals and absolute error for larger residuals.

Key Metrics:

- **R^2**: 0.3934 – A slightly lower R^2 than linear regression, suggesting the model explains less variance in the target variable.
- **MAE:** 2.3699 – Slightly better MAE compared to linear regression, indicating less deviation between predicted and actual values.
- **RMSE**: 3.4924 – Higher RMSE compared to Linear Regression, suggesting that despite being robust to outliers, the model's overall error is higher.

.

3.3) ElasticNet Regression

ElasticNet is a linear regression model that combines the penalties of both **Lasso (L1 regularization)** and **Ridge (L2 regularization)**. It is particularly useful when there are many correlated features, as it can both shrink and select features.

Key Metrics:

- **R^2**: 0.4183 – Slightly better than both Linear and Huber regression models, indicating that the model can explain about 41.83% of the variance in the data.
- **MAE**: 2.4361 – Very similar to Linear Regression's MAE
- **RMSE**: 3.4206 – Close to the Linear Regression's RMSE

3.4) Ridge Regression

Ridge regression is a type of **linear regression** that applies **L2 regularization** (penalty on the size of coefficients), which helps to prevent overfitting by shrinking large coefficients. This model is useful when the dataset has many features.

Key Metrics:

- **R^2**: 0.4187 – Very similar to ElasticNet and slightly better than Linear Regression.
- **MAE:** 2.4380 – Similar to Linear and ElasticNet models.
- **RMSE:** 3.4187 – Matches the RMSE of ElasticNet, indicating comparable predictive performance.

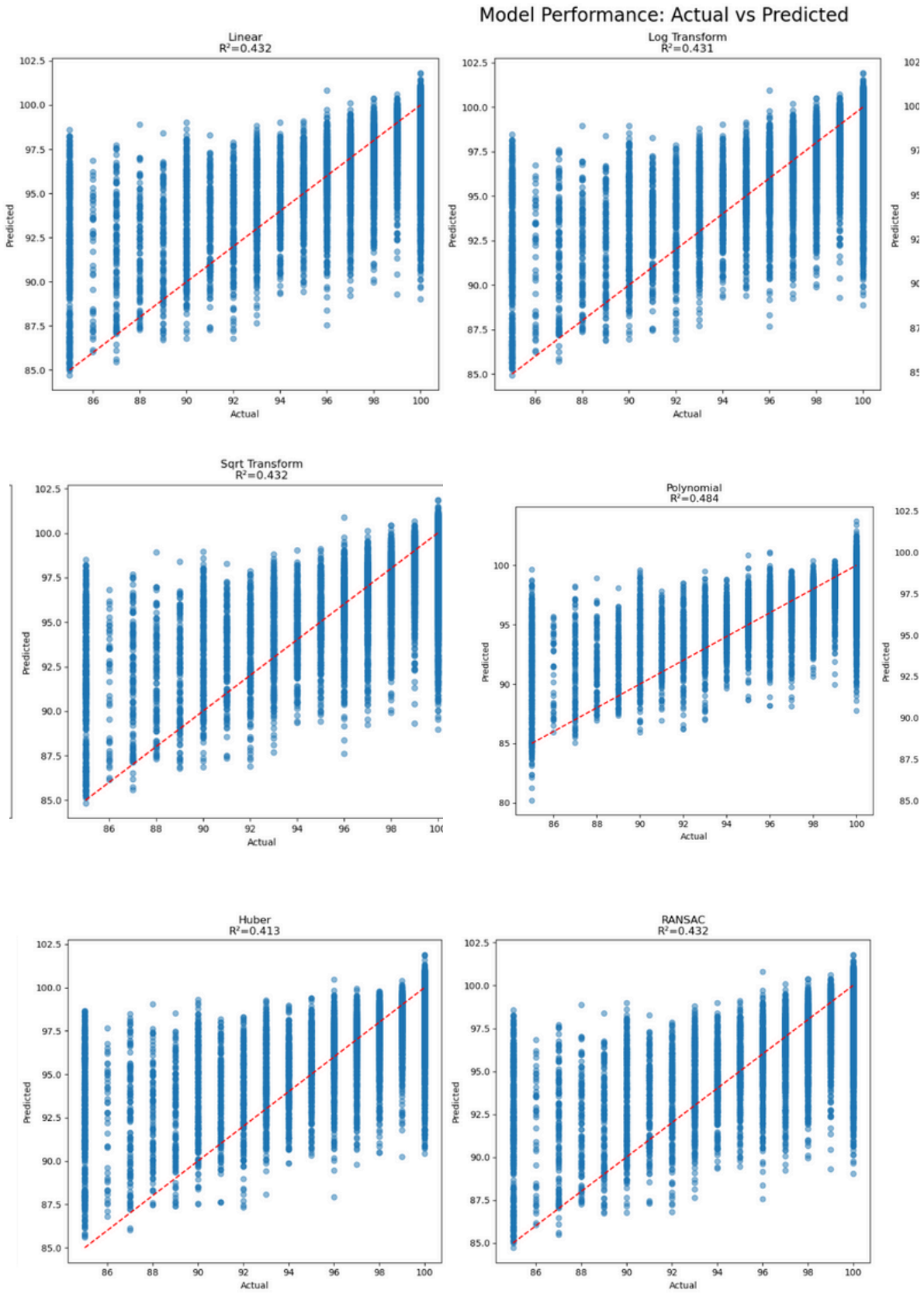
3.5) Lasso Regression

Lasso regression is another form of **regularized linear regression** that applies **L1 regularization**, which can drive some coefficients to zero. This makes Lasso particularly useful for feature selection.

Key Metrics:

- **R^2**: 0.4154 – Similar to Ridge and ElasticNet, with a small decrease from Linear Regression.
- **MAE:** 2.4450 – Slightly worse than Ridge and ElasticNet.
- **RMSE**: 3.4284 – Slightly higher than both Ridge and ElasticNet.

Plots

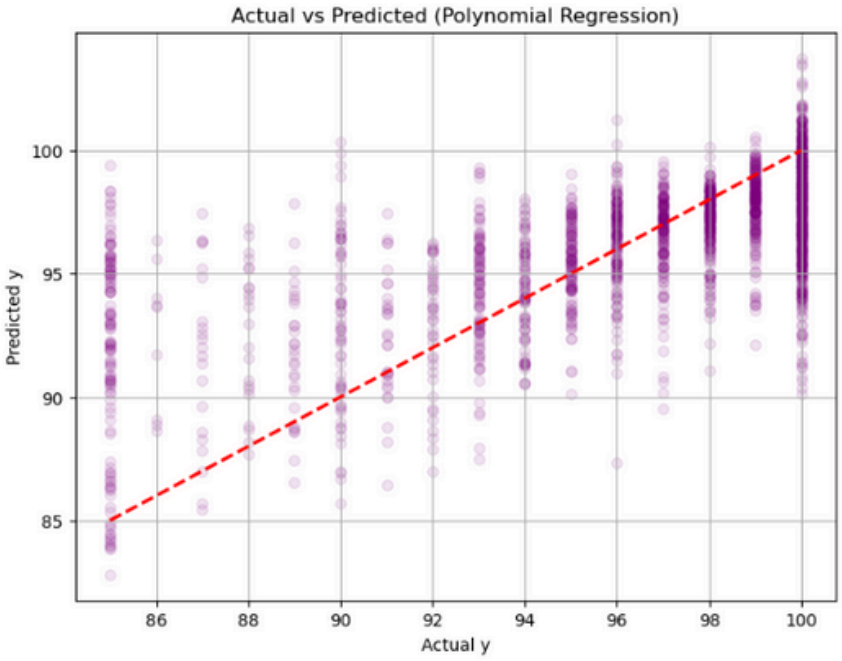

Model Performance: Actual vs Predicted
Linear
R²=0.432
Log Transform
R²=0.431
Sqrt Transform
R²=0.432
Polynomial
R²=0.484
Huber
R²=0.413
RANSAC
R²=0.432
Predicted
Actual

3.6) Polynomial Regression

Polynomial regression is an extension of linear regression that allows for non-linear relationships between the independent variables and the target variable. It creates polynomial features based on the original features.

Key Metrics:

- **R^2**: **0.4841 – The best R^2 score among all the models**, indicating that it explains more variance in the data than the other models.
- **MAE**: 2.2404 – Better than most models, indicating smaller errors on average.
- **RMSE**: 3.1765 – The lowest RMSE, suggesting that the predictions are more accurate.



3.7) Random Forest Regression

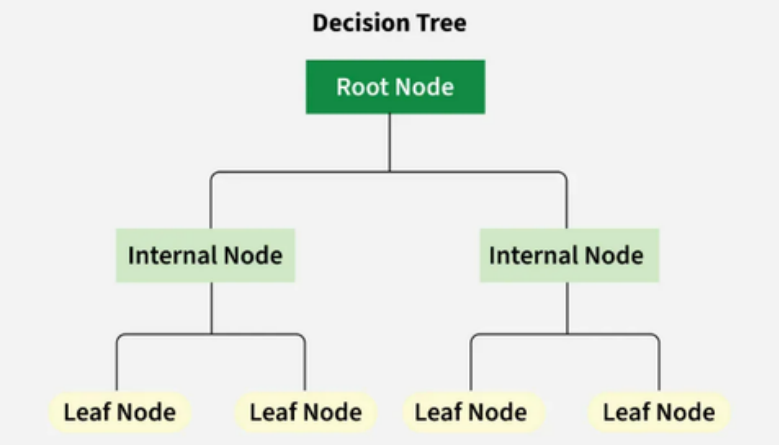
Random Forest is an ensemble learning method that creates multiple decision trees during training and outputs the mean prediction of the individual trees. It is very powerful for capturing non-linear relationships and interactions between features.

Key Metrics:

- **R^2**: 0.3862 – Slightly lower R^2 compared to polynomial regression, suggesting that this model explains less of the variance.
- **MAE**: 2.3402 – Similar to Linear Regression and other models.
- **MSE**: 12.3430 – A higher MSE compared to most linear models, which suggests more variance in prediction errors.

3.8) Decision Tree

Decision Trees are the foundation for many classical machine learning algorithms like Random Forests, Bagging, and Boosted Decision Trees. His idea was to represent data as a tree where each internal node denotes a test on an attribute (basically a condition), each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.



Why Use Decision Trees?

- Decision Trees usually mimic human thinking ability while making a decision, so it is easy to understand.
- The logic behind the decision tree can be easily understood because it shows a tree-like structure.

How to achieve a great result?

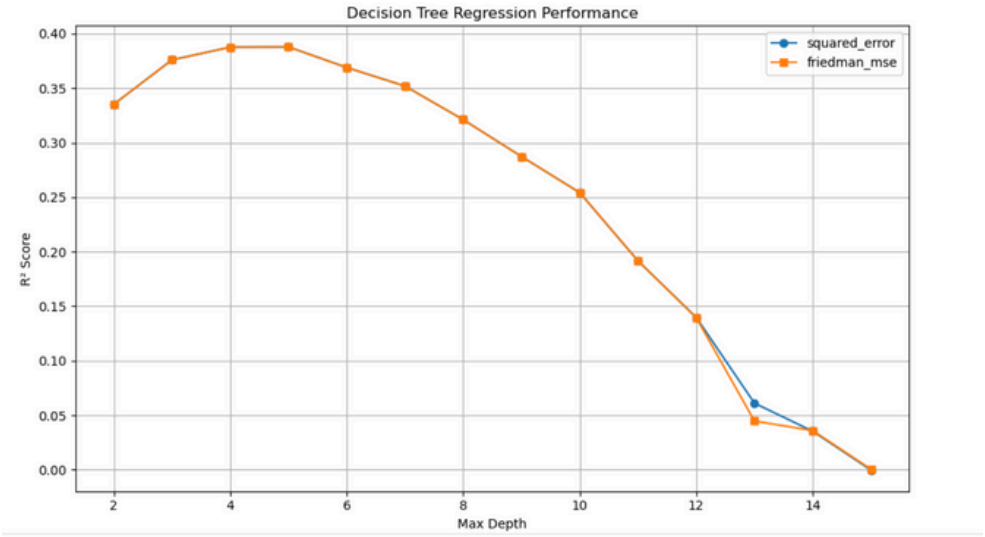
1- Gridsearch is used to compare between hyperparameters to get the best ones after plotting the results we noted that:

- The best max depth is 4.
- Mean Squared Error (MSE): 12.2953
- Mean Absolute Error (MAE): 2.3996
- R^2 Score: 0.3885

2- **Comparing between two splitting criteria:**

- Squared_error
- Friedman_mse

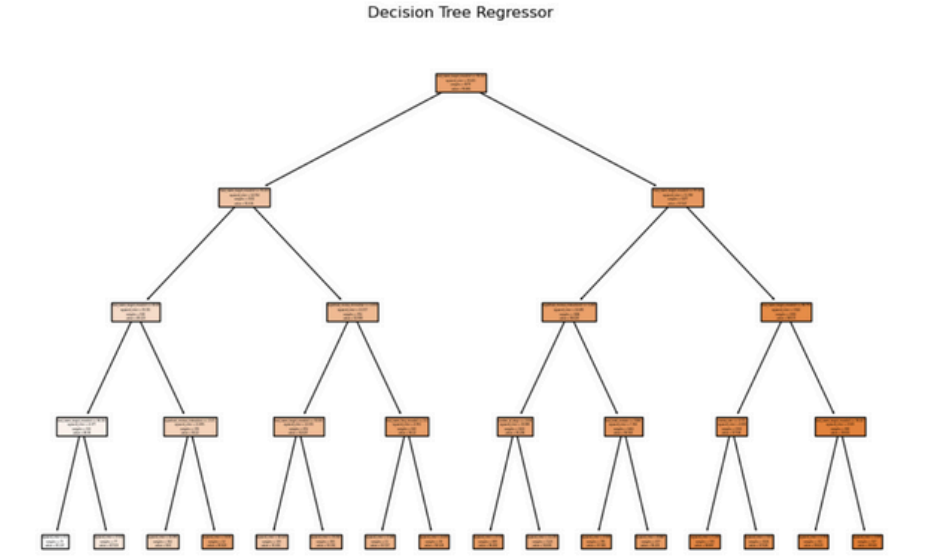
Comparing between two splitting criteria:



Then training the DT on the suitable parameters, according to r^2 scoring making cross validation = 10 samples/set

Final result:

- Decision_Tree R2_Train: 0.4287
- Decision_Tree R2_Test: 0.3876
- CV R^2 Score: 0.4292



3.9) Support Vector Regression

Support Vector Regression (SVR) is an algorithm based on the principles of Support Vector Machines, but designed for regression tasks. It aims to find an approximate function that stays as close as possible to most data points, allowing a certain margin of error (ε). The model seeks to remain as simple as possible and relies only on the points outside this margin (called Support Vectors) to define the final function.

In this work, we developed a Support Vector Regression (SVR) model using the **rbf kernel** to predict target values accurately.

Evaluation:

- Test R^2 Score: 0.3203
- CV R^2 Score: 0.3221

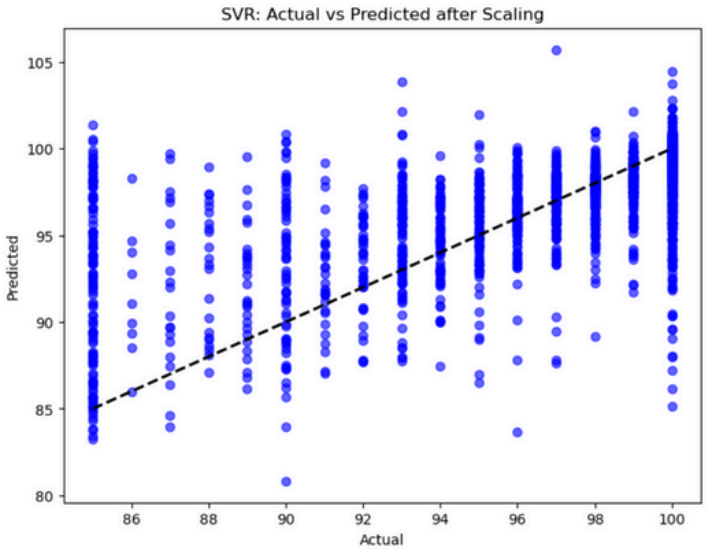
Steps to achieve a great results:

Feature scaling was applied using StandardScaler to standardize the data, ensuring optimal SVR performance and mitigating the effects of different feature scales.

Initialization with optimized hyperparameters (C=100, gamma='scale', epsilon=0.01) to balance model flexibility and robustness against outliers.

10-fold Cross-Validation was conducted, confirming the model's stability and ability to generalize to unseen data.

Actual vs predicted values showing a strong correlation and visually validating the model's effectiveness in capturing the underlying patterns of the data.



3.10) Comparison between Gradient Boosting, Random Forest, XGBoost

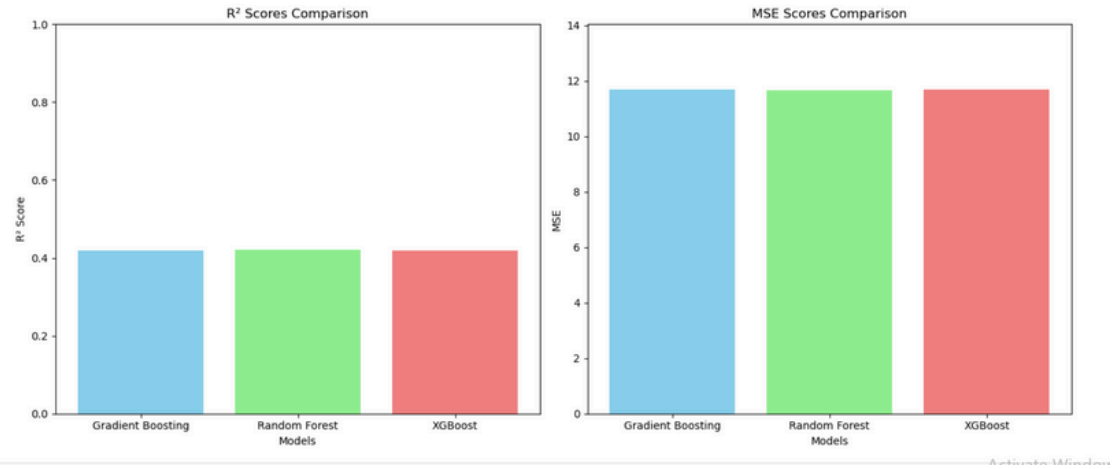
Gradient Boosting is a ensemble learning method used for classification and regression tasks. It is a boosting algorithm which combine multiple weak learner to create a strong predictive model. It works by sequentially training models where each new model tries to correct the errors made by its predecessor.

Random Forest algorithm is a powerful tree learning technique in Machine Learning to make predictions and then we do voting of all the tress to make prediction. They are widely used for classification and regression task

XGBoost is a powerful approach for building supervised regression models. The validity of this statement can be inferred by knowing about its (XGBoost) objective function and base learners. The objective function contains loss function and a regularization term. It tells about the difference between actual values and predicted values

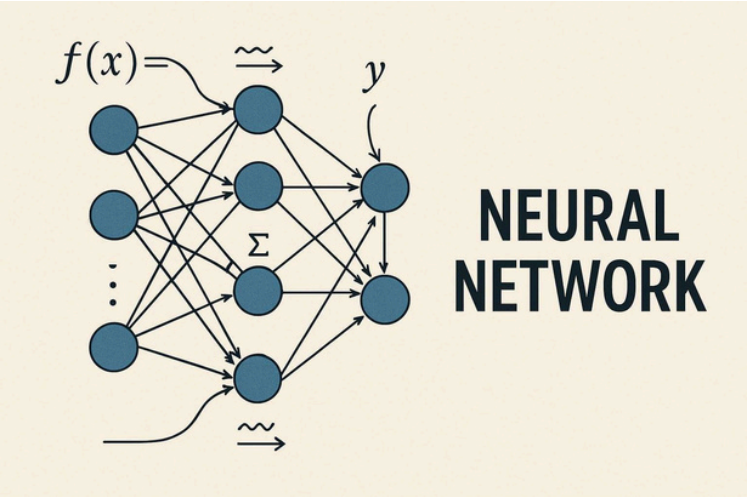
Results according to R^2:

- Gradient Boosting: 0.41806620
- RandomForest: 0.42002479
- XGBoost: 0.41918159



3.11) Neural Network

neural network is a computational model inspired by the way the human brain processes information. It consists of layers of interconnected nodes, or "neurons," that work together to recognize patterns, classify data, and make predictions. Each neuron takes input, processes it using mathematical functions, and passes the result to the next layer. Neural networks are widely used in machine learning tasks like image recognition, natural language processing, and predictive analytics, enabling systems to learn from data and improve over time.



In this work, we developed a Neural Network (NN) Regression Model using Keras and TensorFlow. The model was designed to predict continuous target variables based on input features.

We began by scaling the data using the Min-Max Scaler, ensuring the features were within a normalized range of [0, 1], which helps the neural network learn efficiently. The model itself consisted of an input layer, two hidden layers (with ReLU activation), a dropout layer to mitigate overfitting, and a final output layer with one neuron to predict the target value.

For training, we used 10-fold cross-validation to evaluate the model's performance on different data splits. We also computed R^2, RMSE, and MAE to assess the model's predictive accuracy.

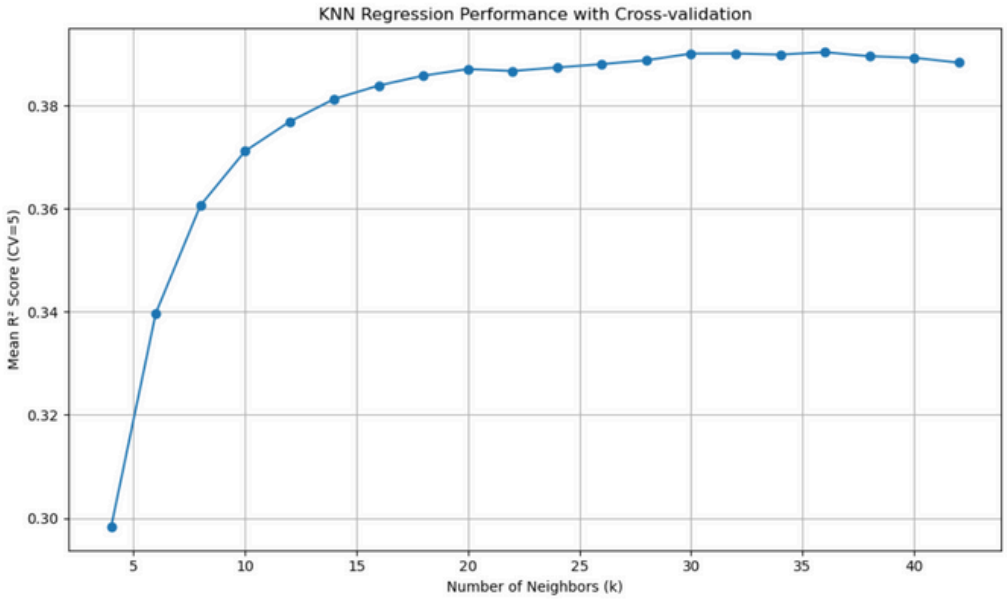
Results of Neural Network

- R2: 0.3857
- RMSE: 3.5145
- MAE: 2.5387
- CV R2: 0.3787 ± 0.0674

3.12) KNN

KNN regression is a non-parametric method used for predicting continuous values. The core idea is to predict the target value for a new data point by averaging the target values of the K nearest neighbors in the feature space.

To get the current result:
K is changed in range to get best r2
and cross validation technique is applied too!



Best number of neighbors (k): 36 with $R^2 = 0.3904$

3.13 & 3.14) CatBoost & LightGBM

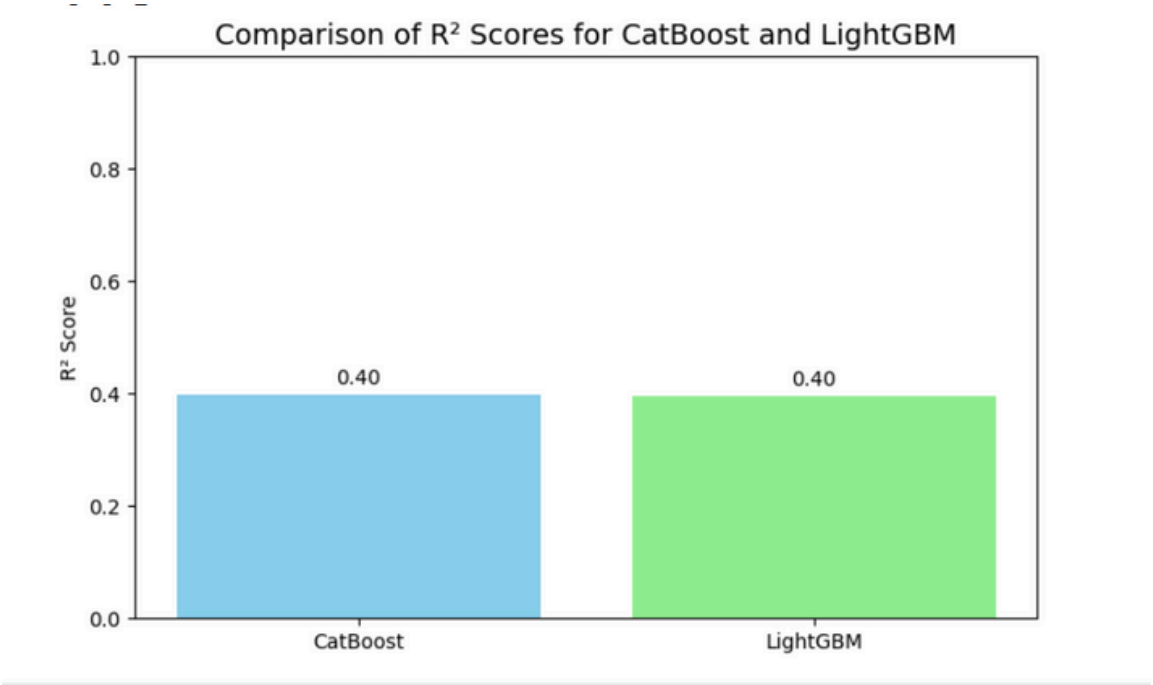
LightGBM and CatBoost are both powerful gradient boosting frameworks designed for efficient, high-performance machine learning.

LightGBM (Light Gradient Boosting Machine) is known for its speed and scalability, particularly in large datasets. It uses histogram-based methods for faster training and reduced memory usage, making it a top choice for many machine learning tasks.

On the other hand, CatBoost (Categorical Boosting) is optimized for handling categorical features without the need for extensive preprocessing like one-hot encoding. It's particularly useful when working with datasets that contain categorical data and offers strong performance with minimal tuning. Both frameworks are popular for classification and regression tasks, offering robust handling of complex data with high accuracy.

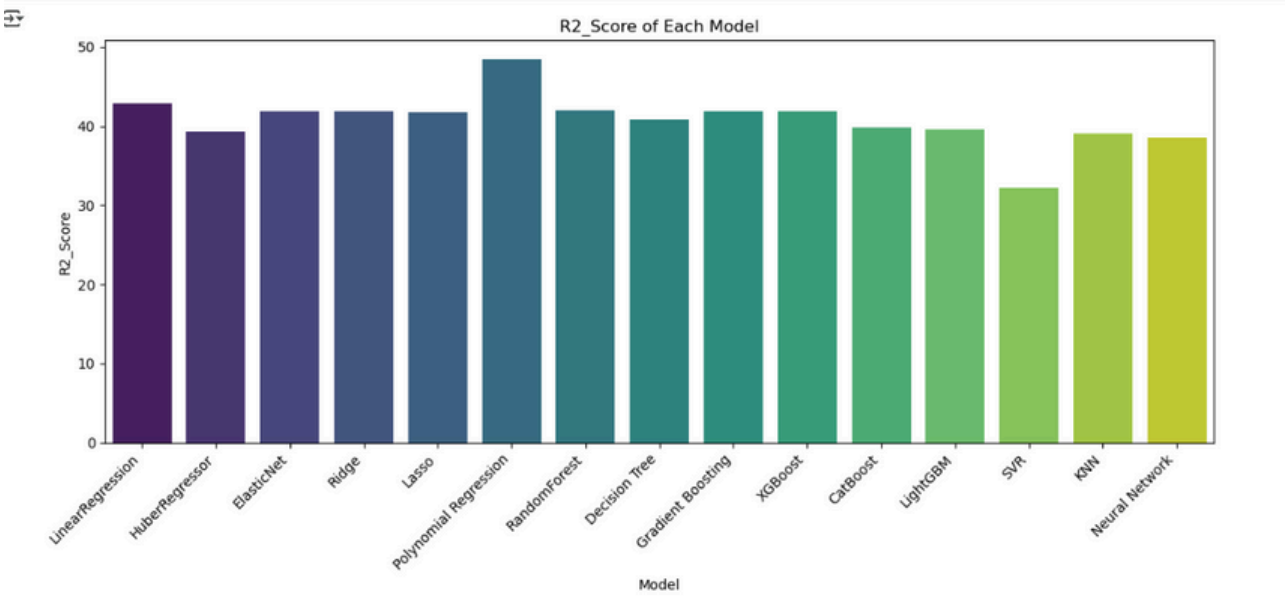
Results:

- catboost_model R^2: 0.3988
- lightgbm_model R^2: 0.3961



R^2 according different Models

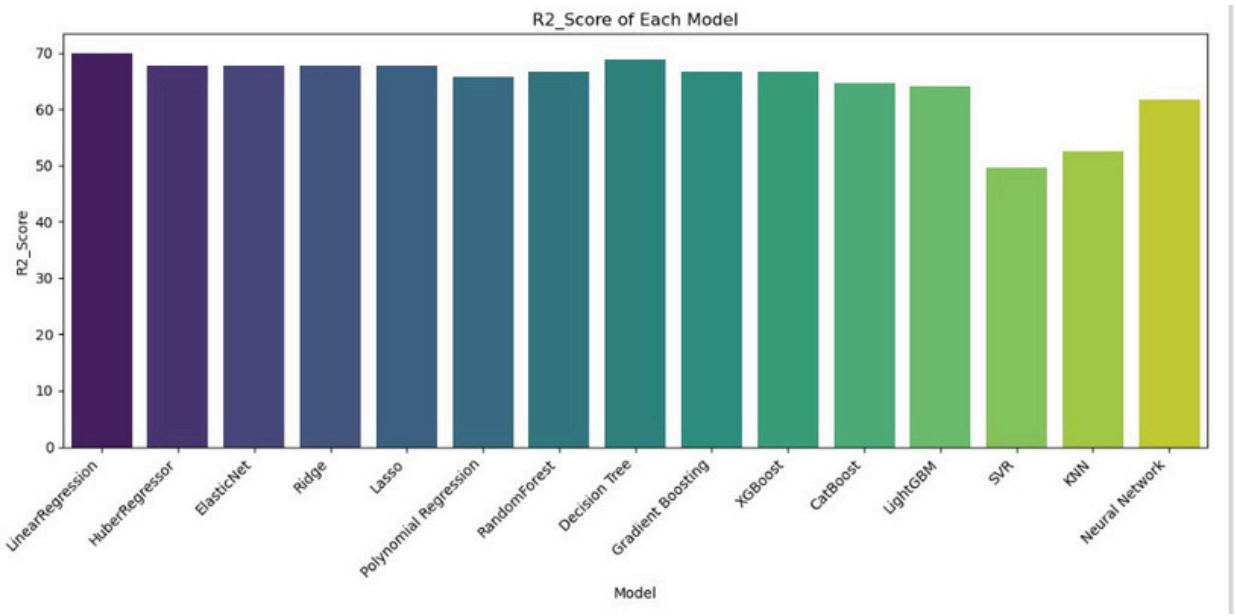
	Model	R2_Score
5	Polynomial Regression	48.421344
0	LinearRegression	42.915668
6	RandomForest	42.002479
9	XGBoost	41.918159
2	ElasticNet	41.834022
8	Gradient Boosting	41.806620
3	Ridge	41.805614
4	Lasso	41.787977
7	Decision Tree	40.873400
10	CatBoost	39.884824
11	LightGBM	39.612626
1	HuberRegressor	39.340759
13	KNN	39.039908
14	Neural Network	38.573110
12	SVR	32.207571



when we use target encoding with train and cross validation and create "host_avg_rate" that gropby "host_id" and "review score rating" mean

[117]:

	Model	R2_Score
0	LinearRegression	69.841522
7	Decision Tree	68.901394
4	Lasso	67.705985
2	ElasticNet	67.694141
3	Ridge	67.693362
1	HuberRegressor	67.652210
8	Gradient Boosting	66.640602
9	XGBoost	66.633419
6	RandomForest	66.562054
5	Polynomial Regression	65.781513
10	CatBoost	64.592223
11	LightGBM	64.066358
14	Neural Network	61.605113
13	KNN	52.523902
12	SVR	49.587542



Summary of Differences between Best Models:

- **Linear Models (Linear, Ridge, Lasso, ElasticNet):**
  - These models generally perform similarly with R^2 scores between 0.39 to 0.42.
  - The regularized versions (Ridge, Lasso, ElasticNet) help to prevent overfitting compared to standard linear regression.
  - Lasso performs feature selection, while Ridge shrinks coefficients to reduce model complexity.

- **Huber Regression:**
  - Slightly lower performance than linear models in terms of R^2 but is more robust to outliers.

- **Polynomial Regression:**
  - Significantly outperforms all other models with the highest R^2 and lowest MAE and RMSE..

- **Random Forest:**
  - More complex and captures non-linear relationships, but at the cost of interpretability. Its performance is comparable to linear models in terms of error metrics but has a slightly lower R^2.

Conclusion

- **Best Model for Predictive Accuracy: Polynomial Regression**, as it provides the highest R^2 score and the lowest error metrics.
- **Best for Robustness to Outliers: Huber Regression** is the most robust to outliers.
- **Best for Simplicity and Regularization: Ridge or ElasticNet** offer a good balance of regularization with simplicity.

4.Features Used and Discarded

In constructing our regression models, a meticulous feature selection and engineering process was executed to ensure both relevance and clarity. Below are the details of features retained and those removed:

• **Features Used:**

- **Numerical Metrics**: Key price variables (nightly price, price per stay), review counts, and host statistics (total listings, average response rate).
- **Categorical Indicators:** Binary flags (superhost status, instant booking), one-hot encoded room and property types, and clustered location labels.
- **Engineered Interactions**: Host experience × superhost status, reviews per stay, price per guest, and other ratio-based features.
- **Text-Derived Insights**: Sentiment compound scores extracted from listing summaries, host bios, and other description fields.
- **Temporal Features:** Days since host registration, first and last review measured from a fixed reference date, including cyclical month representations.
- **Amenities Profile**: Counts of amenities, presence indicators for high-impact amenities (e.g., Wi-Fi, kitchen), and an amenities rarity score based on inverse listing frequency.

• **Features Discarded:**

- **Columns with Excessive Missing Data**: Variables such as host acceptance rate, square footage, and thumbnail URLs were excluded due to over 40% missing values.
- **High-Cardinality Identifiers:** URL and ID fields (listing URL, host ID), street addresses, zipcode, and other location descriptors were removed to simplify the model.
- **Redundant Metrics:** Duplicated host listing counts were consolidated, and raw text fields dropped after sentiment extraction.
- **Low-Variance Binaries**: Flags with near-constant values (e.g., license requirements) were excluded to avoid noise.

5. Dataset Splits

- **Training Set: 80% of the data.**
  - After the train-test split (train_test_split with test_size=0.2), the training set comprises 80% of the total dataset.
  - Size: X_train.shape (**Training set size: (6979, 30)**).
- **Testing Set: 20% of the data.**
  - The remaining 20% is reserved for testing.
  - Size: X_test.shape (**Test set size: (1745, 30)**).
- **Validation Set:** Not explicitly split as a separate set.
  - Instead, **cross-validation** is used within the training set for model evaluation and hyperparameter tuning (e.g., cv=10 in GridSearchCV and cross_val_score). This effectively uses subsets of the training data as validation sets during the process.

The exact sizes depend on the original dataset (GuestSatisfactionPrediction.csv), but the proportions are fixed at 80% training and 20% testing, with validation handled via cross-validation.

6. Techniques to Improve Model Performance

A suite of preprocessing and modeling strategies was employed to maximize predictive accuracy and generalization capability:

1. Data Preprocessing:

- **Imputation:** KNN imputation for numeric gaps; mode substitution for categorical nulls.
- **Outlier Treatmen**t: Winsorization via IQR-based capping to mitigate extreme values.
- **Transformation**: Logarithmic scaling of skewed features (review counts, price variables).

2. Feature Engineering:

- **Interaction Terms:** Combined host and listing attributes (e.g., superhost_review_interaction).
- **Clustering:** KMeans clustering for host behavior and geographic coordinates to reveal latent groupings.
- **Text Sentiment**: NLP sentiment analysis to derive numeric sentiment features from listing descriptions.

3. Feature Selection:

- **Mutual Information:** SelectKBest with mutual_info_regression to identify the top 30 predictors.
- **Correlation Filtering:** Removal of features with correlation >0.9 to reduce multicollinearity.

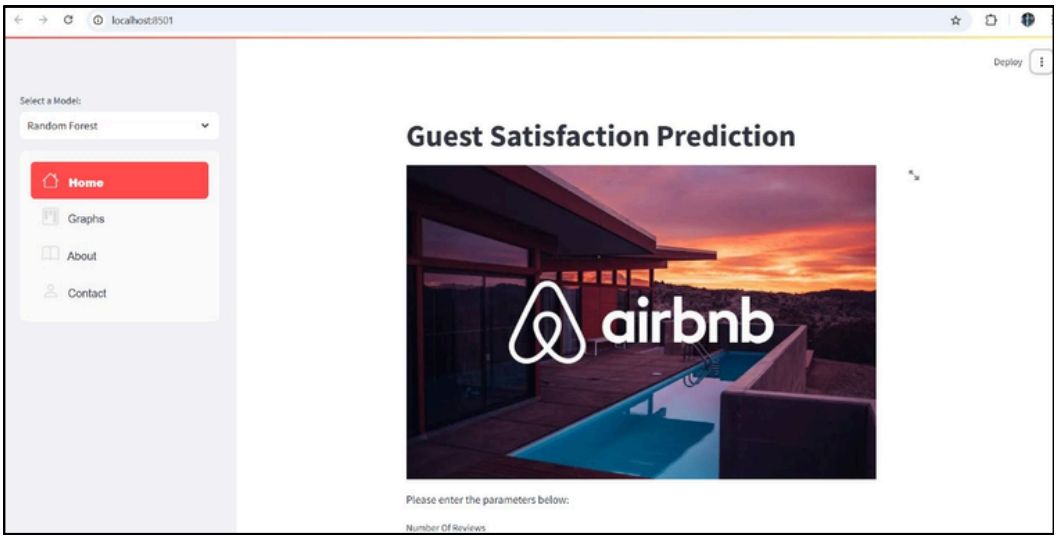
4. Scaling and Encoding:

- **StandardScaler**: Applied to normalize numeric features before model fitting.
- **One-Hot & Target Encoding:** Categorical handling based on cardinality and predictive relevance.

5. Model Tuning and Selection:

- **Cross-Validation**: 10-fold CV used throughout GridSearchCV for ridge, lasso, tree-based, and boosting models.
- **Ensemble Strategies**: Voting and stacking regressors combining Random Forest, XGBoost, and LightGBM.
- **Residual Correction**: Secondary tree-based model trained on linear regression residuals for improved fit.
- **Advanced Models**: Neural network architecture with two hidden layers and dropout to capture complex patterns.

GUI



This project is a Guest Satisfaction Prediction Web Application developed using Streamlit.
It machine learning models (Linear Regression, Decision Tree, Random Forest) to predict guest satisfaction levels for Airbnb stays based on multiple input features such as number of reviews, host listings count, minimum nights, price, and more.

Key features include:

Interactive Web Interface built with Streamlit.

User Input Fields for property and guest details.

Model Selection from the sidebar to choose the prediction algorithm.

Graphs and Visualizations showcasing relationships between features.

About Page describing the project goals.

Contact Form allowing users to send feedback directly to the developer's email.

Additional technologies:

Web animations using Lottie Files.

SMTP email integration for user feedback.

Data preprocessing and model training with Scikit-learn.

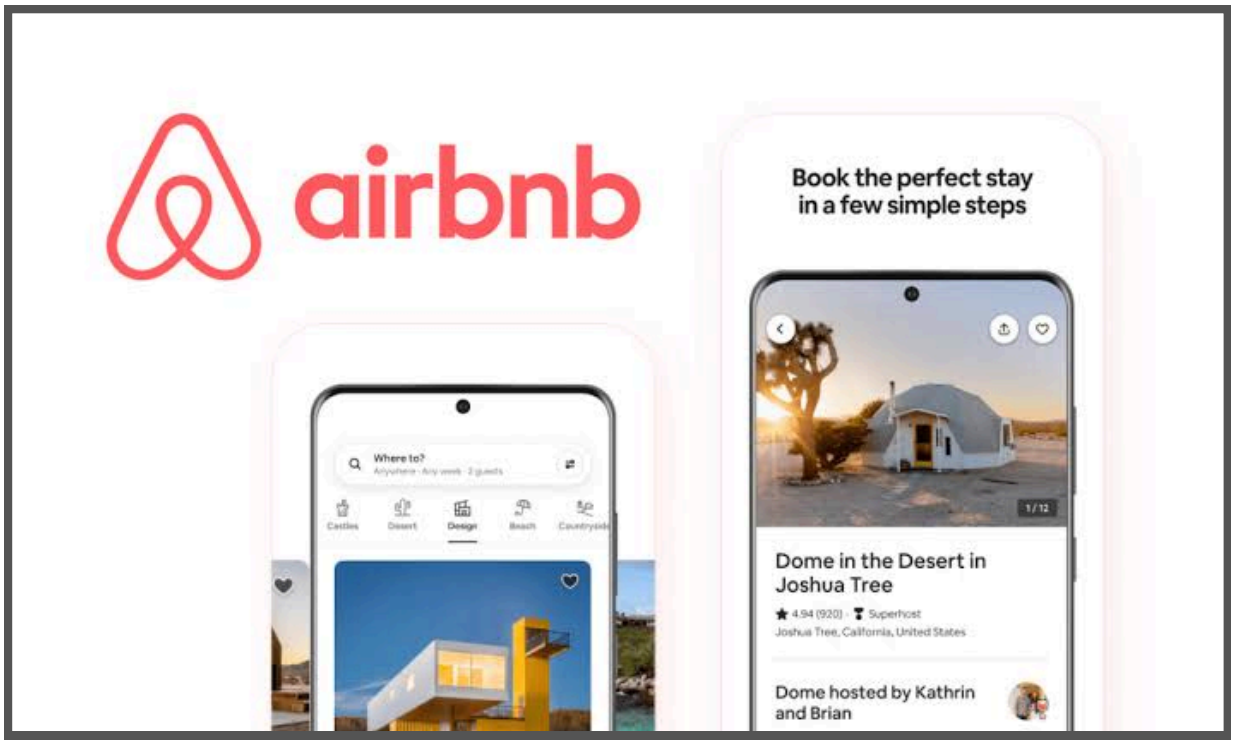
The project aims to empower hosts and Airbnb users by providing accurate satisfaction predictions to enhance the guest experience.

Mention any further techniques that were used to improve the results

We use Feature Engineering to increase corr with the target
AND
Web Scraping, but it needs LogIn:
This script scrapes user profile data from Airbnb after logging in automatically.
Since Airbnb sometimes requires 2FA verification (OTP sent via email), the script also handles fetching the OTP from Gmail automatically.
It uses Selenium (via undetected_chromedriver to avoid bot detection), BeautifulSoup to parse HTML, and IMAP to read emails.

Conclusion

In conclusion, our project combines the power of data-driven web scraping and machine learning to enhance guest satisfaction within the Airbnb community. By securely logging in and extracting valuable user profile data, we lay the foundation for building predictive models that accurately assess guest experiences. The insights gained from analyzing property details, host interactions, and guest feedback enable both guests and hosts to make more informed decisions. Through this work, we aim to foster a more personalized, satisfying, and memorable travel experience for all Airbnb users, advancing the future of hospitality with technology and innovation.



Thank You!