Airbnb Price Prediction Using Machine Learning (Part 1)

1. Introduction

The objective of this project was to build a machine learning model that predicts the log-transformed price of Airbnb listings. Using various listing features, the goal was to explore, preprocess, engineer features, and create a predictive model. The model uses Random Forest Regressor, and we evaluate it using multiple performance metrics such as MAE, RMSE, and R^2.

2. Data Loading and Exploration

The first step in this project was loading the data. We used pandas to read the data from the airbnb_data.csv file. After loading the data, we performed an initial exploration by using the following methods:

- .head(): Displays the first few rows of the dataset for a quick preview.
- .info(): Provides an overview of the dataset, including the number of non-null entries and data types of each column.
- .describe(): Summarizes the statistical distribution of numerical features.
- .isnull().sum(): Helps identify missing values in the dataset.

Significance: These steps helped in understanding the structure of the dataset, checking for missing data, and identifying potential issues like data types that need conversion (e.g., date fields).

3. Data Visualization

Once the data was loaded and understood, we proceeded with visualizing the target variable—log_price—to gain insights into its distribution:

- A **histogram** was plotted to understand the frequency distribution of log-transformed prices. This helps in understanding the shape of the distribution, whether it's skewed, and if log transformation is effective.
- A **boxplot** was used to identify potential outliers in the price data.

Significance: Visualization of the target variable is essential to understanding the data before applying any machine learning model. It helps in deciding transformations (like log transformation) and gives clues about outliers, which can affect model performance.

4. Feature Engineering

Feature engineering plays a crucial role in improving model performance. In this step, we:

- Converted the first_review and last_review columns into datetime objects and extracted new features:
    - first_review_year, first_review_month

    - last_review_year, last_review_month

- Created a new feature: host_experience_days, which represents the number of days since the host's first listing. This can be a strong predictor since experienced hosts tend to have better listings.

Significance: Date features provide temporal context, which is vital in understanding trends over time. The host's experience is a critical feature that impacts a listing's success and price. These transformations make the model more informative.

5. Categorical Encoding

The dataset contains several categorical variables, including property_type, room_type, bed_type, cancellation_policy, city, and neighbourhood. These need to be encoded to be usable by machine learning models:

- **One-hot encoding** was applied to these categorical features. This process converts each category into a separate binary column, indicating the presence or absence of that category.

Significance: Machine learning models require numerical input, and one-hot encoding is a common technique to represent categorical variables in a format that models can understand.

6. Data Cleaning and Scaling

In this step, we focused on handling missing values and scaling the features:

- The zipcode column was cleaned by extracting 5-digit numeric codes and converting them into a numerical format.
- Missing values in numerical columns like accommodates, bathrooms, and beds were imputed using the **median** for better handling of outliers.
- All numerical features were scaled using **StandardScaler**, which standardizes the features to have a mean of 0 and a standard deviation of 1. This is crucial because many machine learning algorithms, including Random Forest, benefit from scaled features.

Significance: Cleaning and scaling data ensures that the model treats all features uniformly, preventing any single feature from dominating due to scale differences. Imputation helps ensure that missing data does not lead to bias or loss of valuable information.

7. Model Development

With the features prepared, we proceeded to build and train the machine learning model. The steps were as follows:

- We used a **RandomForestRegressor** to predict log_price. This model was chosen for its ability to handle complex, non-linear relationships between features.
- The dataset was split into training and testing sets using train_test_split to evaluate the model's performance.

Significance: Random Forest is an ensemble method that can capture complex interactions between features, making it suitable for predicting prices based on a variety of input variables.

8. Preprocessing Pipeline

We constructed a preprocessing pipeline to ensure that all steps, from data imputation to encoding, were performed consistently during training and testing:

- **Numerical transformer**: Handled missing values with median imputation.
- **Categorical transformer**: Handled missing values with most frequent imputation and applied one-hot encoding.
- These transformations were combined using **ColumnTransformer** to apply different preprocessing to numerical and categorical features.

Significance: The preprocessing pipeline ensures that all transformations are applied uniformly to both training and test data. This is important to prevent data leakage and ensure that the model is evaluated on realistic, unseen data.

9. Model Evaluation

After training the model, we evaluated its performance using three key metrics:

- **MAE (Mean Absolute Error)**: Measures the average magnitude of errors in predictions. A lower value indicates better performance.
- **RMSE (Root Mean Squared Error)**: Penalizes larger errors more than MAE, giving us insight into the model's performance when handling extreme values.
- **R^2 (R-squared)**: Represents the proportion of variance in the target variable (log_price) explained by the model. A higher R^2 indicates a better fit.

The model performance was as follows:

- **MAE**: 0.3087
- **RMSE**: 0.4238
- **R^2**: 0.6449 (approximately 64.5% of the variance in price is explained by the model)

Significance: These metrics help assess the quality of the model. A good model should have a low MAE, low RMSE, and high R^2. The model's R^2 value of ~64.5% indicates that it does a fairly good job at explaining the variation in Airbnb prices.

10. Visualization of Model Predictions

To further assess model performance, we created the following visualizations:

- **Actual vs Predicted**: A scatter plot was used to compare actual prices to predicted prices. The closer the points are to the diagonal line, the better the model's predictions.

- **Residuals Distribution**: A histogram of the residuals (errors) was plotted to check for bias or skew in predictions. Ideally, the residuals should be normally distributed around zero.

Significance: These visualizations help in diagnosing the model's performance and understanding any potential issues, such as bias or systematic errors in predictions.

11. Conclusion

The project involved a systematic approach to predicting Airbnb prices using a Random Forest Regressor. Key steps included data exploration, cleaning, feature engineering, encoding, scaling, and model evaluation. The model performed reasonably well, with an R^2 value of 64.5%, meaning it explains a substantial portion of the variation in Airbnb prices. The combination of thoughtful preprocessing, feature creation, and model evaluation makes this a strong predictive model for Airbnb price estimation.

By following these steps, we can ensure that the model is accurate, interpretable, and robust, helping Airbnb hosts, guests, and even business analysts understand pricing patterns better.

Customer Churn Prediction Using Machine Learning (Part 2)

1. Data Loading and Exploration

The project began with loading customer data from 'customer_data.csv' containing 7,043 records and 21 features. Initial exploration was conducted using:

- df.shape: Confirmed 7,043 rows and 21 columns
- df.info(): Revealed 17 object columns, 2 integer columns, and 2 float columns
- df.describe(): Provided statistical summaries of numerical features
- df.isnull().sum(): Identified missing values in 'TotalCharges' (11 nulls)

Significance: These steps provided essential insights into data structure and quality issues requiring attention before model development.

2. Data Preprocessing

Several preprocessing steps were implemented to prepare the data for machine learning:

- **Missing Value Handling**: TotalCharges column was converted to numeric format using pd.to_numeric(errors='coerce') and missing values filled with the median value.
- **Binary Encoding**: Label encoding was applied to binary categorical variables ('gender', 'Partner', 'Dependents', 'PhoneService', 'PaperlessBilling', 'Churn') using LabelEncoder().
- **Categorical Encoding**: One-hot encoding was applied to multi-class categorical features ('MultipleLines', 'InternetService', 'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies', 'Contract', 'PaymentMethod') using pd.get_dummies().
- **Feature Scaling**: Numerical features ('MonthlyCharges', 'TotalCharges', 'tenure') were standardized using StandardScaler() to ensure all features contribute equally to model training.

Significance: These transformations converted all features into numerical format compatible with machine learning algorithms while preserving the information content of categorical variables.

3. Model Development

With preprocessed data, the modeling process proceeded:

- **Train-Test Split**: The dataset was divided into 80% training and 20% testing sets using train_test_split() with stratify=y to maintain the same class distribution in both sets.
- **Model Training**: Multiple classification algorithms were implemented:
    - Logistic Regression
    - Random Forest Classifier
    - Support Vector Machine (SVM)
    - K-Nearest Neighbors (KNN)
- **Hyperparameter Settings**:

- Logistic Regression: max_iter=1000
- Random Forest: n_estimators=100, random_state=42
- SVM: probability=True to enable probability estimates
- KNN: Default parameters

Significance: Different algorithms were tested to identify the most effective model for customer churn prediction.

4. Model Evaluation

Each model was evaluated using multiple metrics:

- **Performance Metrics**:
  - Accuracy: Proportion of correctly classified instances
  - Precision: Proportion of correctly predicted positive instances
  - Recall: Proportion of actual positive instances correctly identified
  - F1 Score: Harmonic mean of precision and recall
  - ROC AUC Score: Area under the ROC curve measuring discriminative ability
- **Visualization**:
  - Confusion matrices were generated to visualize true/false positives and negatives
  - Heatmap representations enhanced interpretability of model performance

Significance: These metrics provided a comprehensive assessment of each model's performance, particularly for the imbalanced classification problem of customer churn.

5. Results and Insights

The evaluation revealed:

- **Logistic Regression**: Achieved the best overall performance with:
    - Accuracy: 0.806
    - Precision: 0.658
    - Recall: 0.561
    - F1 Score: 0.606
    - ROC AUC: 0.842
- **Random Forest**:
    - Accuracy: 0.783
    - Precision: 0.610
    - Recall: 0.503
    - F1 Score: 0.551
    - ROC AUC: 0.818

The results indicate that while all models performed reasonably well, logistic regression provided the best balance between precision and recall for predicting customer churn, making it the most suitable model for this specific business problem.