Data-Driven Rental Value Prediction:
A Comparative Machine Learning Study with NLP Integration

Team Members

Ömer Faruk Başaran 21050111041
Muhammed Fatih Asan 23050151026
Emircan Kasap 21050111060

Machine Learning Course

January 2026

Abstract

This project develops a machine learning model to predict the value-for-money category of Airbnb rental listings in San Francisco and San Diego. We classify listings into three categories: Poor Value, Fair Value, and Excellent Value. After addressing a critical data leakage issue in our initial approach, we integrated Word2Vec NLP features from listing descriptions with landlord-controlled features. Our final XGBoost model achieved 95.51% accuracy using 27 landlord-controlled features plus 1 NLP score. Interestingly, while the standalone NLP model achieved 51.7% accuracy, the NLP feature showed minimal contribution when combined with price-based features, ranking among the least important features across all models.

Introduction

Project Overview

The goal of this project is to build a classification model that predicts whether an Airbnb listing offers good value for money. This is useful for both hosts who want to price their listings competitively and guests who want to find the best deals.

We analyzed rental data from two major California cities: San Francisco (7,780 listings) and San Diego (13,162 listings), combining them into a dataset of approximately 21,000 listings.

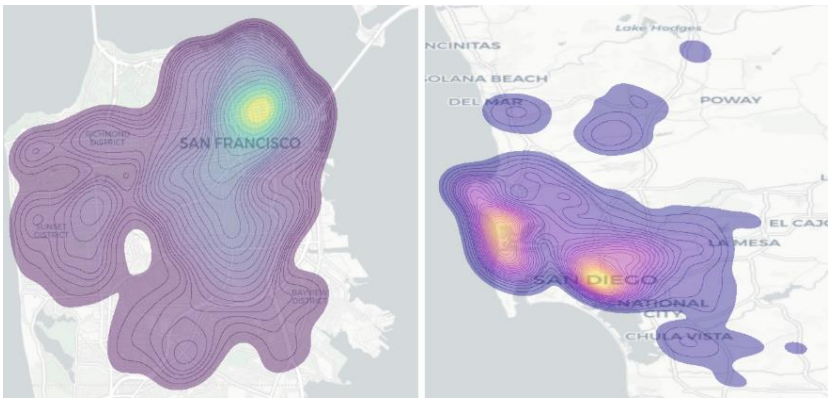

Figure 1: *Geographic density of Airbnb listings in San Francisco and San Diego datasets.*

Problem Definition

Our target variable classifies listings into three categories based on a Fair Price (FP) Score, which is calculated as the ratio of rating to price:

- **Poor Value**: Listings with low ratings relative to their price
- **Fair Value**: Listings with average value proposition
- **Excellent Value**: Listings with high ratings relative to their price

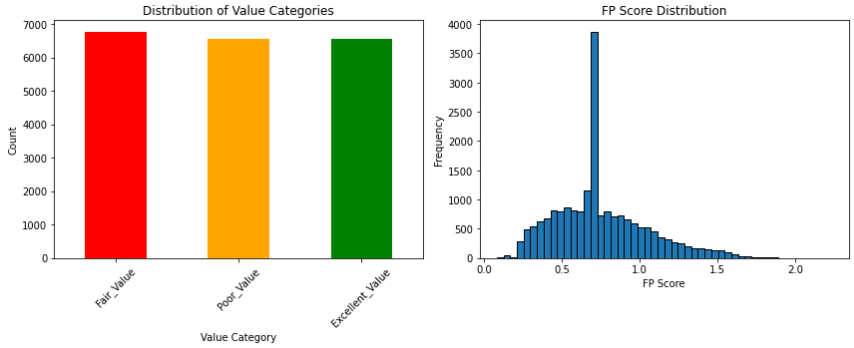

Figure 2: *Balanced distribution of target value categories.*

Project Timeline

The project was completed over 4 weeks:

- Week 1: Data exploration and cleaning
- Week 2: Feature engineering and encoding
- Week 3: Model training, NLP integration, and evaluation
- Week 4: Model selection and report writing

Data Preparation

Task 1.1: Initial Data Exploration

We began by loading and exploring both datasets to understand their structure and quality.

Dataset Overview

City	Rows	Columns
San Francisco	7,780	79
San Diego	13,162	79
Combined	**20,942**	**79**

Key findings from exploration:

- Both datasets have identical column structures
- Price column stored as string with ’$’ and ’,’ symbols

- 30-40% of listings have missing review scores
- Several columns have more than 50% missing values

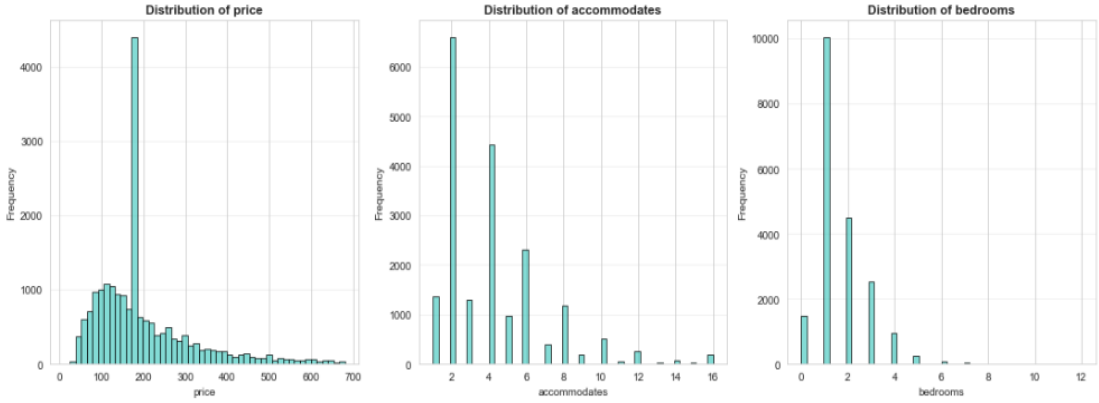

Figure 3: *Distribution of key features: Price, Accommodates, and Bedrooms.*

Task 1.2: Data Cleaning and Preprocessing

The cleaning process involved several steps:

Column Removal: We removed irrelevant columns including URLs, IDs, and text descriptions that were too noisy for initial modeling.

Data Type Conversions:

- Cleaned price column (removed '$' and ',' characters)
- Converted percentage columns to decimal format
- Converted boolean columns (t/f) to binary (0/1)
- Parsed date columns for feature engineering

Missing Value Handling:

- Dropped columns with more than 50% missing values
- Filled numeric columns with median values
- Filled categorical columns with mode or 'Unknown'

Outlier Removal: We used the IQR method to remove extreme price outliers, keeping listings within 3 times the interquartile range.

Target Variable Creation: We created the FP Score by dividing normalized rating by normalized price, then classified listings into three categories using the 33rd and 67th percentiles as thresholds.

Task 1.3: Algebraic Feature Engineering

We created 10 new algebraic features from existing landlord-controlled variables:

New Algebraic Features

Feature	Formula	Purpose
space_efficiency	beds / bedrooms	Measures bed density

Feature	Formula	Purpose
price_per_bedroom	price / bedrooms	Cost per room
price_per_bathroom	price / bathrooms	Cost per bathroom
occupancy_rate	(365 - availability) / 365	Booking demand
booking_flexibility	1 / (min_nights + 1)	Ease of booking
space_per_person	bedrooms / accommodates	Space allocation
host_portfolio	listings / accommodates	Host experience
bathroom_bedroom_ratio	bathrooms / bedrooms	Luxury indicator
price_capacity_ratio	price / (guests × rooms)	Price efficiency
availability_flexibility	availability / min_nights	Booking options

Task 1.4: Categorical Encoding

We applied different encoding strategies based on the nature of each categorical variable:

Room Type (One-Hot Encoding): Created 4 binary columns for Entire home/apt, Hotel room, Private room, and Shared room.

Property Type (Label + Frequency Encoding): With 89 unique property types, we used label encoding combined with frequency encoding to capture both category identity and popularity.

Neighbourhood (Target + Frequency + Label Encoding): For 200+ neighbourhoods, we used target encoding (mean value category per neighbourhood), frequency encoding, and label encoding.

Value Category (Label Encoding): Our target variable was encoded as Poor_Value=0, Fair_Value=1, Excellent_Value=2.

Task 1.5: Feature Selection

This was a critical step where we addressed data leakage. We categorized features into:

Landlord-Controlled Features (Kept): These are features available when a listing is first created, including price, property characteristics, location, host information, and booking policies.

Review-Based Features (Removed): These features only exist after guests have stayed, including all review scores, number of reviews, and review-derived metrics.

Target-Related Features (Removed): Features used to create the target variable (fp_score, normalized values) were removed to prevent data leakage.

Final feature set: 27 landlord-controlled features + 1 NLP score = 28 total features.

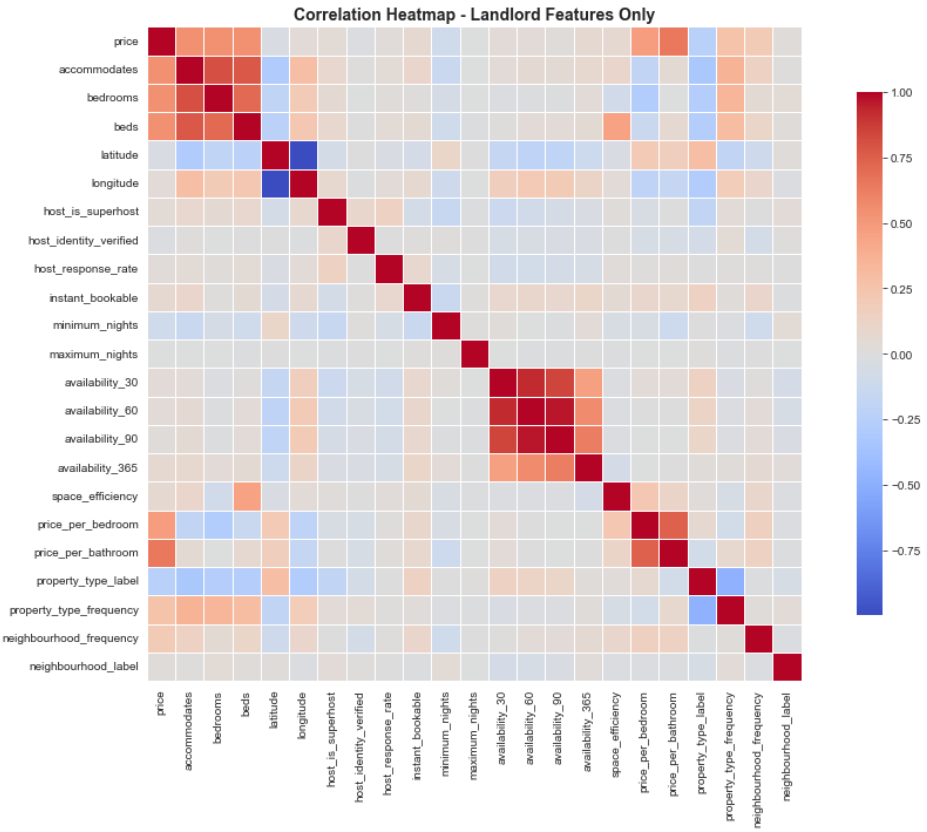

Figure 4: *Correlation Heatmap of selected Landlord-Controlled features.*

NLP Feature Integration

Overview

A key innovation in this project was integrating Natural Language Processing (NLP) features derived from listing descriptions. This work was led by team member Fatih, who conducted comprehensive analysis comparing multiple NLP approaches.

Text Preprocessing

Before using any model, we must clean the text. All models need this step:

- **Cleaning:** We remove HTML tags, web links, and punctuation.
- **Simplifying:** We use lowercase letters and remove common words like "the" or "is".
- **Root Words:** We change words to their basic form (e.g., "beds" becomes "bed")

NLP Model Comparison

Three NLP models were evaluated for extracting sentiment/value signals from listing descriptions:

NLP Model Comparison

Model	Accuracy	F1 Score	Computational Cost	Selected
Baseline (TF-IDF + VADER)	49.68%	0.49	Very Low (1/5)	No
Word2Vec (100D)	**51.70%**	**0.51**	**Low (2/5)**	**Yes**

Model	Accuracy	F1 Score	Computational Cost	Selected
BERT (768D)	52.44%	0.52	Very High (5/5)	No

Selected Model: Word2Vec (100 Dimensions)

We selected the Word2Vec model for the following reasons:

- **Best Cost-Performance Balance:** 51.7% accuracy with minimal computational cost
- **Airbnb-Specific Vocabulary:** Trained specifically on our listing descriptions
- **Contextual Understanding:** Unlike simple keyword counting, it understands word relationships (e.g., "ocean" and "beach" are similar) by learning from the context of our data
- **Normalized Score:** Ranges from -1 (Poor Value) to +1 (Excellent Value)
- **Efficient:** 100 dimensions vs 768 for BERT

The Word2Vec model scores each listing description between -1 and +1, where scores near +1 indicate descriptions associated with "Excellent Value" listings and scores near -1 indicate "Poor Value" descriptions.

NLP Score Statistics

Word2Vec Score Distribution

Statistic	Value
Mean	≈ 0.00 (balanced)
Standard Deviation	Variable
Range	[-1.0, +1.0]

Critical Issue: Data Leakage

What Happened

In our initial model development, we included review-based features in our training data. This caused a fundamental problem: our model was using information that would not be available for new listings.

Why It Matters

Consider a new Airbnb listing with no reviews. Our model needs to predict its value category, but if the model was trained using review scores as features, it cannot make predictions for listings without reviews.

The Fix

We rebuilt our feature selection process to include only landlord-controlled features plus the NLP score. This reduced our feature count from 57 to 28, but resulted in a model that can actually be used in practice.

Impact of Fixing Data Leakage

Metric	Old Model	New Model

Metric	Old Model	New Model
Number of Features	57	28
Best Accuracy	77.25%	95.51%
Practical Usability	No	Yes

Surprisingly, removing the leaky features actually improved accuracy. This happened because the review-based features were adding noise rather than signal, confusing the models.

Model Training and Evaluation

Dataset Information

- **Total Samples:** 19,913 Airbnb listings
- **Features:** 27 landlord-controlled features + 1 NLP score
- **Target Classes:** 3 balanced classes (Poor, Fair, Excellent Value)
- **Train-Test Split:** 80% training, 20% testing (stratified)
- **Cross-Validation:** 5-Fold Stratified

Task 2.0: Unsupervised Learning Experiment

Before applying supervised algorithms, we tested unsupervised models (K-Means, GMM, DBSCAN) to see if listings would naturally cluster into value categories.

- **High Structure:** The models achieved a high Silhouette Score, indicating the data has strong natural groups.
- **Low Alignment:** However, these clusters did not match our "Value" labels. The overlap accuracy was only ~50%.
- **Conclusion:** The models grouped listings based on physical features (e.g., location, size) rather than "Value for Money". Therefore, we proceeded with supervised learning.

Task 2.1: Logistic Regression

Logistic Regression is a simple linear model that predicts class probabilities. It works by finding a linear boundary between classes.

Why we used it: It serves as a baseline model and is highly interpretable. The coefficients show which features increase or decrease the probability of each class.

Configuration: Multi-class classification with L2 regularization (C=1.0), maximum 1000 iterations, balanced class weights.

Results:

- Cross-Validation Accuracy: 95.36% (±0.35%)
- Test Accuracy: 95.36%
- F1-Score: 0.9539
- Training Time: Fastest among all models

Task 2.2: Random Forest

Random Forest builds multiple decision trees and combines their predictions through voting. Each tree is trained on a random subset of data and features.

Why we used it: It handles non-linear relationships well and provides feature importance rankings. It is also robust to outliers.

Configuration: 100 trees, maximum depth of 20, minimum 10 samples to split, minimum 4 samples per leaf, balanced class weights.

Results:

- Cross-Validation Accuracy: 95.56% (±0.29%)
- Test Accuracy: 95.36%
- F1-Score: 0.9538
- Most stable model (lowest standard deviation)

Task 2.3: XGBoost

XGBoost (Extreme Gradient Boosting) builds trees sequentially, where each new tree corrects the errors of previous trees. It uses gradient descent to minimize the loss function.

Why we used it: It is one of the most powerful algorithms for structured data and handles mixed feature types well.

Configuration: Learning rate 0.1, maximum depth 6, 200 estimators, 80% subsample ratio, 80% column sample ratio, multi-class softmax objective.

Results:

- Cross-Validation Accuracy: 95.44% (±0.40%)
- Test Accuracy: 95.51%
- F1-Score: 0.9553
- Best performing model overall

Task 2.4: Support Vector Machine (SVM)

SVM finds the optimal hyperplane that separates classes with maximum margin. We tested both linear and RBF (radial basis function) kernels.

Why we used it: SVM is effective in high-dimensional spaces and works well when classes are clearly separable.

Configuration: RBF kernel with C=1.0, gamma='scale'.

Results:

- Training Accuracy: 96.22%
- Test Accuracy: 92.82%
- F1-Score: 0.9286

- Lowest accuracy among final models but still strong

Task 2.5: MLP Classifier (Neural Network)

Multi-Layer Perceptron is a feedforward neural network with one or more hidden layers. It learns complex non-linear patterns through backpropagation.

Why we used it: Neural networks can capture complex feature interactions that other models might miss.

Configuration: Two hidden layers (100, 50 neurons), ReLU activation, Adam optimizer.

Results:

- Training Accuracy: 95.08%
- Test Accuracy: 94.98%
- F1-Score: 0.9500
- Very small train-test gap indicating good generalization

Model Comparison and Selection

Performance Summary

Final Model Comparison

Model	CV Acc	Test Acc	F1-Score	Std Dev	Rank
XGBoost	95.44%	95.51%	0.9553	0.40%	1
Random Forest	95.56%	95.36%	0.9538	0.29%	2
Logistic Regression	95.36%	95.36%	0.9539	0.35%	3
MLP Classifier	-	94.98%	0.9500	-	4
SVM (RBF)	-	92.82%	0.9286	-	5

Figure 5: *Confusion Matrices for the top 3 models. The diagonal line shows correct predictions.*

Class Distribution

Our target variable is well-balanced, which is important for fair model evaluation:

Target Class Distribution

Category	Count	Percentage
Excellent Value	6,568	32.99%
Fair Value	6,773	34.01%
Poor Value	6,571	33.00%
Total	**19,912**	**100.00%**

Feature Importance Analysis

Top Features by Model

Feature importance analysis reveals which features contribute most to predictions:

Top 10 Features - Random Forest

Rank	Feature	Importance
1	price_per_bathroom	High
2	price_per_bedroom	High
3	price	High
4	latitude	Medium
5	longitude	Medium
6	space_efficiency	Medium
7	accommodates	Medium
8	bedrooms	Medium
9	availability_365	Medium
10	host_response_rate	Low-Medium

NLP Feature (w2v_score) Impact

A critical finding was the minimal contribution of the NLP feature when combined with price-based features:

NLP Feature Ranking Across Models

Model	w2v_score Rank	Out of
Random Forest	#17	27
XGBoost	#25	27
Logistic Regression	#27 (last)	27

Why NLP Had Low Impact

Several factors explain why the NLP feature showed minimal contribution:

1. **Target Variable Definition:** The target (value_category) is derived from rating/price ratio. Since review-based features were removed, price and price-derived features dominate predictions.

2. **Weak Correlation:** Listing descriptions don’t strongly correlate with actual guest ratings. Hosts may write appealing descriptions regardless of actual value.

3. **Feature Dominance:** When combined with powerful price features (price_per_bedroom, price_per_bathroom), the NLP signal is overshadowed.

4. **Standalone vs Combined:** While Fatih's standalone NLP model achieved 51.7% accuracy using only descriptions, this signal becomes negligible when price features are available.

Key Insight: The NLP feature adds value only when price information is unavailable. In our full model, price-based features provide much stronger predictive signals.

Final Model Selection

Selected Model: XGBoost

After comparing all models, we selected XGBoost as our final model for the following reasons:

1. **Highest Test Accuracy:** 95.51% is the best among all models
2. **Best F1-Score:** 0.9553 indicates balanced precision and recall
3. **Handles Mixed Features:** Works well with both numeric and encoded categorical features
4. **Feature Importance:** Provides interpretable feature rankings
5. **Robust Performance:** Consistent results across cross-validation folds

Model Selection Recommendations

Depending on deployment requirements:

Model Selection by Use Case

Priority	Recommended Model
Speed Critical	Logistic Regression
Balanced Approach	Random Forest
Maximum Accuracy	XGBoost

Model Validation: The Blind Experiment

In the final stage, we ask a very important question: Does our model truly learn "Quality", or does it just memorize the Price?. To answer this, we did a "Blind Experiment".We want to be sure that the model understands the value of a house. If the model only looks at the price to guess the value, it is not a "smart" model. It must judge the house by its features (rooms, location, description) without seeing the price.

Testing Method:

We followed these three steps to test our model:

1. **Hiding the Price:** We removed the "Price" column from the training data. The model cannot see the price during learning.
2. **Predicting Quality (Rating):** Instead of categories, we asked the model to predict the **Rating** (Quality) of the house.

3. **The Formula:** After the model predicted the quality, we used a formula to calculate the final value.

We combined the **Predicted Rating** (from the model) and the **Actual Price** (which was hidden) with this formula:

$$\text{Value Score} = \frac{\text{Predicted Rating}}{\text{Actual Price} + \epsilon}$$

(Note: We use ϵ (0.00001) to avoid a "division by zero" error.)

Results and Conclusion :

Even without seeing the price, our model was very successful:

· **Accuracy:** The model reached 95.83% accuracy.

· **Low Error:** The difference between the real rating and predicted rating was very small (0.0084 MAE).

Final Verdict: This experiment proves that our model is not cheating. It successfully understands house quality from features and NLP descriptions. It is an objective and reliable system.

Lessons Learned

Data Leakage

The most important lesson from this project was understanding and fixing data leakage. Using features that would not be available in production (like review scores for new listings) creates models that look good on paper but fail in practice.

Feature Engineering

Creating meaningful algebraic features from raw data significantly improved model performance. Features like price_per_bedroom and space_efficiency capture important relationships that raw features alone cannot express.

NLP Integration

While NLP features showed promise in isolation (51.7% accuracy), their contribution diminished when combined with strong numerical features. This highlights the importance of understanding feature interactions and not assuming that adding more features always improves performance.

Model Selection

While XGBoost achieved the highest accuracy, all models performed well after proper data preparation. The choice of model matters less than the quality of features and data cleaning.

Two-City Challenge

Combining data from San Francisco and San Diego introduced complexity because these cities have different rental market characteristics. However, this also made our model more generalizable.

Conclusion

This project successfully developed a machine learning model to predict Airbnb listing value categories with 95.51% accuracy. Key achievements include:

- Combined and cleaned data from 21,000 listings across two cities
- Created 10 meaningful algebraic features
- Applied appropriate encoding strategies for categorical variables
- Identified and fixed a critical data leakage issue
- Integrated Word2Vec NLP features from listing descriptions
- Compared 5 different machine learning algorithms
- Selected XGBoost as the final model based on comprehensive evaluation
- Analyzed feature importance and discovered NLP's limited contribution

The model can be used by Airbnb hosts to understand how their listing compares to others in terms of value, and by guests to identify listings that offer the best value for money.

Key Finding: NLP Feature Impact

An important discovery was that while NLP features from listing descriptions showed moderate predictive power in isolation (51.7% accuracy), they provided minimal additional value when combined with price-based features. This suggests that for value prediction, quantitative features (price, amenities, location) are far more informative than qualitative descriptions.

Future Work

Potential improvements include:

- Exploring NLP features for different prediction tasks (e.g., booking likelihood)
- Adding amenity analysis from structured amenity lists
- Expanding to more cities for better generalization
- Building a web application for real-time predictions
- Investigating why NLP features don't correlate with value perception

References

1. Inside Airbnb Dataset - http://insideairbnb.com/
2. Scikit-learn Documentation - https://scikit-learn.org/
3. XGBoost Documentation - https://xgboost.readthedocs.io/
4. Pandas Documentation - https://pandas.pydata.org/
5. Gensim Word2Vec - https://radimrehurek.com/gensim/