ACL Report

Seif Selim Halla Tarek Youssef Yasser Omar Ayman

October 2025

Contents

1 Introduction

1.1 Dataset Overview

The analysis was conducted using three tables — **Hotels**, **Reviews**, and **Users** — from the publicly available Hotels dataset on Kaggle [1].

- **Source:** Kaggle
- **Tables Used:** Hotels, Reviews, Users
- **Number of Samples:** Hotels – 25, Reviews – 50,000, Users – 2,000
- **Number of Features:** Hotels – 10, Reviews – 10, Users – 6

2 Data Engineering

This section outlines the data engineering process and analyses conducted to answer the two main questions of the project. It includes dataset description, cleaning, column selection, and implementation details for each question.

2.1 Data Cleaning

All three tables were checked for missing values and duplicates. No issues were found, confirming the dataset's integrity.

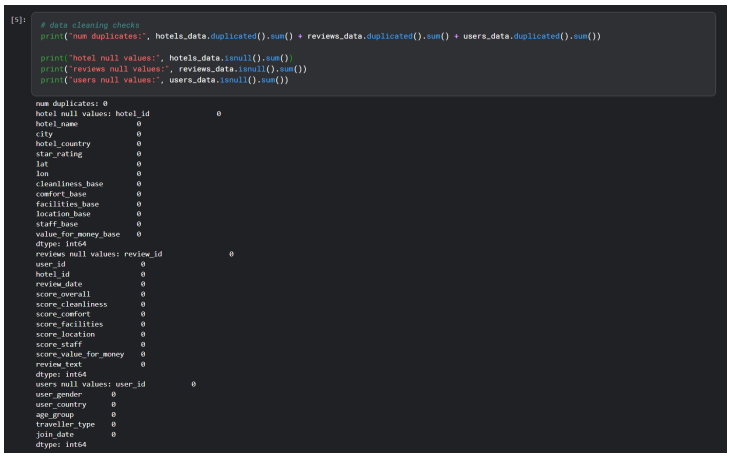


Figure 1: Data validation results confirming zero duplicates and missing values across all tables.

2.2 Columns Used

The analysis utilized selected columns from each table:

- **Users:** `user_id`, `traveller_type`, `age_group`
- **Hotels:** `hotel_id`, `city`, `hotel_country`
- **Reviews:** `review_id`, `score_overall`, `score_value_for_money`

2.3 Question 1: Which city is best for each traveler type?

Objective: Identify the highest-rated city for each traveler type based on the overall review score.

Procedure:

1. Merge `Users` and `Reviews` on `user_id` using a left join to combine traveler type with review scores.
2. Merge the resulting dataset with `Hotels` on `hotel_id` to include city information.
3. The resulting columns include: `review_id`, `user_id`, `hotel_id`, `score_overall`, `traveller_type`, and `city`.
4. Group by `traveller_type` and `city`, then compute the mean of `score_overall`.
5. Sort cities within each traveler type by descending average score.
6. Select the top city per traveler type.

```
# data engineering question (1)
df_one = reviews_data.merge(users_data[["user_id", "traveller_type"]], on = "user_id", how = "left")
df_one = df_one.merge(hotels_data[["hotel_id", "city"]], on = "hotel_id", how = "left")

city_traveler_scores = df_one.groupby(['traveller_type', 'city'])['score_overall'].mean().reset_index()

city_traveler_scores = city_traveler_scores.sort_values(
    by=['traveller_type', 'score_overall'],
    ascending=[True, False]
)

best_city_per_traveler = city_traveler_scores.groupby('traveller_type').first().reset_index()

best_city_per_traveler.head()
```

Figure 2: Implementation workflow for Question 1.

Results:

[11]:

	traveller_type	city	score_overall
0	Business	Dubai	8.965668
1	Couple	Amsterdam	9.096989
2	Family	Dubai	9.214381
3	Solo	Amsterdam	9.108454

Figure 3: Top city for each traveler type based on average overall score.

Visualization:

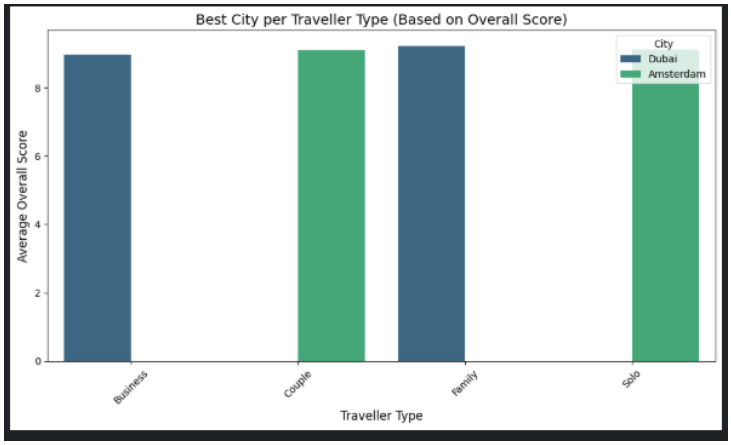


Figure 4: Visualization of the best city per traveler type.

2.4 Question 2: What are the top 3 countries with the best value-for-money score per age group?

Objective: Determine which countries have the best perceived value-for-money across different age groups.

Procedure:

1. Merge `Users` and `Reviews` on `user_id` using a left join to combine age group with review data.
2. Merge the result with `Hotels` on `hotel_id` to include `hotel_country`.
3. The resulting columns include: `review_id`, `user_id`, `hotel_id`, `score_value_for_money`, `age_group`, and `hotel_country`.
4. Group by `age_group` and `hotel_country`, then calculate the mean of `score_value_for_money`.
5. Sort the countries for each age group and select the top three.

```
[9]: # data engineering question (2)
df_two = reviews_data.merge(users_data[["user_id", "age_group"]], on = "user_id", how = "left")
df_two = df_two.merge(hotels_data[["hotel_id", "hotel_country"]], on = "hotel_id", how = "left")

country_age_vfm = df_two.groupby(
    ['age_group', 'hotel_country']
)['score_value_for_money'].mean().reset_index()

top_3_per_age = country_age_vfm.groupby('age_group').head(3)

print(top_3_per_age)
```

Figure 5: Implementation workflow for Question 2.

Results:

	age_group	hotel_country	score_value_for_money
0	18-24	Argentina	8.557732
1	18-24	Australia	8.400926
2	18-24	Brazil	8.477778
25	25-34	Argentina	8.562025
26	25-34	Australia	8.378873
27	25-34	Brazil	8.535878
50	35-44	Argentina	8.579133
51	35-44	Australia	8.370909
52	35-44	Brazil	8.490172
75	45-54	Argentina	8.510938
76	45-54	Australia	8.337324
77	45-54	Brazil	8.511438
100	55+	Argentina	8.545989
101	55+	Australia	8.355714
102	55+	Brazil	8.467380

Figure 6: Top 3 countries per age group by average value-for-money score.

Visualization:

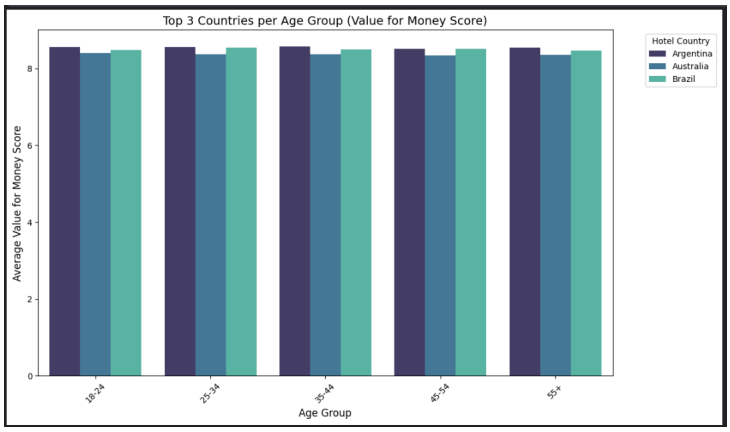


Figure 7: Visualization of the top countries for each age group.

3 Predictive Model

This section describes the models built, their architectures, preprocessing techniques, evaluation metrics, and comparative performance.

3.1 Features Used

Dropped Features:

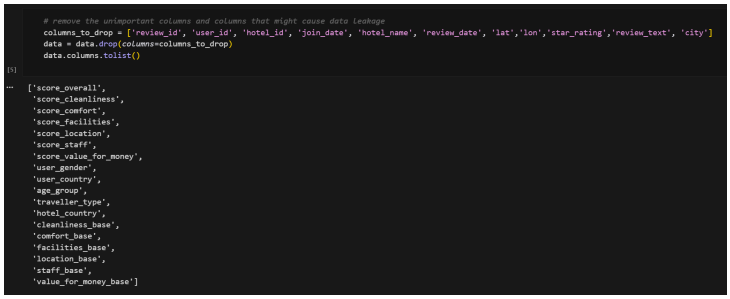


Figure 8: Features removed during preprocessing

Final Available Features: `score_overall`, `score_cleanliness`, `score_comfort`, `score_facilities`, `score_location`, `score_staff`, `score_value_for_money`, `user_gender`, `user_country`, `age_group`, `traveller_type`, `hotel_country`, `cleanliness_base`, `comfort_base`, `facilities_base`, `location_base`, `staff_base`, `value_for_money_base`.

3.2 Data Preprocessing

1. **Label Encoding for User Gender:** Encoded `user_gender` with the mapping: `"Male" = 0`, `"Female" = 1`, and `"Other" = 2`. Checked data consistency (capitalization of first letters) before encoding.

2. **Label Encoding for Age Group:** Encoded `age_group` with the mapping: `"18-24" = 0`, `"25-34" = 1`, `"35-44" = 2`, `"45-54" = 3`, `"55+" = 4`. The encoded values were stored in a new column `age_encoded`, and the original `age_group` column was dropped.

3. **One-Hot Encoding:** Applied one-hot encoding to:

   - `traveller_type` using the prefix `traveller_`.
   - `user_country` using the prefix `user_country_`.

4. **Hotel Country Grouping:** The `hotel_country` feature was mapped using a custom `country_group_map`. The new grouped feature was added as `country_group`, and the original column was dropped to prevent overfitting.

Figure 9: Country grouping process for hotel locations

3.3 Model Selection

All experiments utilized the **Random Forest Classifier**, with an 80/20 train-test split and a fixed `random_state = 42` (a common industry standard for reproducibility).

We tested three different configurations, each addressing potential overfitting and data leakage in distinct ways. The performance and insights for each are detailed below.

3.3.1 Model 1: Baseline (No Feature Engineering or Overfitting)

Approach: Columns that could uniquely identify hotels were removed to avoid overfitting.

Dropped Columns: `country_group`, `comfort_base`, `facilities_base`, `location_base`, `staff_base`, `value_for_money_base`, `cleanliness_base`.

Results:

```
# Model 1
# create a model that trains on the data to predict the country_group based on the current remaining features
# for the first model we'll remove the base rating columns to avoid data leakage
X = data.drop(columns=['country_group', 'comfort_base', 'facilities_base', 'location_base', 'staff_base', 'value_for_money_base', 'cleanliness_base'
])
y = data['country_group']
X_train_1, X_test_1, y_train_1, y_test_1 = train_test_split(X, y, test_size=0.2, random_state=42)
rf_1 = RandomForestClassifier(n_estimators=300, random_state=42)
rf_1.fit(X_train_1, y_train_1)
y_pred = rf_1.predict(X_test_1)
accuracy = accuracy_score(y_test_1, y_pred)
print("Accuracy:", accuracy)
print(classification_report(y_test_1, y_pred))
```

✓ 162s

```
Accuracy: 0.3942
                     precision    recall  f1-score   support

             Africa       0.38      0.46      0.41      1247
          East_Asia       0.44      0.46      0.45      1195
     Eastern_Europe       0.24      0.06      0.10       358
        Middle_East       0.42      0.24      0.30       816
      North_America       0.43      0.25      0.32       776
North_America_Mexico       0.29      0.09      0.14       400
            Oceania       0.33      0.11      0.17       805
      South_America       0.33      0.25      0.28       813
         South_Asia       0.16      0.04      0.06       389
    Southeast_Asia       0.41      0.20      0.27       818
    Western_Europe       0.41      0.88      0.54      2383

          accuracy                           0.39     10000
         macro avg       0.35      0.27      0.28     10000
      weighted avg       0.38      0.39      0.35     10000
```

Figure 10: Model 1 results (baseline)

Accuracy: 39%

Conclusion: The current features lacked sufficient predictive strength to identify meaningful patterns, leading to poor model generalization.

3.3.2 Model 2: Final Model (No Leakage)

Approach: To eliminate leakage, base features were averaged across all hotels and rounded to two decimal places—similar to review score formatting—thus removing their unique identifying power.

Results:

```
# Model 2
# create a model that trains on the data to predict the country_group based on features including an engineered average for base ratings
# the average will be rounded to 1 decimal place which might help reduce noise and cluster the data better
model2_data = data.copy()
model2_data['hotel_avg_base'] = model2_data[['cleanliness_base', 'comfort_base', 'facilities_base',
                                             'location_base', 'staff_base', 'value_for_money_base']].mean(axis=1).round(1)
X = model2_data.drop(columns=['country_group', 'comfort_base', 'facilities_base', 'location_base', 'staff_base', 'value_for_money_base', 'cleanliness_base'
])
y = model2_data['country_group']
X_train_2, X_test_2, y_train_2, y_test_2 = train_test_split(X, y, test_size=0.2, random_state=42)
rf_2 = RandomForestClassifier(n_estimators=300, random_state=42)
rf_2.fit(X_train_2, y_train_2)
y_pred = rf_2.predict(X_test_2)
accuracy = accuracy_score(y_test_2, y_pred)
print("Accuracy:", accuracy)
print(classification_report(y_test_2, y_pred))
```

✓ 139s

```
Accuracy: 0.6124
                     precision    recall  f1-score   support

             Africa       0.73      0.73      0.73      1247
          East_Asia       0.58      0.55      0.56      1195
     Eastern_Europe       0.65      0.57      0.61       358
        Middle_East       0.75      0.70      0.73       816
      North_America       0.66      0.57      0.61       776
North_America_Mexico       0.70      0.51      0.59       400
            Oceania       0.49      0.29      0.37       805
      South_America       0.54      0.62      0.57       813
         South_Asia       0.52      0.38      0.44       389
    Southeast_Asia       0.59      0.42      0.49       818
    Western_Europe       0.58      0.80      0.67      2383

          accuracy                           0.61     10000
         macro avg       0.62      0.56      0.58     10000
      weighted avg       0.61      0.61      0.60     10000
```

Figure 11: Model 2 results (no leakage)

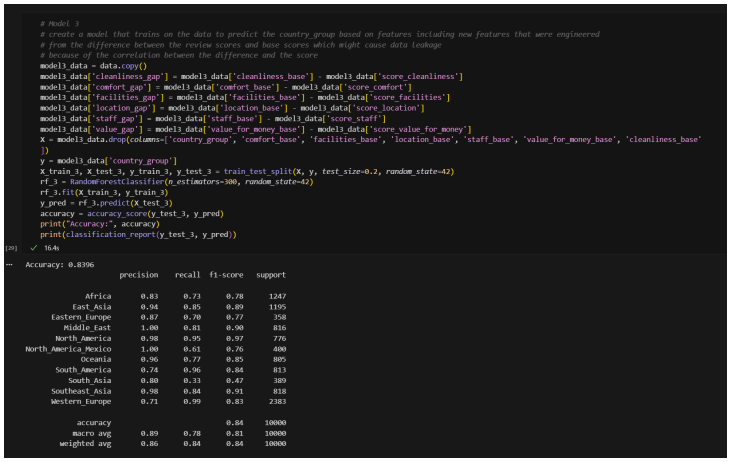
Accuracy: 61%

Conclusion: By averaging base features, unique identifiers were removed, improving generalization. This allowed the model to group similar records and leverage other meaningful features for better predictive accuracy.

3.3.3 Model 3: With Partial Data Leakage

Approach: Base hotel features created a 1:1 mapping with predictions, causing data leakage. To explore this, derived features were generated indirectly from existing review and hotel data without using raw base columns directly.

Results:



```
# Model 3
# create a model that trains on the data to predict the country_group based on features including new features that were engineered
# from the difference between the review scores and base scores which might cause data leakage
# because of the correlation between the difference and the score
model3_data = data.copy()
model3_data['cleanliness_gap'] = model3_data['cleanliness_base'] - model3_data['score_cleanliness']
model3_data['comfort_gap'] = model3_data['comfort_base'] - model3_data['score_comfort']
model3_data['facilities_gap'] = model3_data['facilities_base'] - model3_data['score_facilities']
model3_data['location_gap'] = model3_data['location_base'] - model3_data['score_location']
model3_data['staff_gap'] = model3_data['staff_base'] - model3_data['score_staff']
model3_data['value_gap'] = model3_data['value_for_money_base'] - model3_data['score_value_for_money']
X = model3_data.drop(columns=['country_group', 'comfort_base', 'facilities_base', 'location_base', 'staff_base', 'value_for_money_base', 'cleanliness_base'
])
y = model3_data['country_group']
X_train_3, X_test_3, y_train_3, y_test_3 = train_test_split(X, y, test_size=0.2, random_state=42)
rf_3 = RandomForestClassifier(n_estimators=300, random_state=42)
rf_3.fit(X_train_3, y_train_3)
y_pred = rf_3.predict(X_test_3)
accuracy = accuracy_score(y_test_3, y_pred)
print("Accuracy:", accuracy)
print(classification_report(y_test_3, y_pred))
```

[29] ✓ 164s

```
Accuracy: 0.8396
                      precision    recall  f1-score   support

              Africa       0.83      0.73      0.78      1247
           East_Asia       0.94      0.85      0.89      1195
      Eastern_Europe       0.87      0.70      0.77       358
         Middle_East       1.00      0.81      0.90       816
       North_America       0.98      0.95      0.97       776
North_America_Mexico       1.00      0.61      0.76       400
             Oceania       0.96      0.77      0.85       805
       South_America       0.74      0.96      0.84       813
          South_Asia       0.80      0.33      0.47       389
     Southeast_Asia       0.98      0.84      0.91       818
     Western_Europe       0.71      0.99      0.83      2383

           accuracy                           0.84     10000
          macro avg       0.89      0.78      0.81     10000
       weighted avg       0.86      0.84      0.84     10000
```

Figure 12: Model 3 results (partial leakage)

Accuracy: 84%

Conclusion: Although performance improved substantially, the derived features introduced a form of *mathematical leakage*. Since each base value could be reconstructed using the relation:

$$\text{Base Feature} = \text{Derived Feature} + \text{Review Score}$$

this model unintentionally encoded true target information.

3.4 Models' Performance Comparison: Train vs Test Accuracy

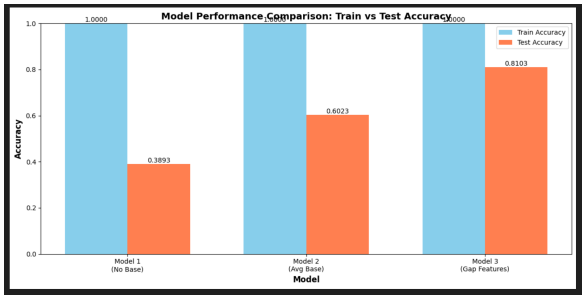


Figure 13: Models' Accuracy Comparison

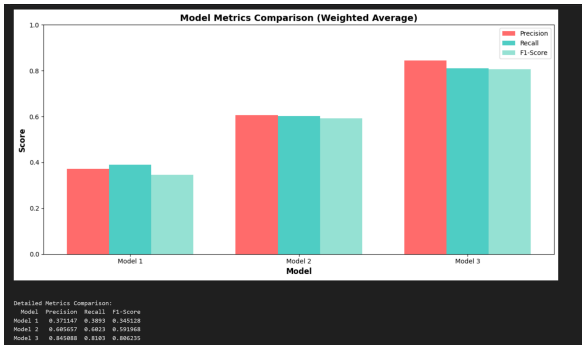


Figure 14: Models' Detailed Metrics Comparison

3.5 Model Explainability

To interpret and validate the Random Forest model's predictions, we applied two popular explainability techniques: **LIME (Local Interpretable Model-Agnostic Explanations)** and **SHAP (SHapley Additive exPlanations)**.

3.6 First Model

3.6.1 LIME Analysis

LIME explains predictions for the first record.

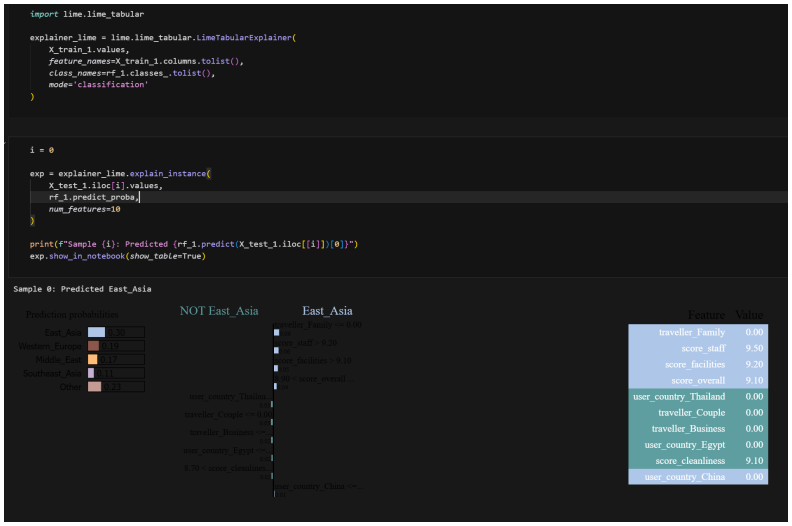


Figure 15: LIME visualization showing local feature contributions for a sample prediction.

The LIME analysis conducted on the first record revealed that the features **traveller_type**, **score_staff** > 9.2, **score_facilities** > 9.1, and **score_overall** > 8.9 were the most influential factors driving the model's prediction of *East_Asia*.

3.6.2 SHAP Analysis

SHAP (SHapley Additive exPlanations) provides both **global** and **local** interpretability, offering insights into how each feature contributes to model predictions at both the dataset and individual observation levels.

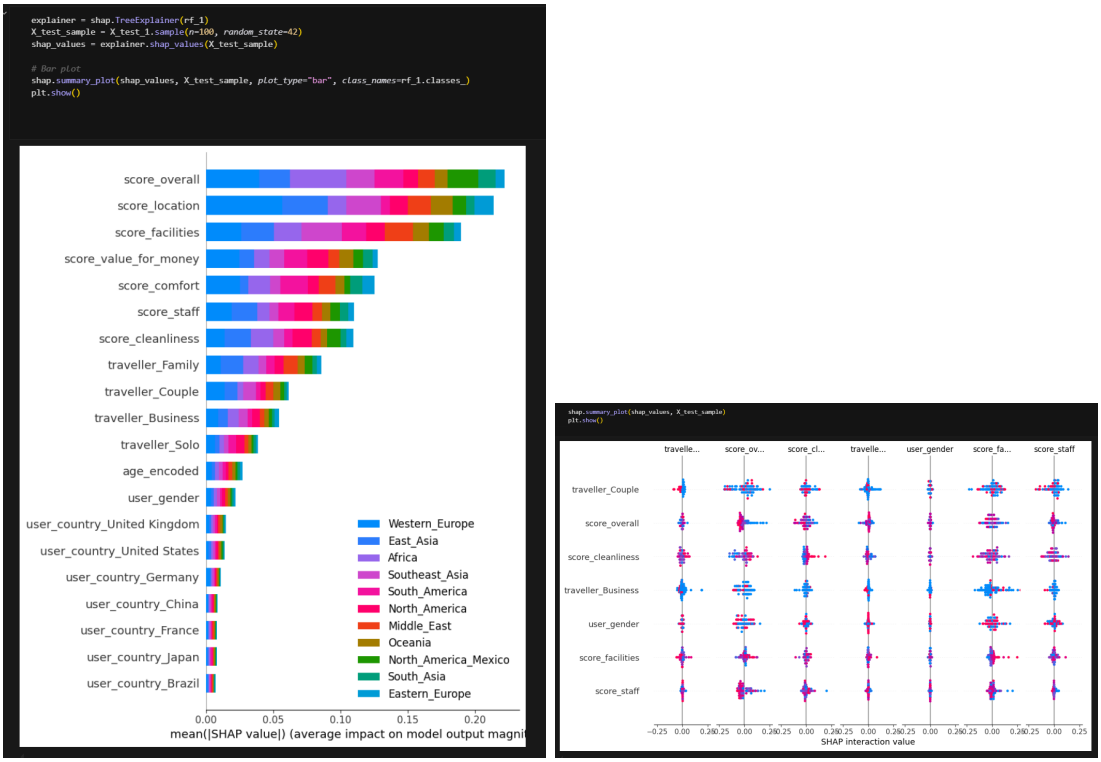


Figure 16: SHAP summary (left) and bar (right) plots illustrating global feature importance across the dataset.

Global Feature Importance On a global scale, SHAP results indicate that the **review score-related features** (e.g., *score_overall*, *score_staff*, *score_facilities*) had the greatest influence on the model's predictions, while the *user_country* feature contributed the least to the overall classification outcomes.

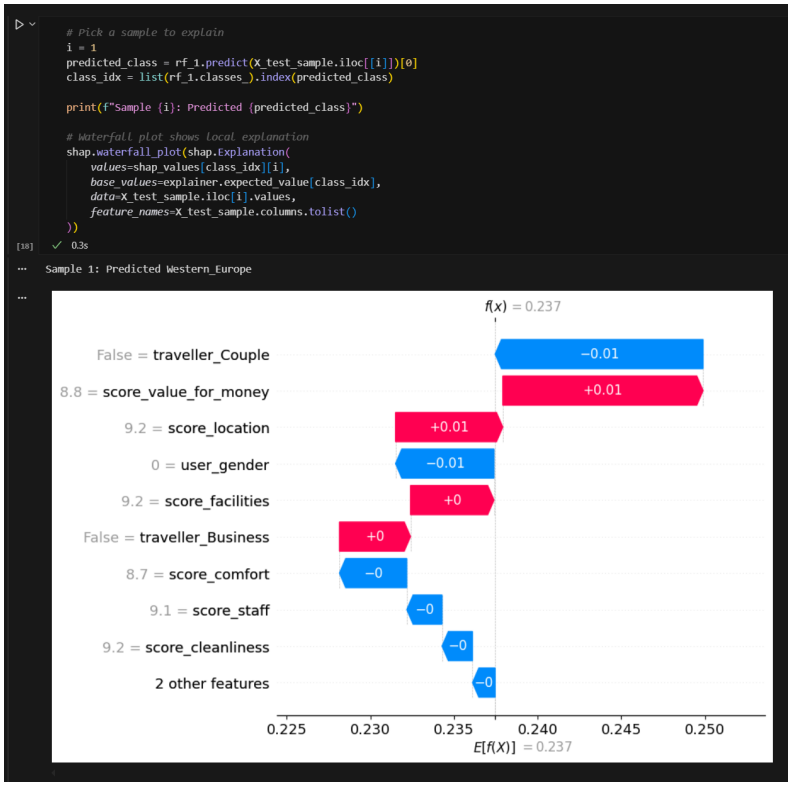


Figure 17: SHAP force plot for an individual prediction showing positive and negative feature contributions.

Local Explanation on the Second Record At the local level, the SHAP explanation for the **second record** highlights that *score_overall*, *age_encoded*, and *user_gender* were the dominant factors leading the model to predict the *Western_Europe* class. These features exhibited the highest absolute SHAP values, signifying their strong contribution to the individual prediction.

3.7 Second Model

3.7.1 LIME Analysis

LIME explains predictions for the fifth record.

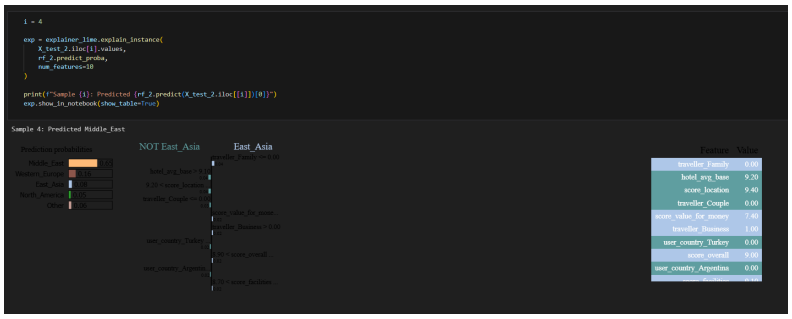


Figure 18: LIME visualization showing local feature contributions for a sample prediction.

The LIME analysis conducted on the fifth record revealed that the features **traveller_type**, **score_value_for_money**, **score_overall**, and **score_facilities** played the

largest roles (with the highest weights in the final prediction), leading to the final model prediction of *Middle_East*.

3.7.2 SHAP Analysis

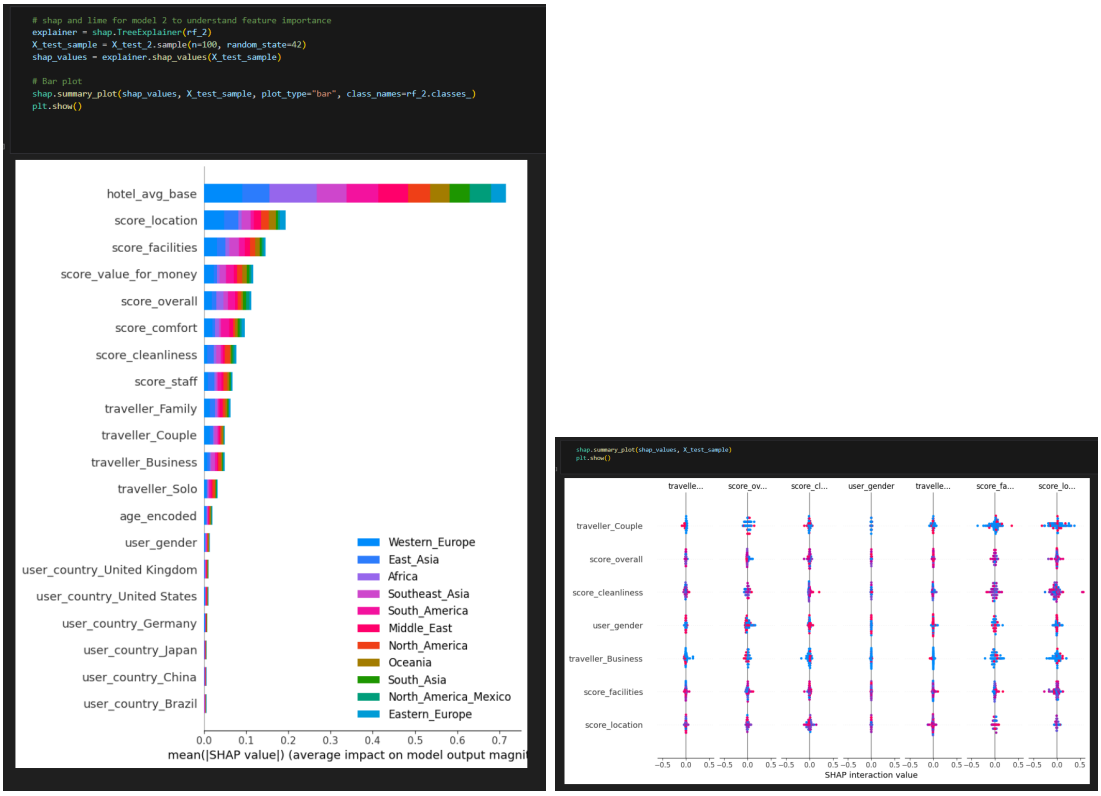


Figure 19: SHAP summary (left) and bar (right) plots illustrating global feature importance across the dataset.

Global Feature Importance On a global scale, SHAP results indicate that the **hotel_avg_base** had the greatest influence on the model's predictions, while the *user_country* feature contributed the least to the overall classification outcomes.

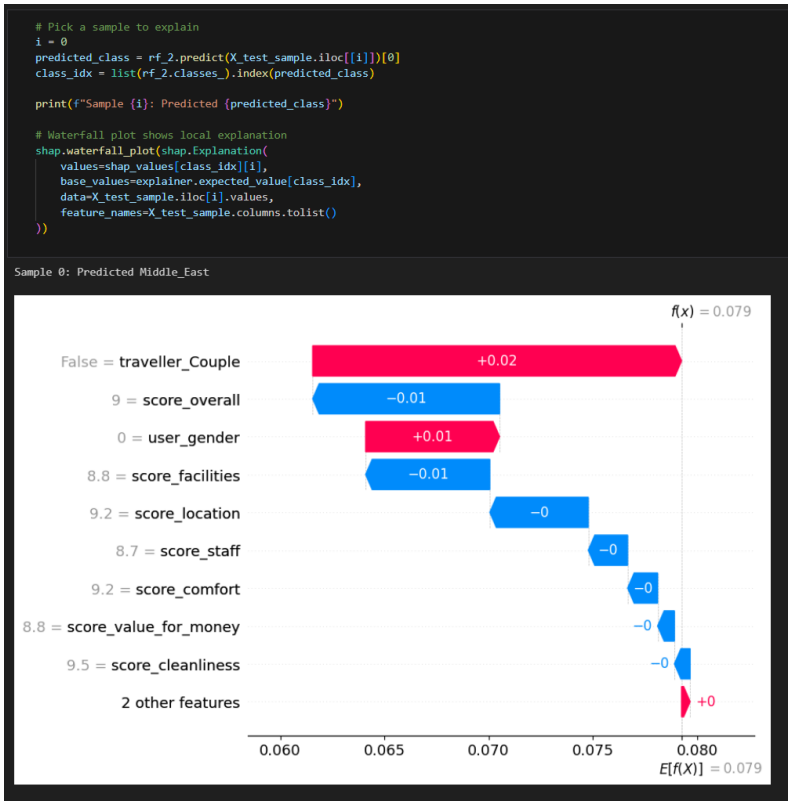


Figure 20: SHAP force plot for an individual prediction showing positive and negative feature contributions.

Local Explanation on the First Record At the local level, the SHAP explanation for the **first record** highlights that *traveller_type*, *user_gender*, and two other features were the dominant factors leading the model to predict the *Middle_East* class. These features exhibited the highest absolute SHAP values, signifying their strong contribution to the individual prediction.

3.8 Third Model

3.8.1 LIME Analysis

LIME explains predictions for the fifth record.

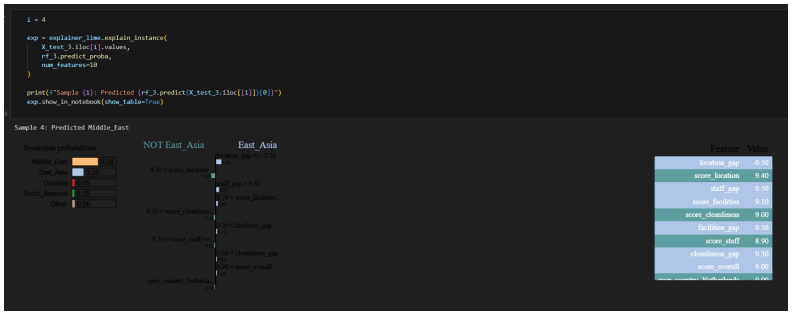


Figure 21: LIME visualization showing local feature contributions for a sample prediction.

The LIME analysis conducted on the first record revealed that the features **location_gap**, **staff_gap**, **score_facilities** $>$ **8.7**, and **facilities_gap** $>$ **0.2**, among two

additional features, were the most influential factors driving the model's prediction of *Middle_East*.

3.8.2 SHAP Analysis

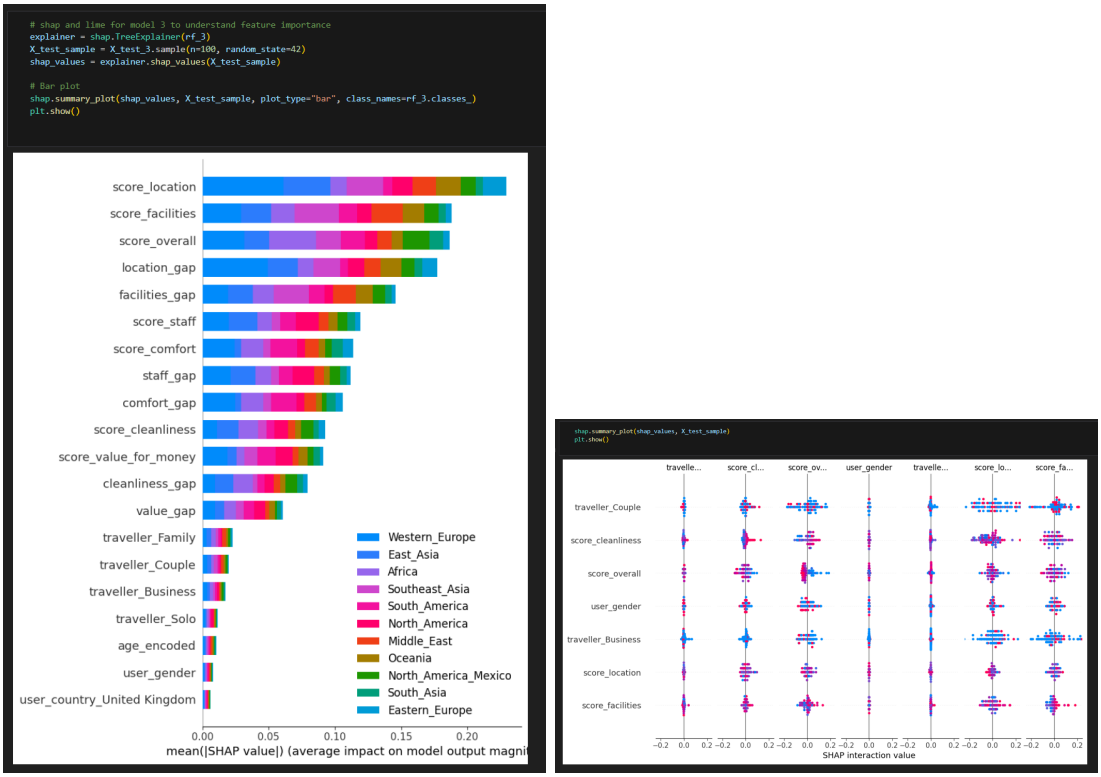


Figure 22: SHAP summary (left) and bar (right) plots illustrating global feature importance across the dataset.

Global Feature Importance On a global scale, SHAP results indicate that the **review score-related features** (e.g., *score_overall*, *score_staff*, *score_facilities*) had the greatest influence on the model's predictions, while the *user_country* feature contributed the least to the overall classification outcomes.

Figure 23: SHAP force plot for an individual prediction showing positive and negative feature contributions.

Local Explanation on the First Record At the local level, the SHAP explanation for the **first record** highlights that *score_overall*, *score_staff*, *age_encoded*, and *user_gender* were the dominant factors leading the model to predict the *Oceania* class. These features exhibited the highest absolute SHAP values, signifying their strong contribution to the individual prediction.

References

[1] A. Myung, "International hotel booking analytics dataset." `https://www.kaggle.com/datasets/alperenmyung/international-hotel-booking-analytics/data`, 2024. Accessed: 2025-10-24.