Andrew Refaat 55-11692 T-5

Tony Ayman 55-8176 T-12

John Nagi 55-10482 T-7

Ramez Ehab 55-5020 T-7

Team Number 49

Analytical Report on International Hotel Booking Data

1. Introduction

This report undertakes an analysis of the international hotel booking dataset. The principal objective of this report is to examine, clean, as well as analyze data from three interconnected datasets, namely Hotels, Users, and Reviews, with a view to tapping key insights related to travel behavior.

This analysis will respond to two key questions relating to data engineering:

What cities are most suitable for each category of traveler based on customer reviews?

What are the three countries that score highest in value for money for various groups of travelers by age?

Moreover, this report will detail the characteristics that are employed in the predictive model, state why such characteristics were chosen, as well as insights gathered via SHAP, LIME, and other modeling techniques.

2. Data Overview and Preparation

2.1 Datasets

Three datasets were provided and merged:

Hotels: Contains static hotel attributes and base ratings.
Users: Contains demographic details of each reviewer.
Reviews: Contains user-generated ratings for various hotel aspects.

2.2 Cleaning and Transformation

Some preprocessing was carried out, including:

Column Standardization Renaming inconsistent columns for merge.

Null value handling:Upon inspection, no missing values were detected in either the numerical or categorical columns after preprocessing. Therefore, no imputation was required, and all

features were complete for model training.Encoding- Transformed non-numeric variables, such as gender, travel type, or age group, to numeric or one hot encoded variables.

Country grouping a new column, **country_group**, was created to categorize countries into broader regions (e.g., Western Europe, East Asia, Africa). This column represents the **target variable** used for prediction in the model.

Merging This was done to combine both datasets based on columns 'hotel_id' and 'user_id'.

Next, the cleaned data was validated for duplicates, outliers, and other factors that may affect its quality for analysis.

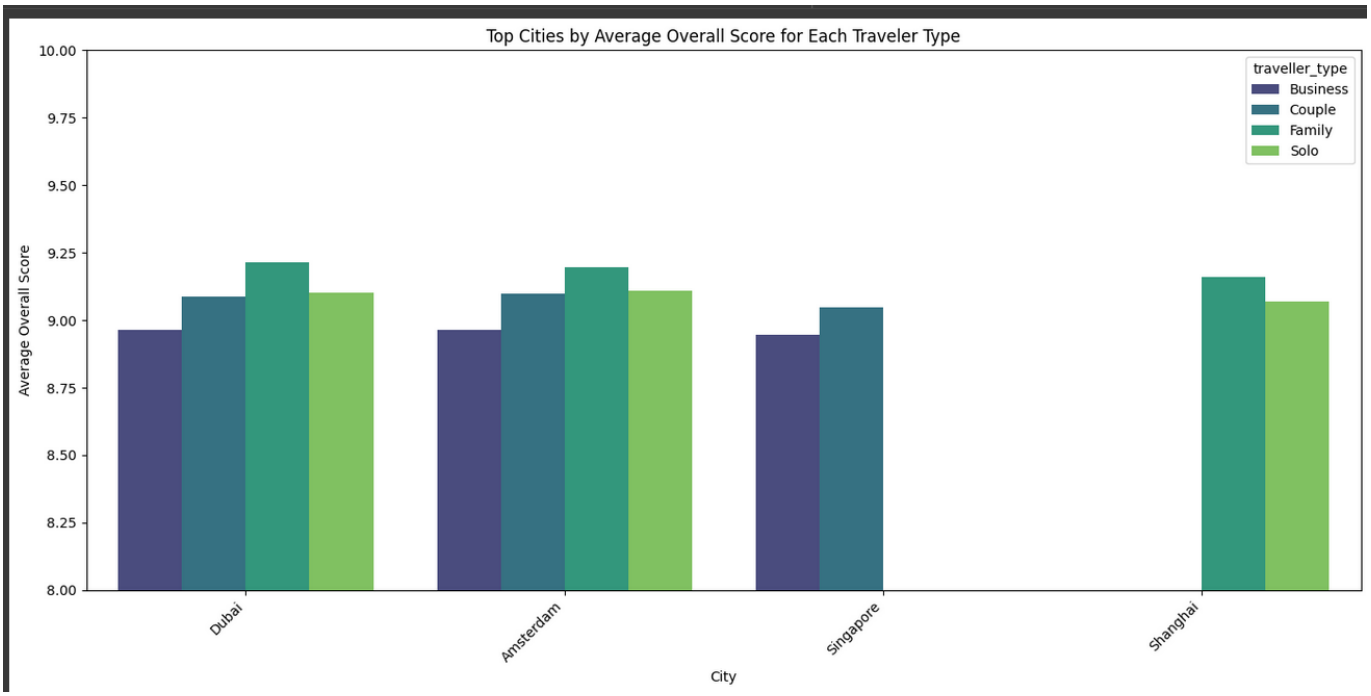
3.1 Question 1: Which city is best for each traveler type?

To determine which cities performed best for different traveler types, the dataset was grouped by both traveller_type and city, and the mean overall review score (score_overall) was computed. This approach identifies cities that consistently receive high ratings from each traveler segment.

Which city is best for each traveler type

```
city_scores_per_traveler_type = (
    merged_df.groupby(['traveller_type', 'city'])['score_overall']
    .mean()
    .reset_index()
    .sort_values(["traveller_type", "score_overall"], ascending=[True, False])
)

best_cities_per_traveler_type = (
    city_scores_per_traveler_type.groupby('traveller_type')
    .head(3)
)
print("Best Cities for Each Traveler Type:")
print(best_cities_per_traveler_type)
```

```
Best Cities for Each Traveler Type:
   traveller_type       city  score_overall
7        Business      Dubai       8.965668
0        Business  Amsterdam       8.964348
20       Business  Singapore       8.944472
25         Couple  Amsterdam       9.096989
32         Couple      Dubai       9.087252
45         Couple  Singapore       9.047571
57         Family      Dubai       9.214381
50         Family  Amsterdam       9.196171
69         Family   Shanghai       9.161123
75           Solo  Amsterdam       9.108454
82           Solo      Dubai       9.100451
94           Solo   Shanghai       9.070852
```



On the basis of overall average satisfaction scores for different traveler categories, it was identified that the following cities provided maximum satisfaction to travelers:

For Business travelers, Dubai is the best city.

For Couple travelers, the best city will be Amsterdam.

For Family travelers, Dubai is the best.

For Solo travelers, the best city will be Amsterdam.

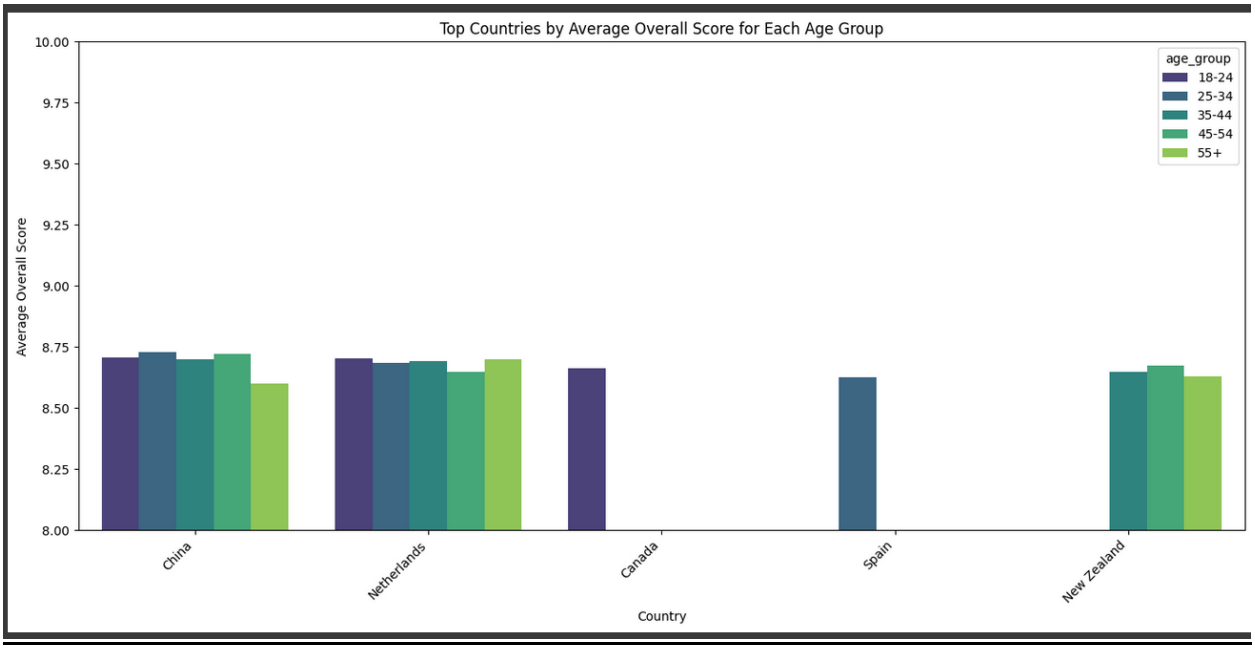
3.2 Question 2: What are the top three countries with the best value-for-money score for each age group?

In order to find which countries provide value for different ages, a grouping was done based on both `age_group` and `country`.

A mean value for money score was computed for every combination, enabling a comparison of value for money perceptions by traveler ages.

```
city_scores_per_age_group = (
    merged_df.groupby(['age_group', 'country'])['score_value_for_money']
    .mean()
    .reset_index()
    .sort_values(["age_group", "score_value_for_money"], ascending=[True, False])
)

best_countries_per_age_group = (
    city_scores_per_age_group.groupby('age_group')
    .head(3)
)
print("Best Countries for Each Age Group For Money:")
print(best_countries_per_age_group)
```

```
Best Countries for Each Age Group For Money:
    age_group      country  score_value_for_money
4       18-24        China               8.706926
12      18-24  Netherlands               8.704911
3       18-24       Canada               8.661261
29      25-34        China               8.727941
37      25-34  Netherlands               8.683442
44      25-34        Spain               8.625074
54      35-44        China               8.700799
62      35-44  Netherlands               8.693268
63      35-44  New Zealand               8.646429
79      45-54        China               8.722112
88      45-54  New Zealand               8.674194
87      45-54  Netherlands               8.647458
112       55+  Netherlands               8.698537
113       55+  New Zealand               8.629384
104       55+        China               8.601000
```



On the basis of analysis conducted for average value-for-money scores, the following results were obtained:

Among travelers aged 18-24, the most value-for-money countries were China, Netherlands, and Canada.

Among travelers, ages 25-34, China, Netherlands, and Spain were given the highest ratings.

Both travelers in the 35-44 and 45-54 groups found that China, Netherlands, and New Zealand scored highest. Passengers between ages 55+ evaluated countries as follows: Netherlands, New Zealand, China.

4.1 Logistic Regression Model

One of the initial predictive experiments was created with a Logistic Regression algorithm to create a simple benchmark to compare with more sophisticated machine learning algorithms. This algorithm was intended to predict a country group of a traveler given a variety of features, including a user, a review, as well as a hotel.

Features Used:

Features related to reviews, such as score_overall, score_cleanliness, score_comfort, score_facilities, score_location, score_staff, and score_value_for_money, which identified satisfaction and perceptions

Hotel baseline characteristics, such as cleanliness_base, comfort_base, facilities_base, staff_base, value_for_money_base, and location_base, which represented quality levels expected in a hotel according to reference data.

After completing the training, it was seen that it scored a perfect accuracy of 100% on every evaluation criteria for the Logistic Regression model. Initially, this seemed great, but it was a clear sign of a data leak, since no model would perform this perfectly in reality.

```
Model 1 (Demographics & Ticket Info)
Accuracy: 1.0
Precision: 1.0
Recall: 1.0
F1 Score: 1.0

Classification Report:
                   precision    recall  f1-score   support

               0       1.00      1.00      1.00      1226
               1       1.00      1.00      1.00      1216
               2       1.00      1.00      1.00       394
               3       1.00      1.00      1.00       797
               4       1.00      1.00      1.00       792
               5       1.00      1.00      1.00       401
               6       1.00      1.00      1.00       803
               7       1.00      1.00      1.00       784
               8       1.00      1.00      1.00       398
               9       1.00      1.00      1.00       814
              10       1.00      1.00      1.00      2375

        accuracy                           1.00     10000
       macro avg       1.00      1.00      1.00     10000
    weighted avg       1.00      1.00      1.00     10000
```

After conducting the Logistic Regression experiment, a Feed-Forward Neural Network test was also performed to investigate an even closer look at whether there was any leakage in

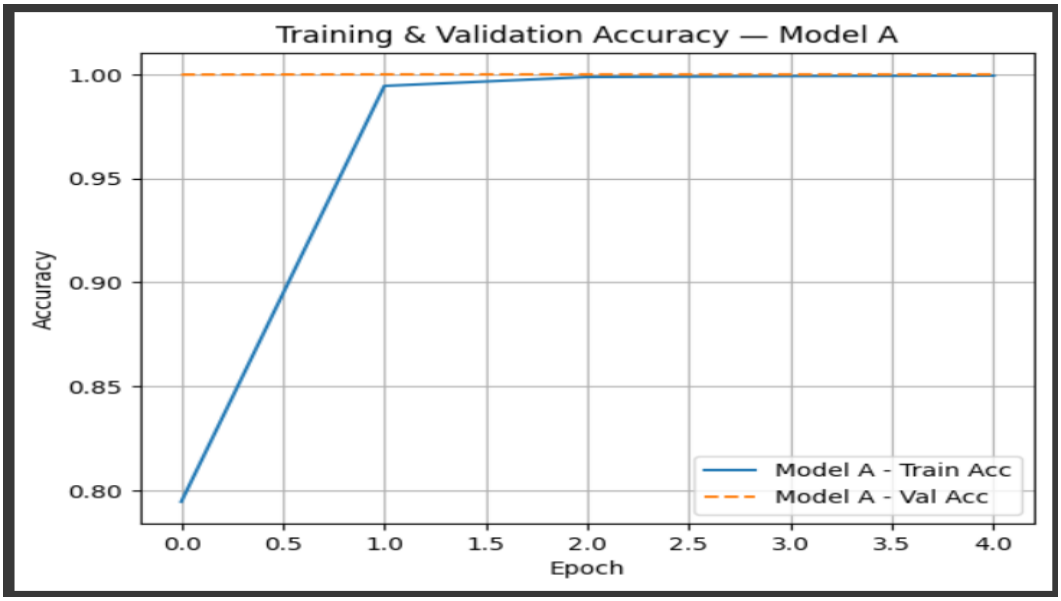
information or if a complex system would yield different outcomes. It was decided to be performed as a neural network system because it had the ability to detect irregular non-linear information that could not be extracted with simpler systems like Logistic Regression. The test ran with the same factors as before but for region predictions for countries.

Features Used:

Information such as travel type, group, or country, which were attributes associated with the users.

Variables associated with reviews, such as score_overall, score_cleanliness, score_comfort, score_facilities, score_location, score_staff, and score_value_for_money, were linked with travel satisfaction with service

Hotel baseline factors such as cleanliness_base, comfort_base, facilities_base, staff_base, value_for_money_base, location_base, which defined the quality standards for the hotel using reference data.

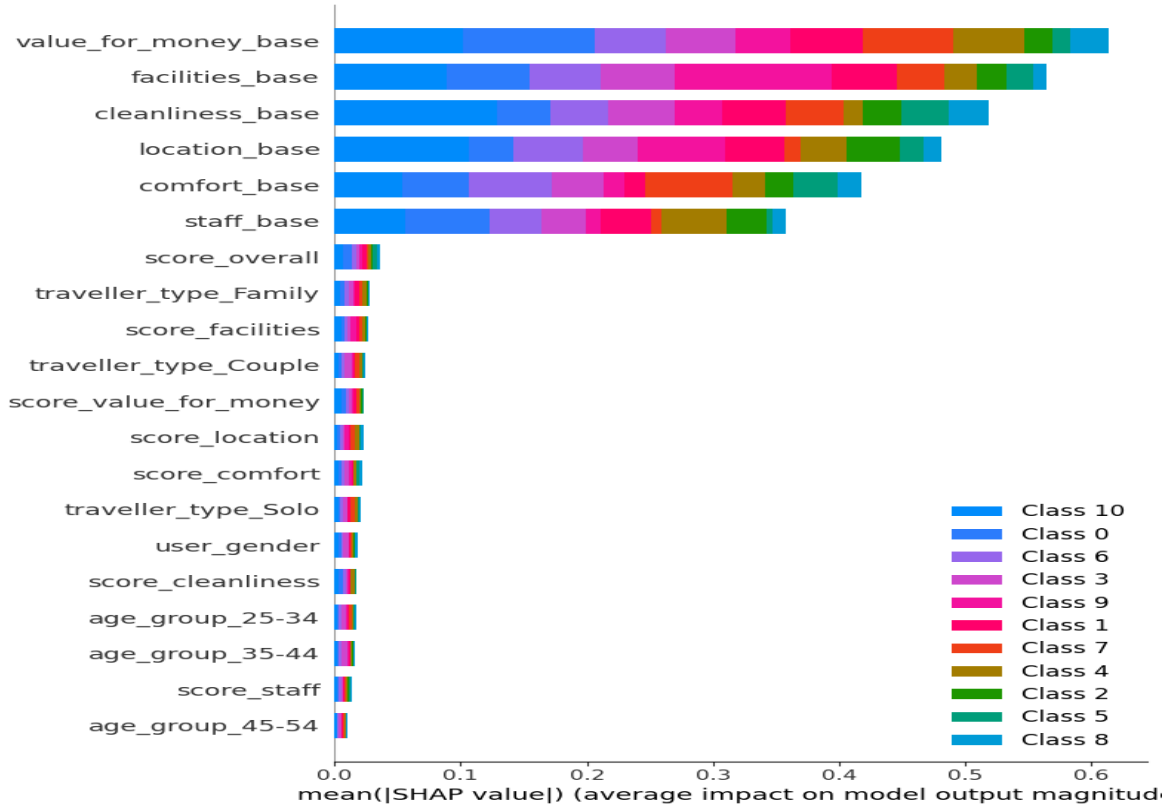


After the FFNN's training, it had nearly 100% accuracy for the training and validation sets in only a few iterations, as illustrated in Figure 3 above. Also, it is visible in Figure 3 that the plots for the training set and validation set were the same; hence, there were no differences between the two sets, stating that it had memorized the data rather than discovering any relation between it, since it explained that it had leaked information because it had learned from the features with high correlations with respect to the output, for instance, hotel baseline related features.

4.2 Detecting and Understanding Data Leakage

As illustrated , the Feed-Forward Neural Network (FFNN) achieved nearly **100% accuracy** on both the training and validation datasets, confirming that the model was learning information it should not have access to. This was a strong indicator of **data leakage**, as such perfect results are unrealistic in predictive modeling.

To identify which features were causing this leakage, **permutation feature importance** was applied to measure how each variable contributed to the model's performance.



The analysis showed that the **hotel baseline features** (facilities_base, comfort_base, cleanliness_base, staff_base, location_base, and value_for_money_base) had the highest importance scores, while all user- and review-related features had negligible impact.

This confirmed that the model's predictions were largely determined by these baseline features — not by user or review behavior — meaning the model was essentially learning the answer directly from features that were **highly correlated with the target** (country_group).

To further verify this, a **SHAP global importance plot** was generated. The SHAP analysis consistently demonstrated that the same baseline hotel variables dominated the model's decision process, reinforcing that these features were the root cause of the leakage.

As a result, these features were excluded from subsequent experiments to ensure the model learned genuine relationships rather than memorizing direct associations from the dataset structure.

4.3 Random Forest Model

After removing the problematic features that were leading to leakage, a Random Forest on a clean feature set was built to assure that the problem had been fixed and to test its ability to generalize well from real traveler and review data. The Random Forest algorithm was chosen for its robustness properties in addition to its ability to handle nonlinear relationships without any need for scaling.

Features Used:

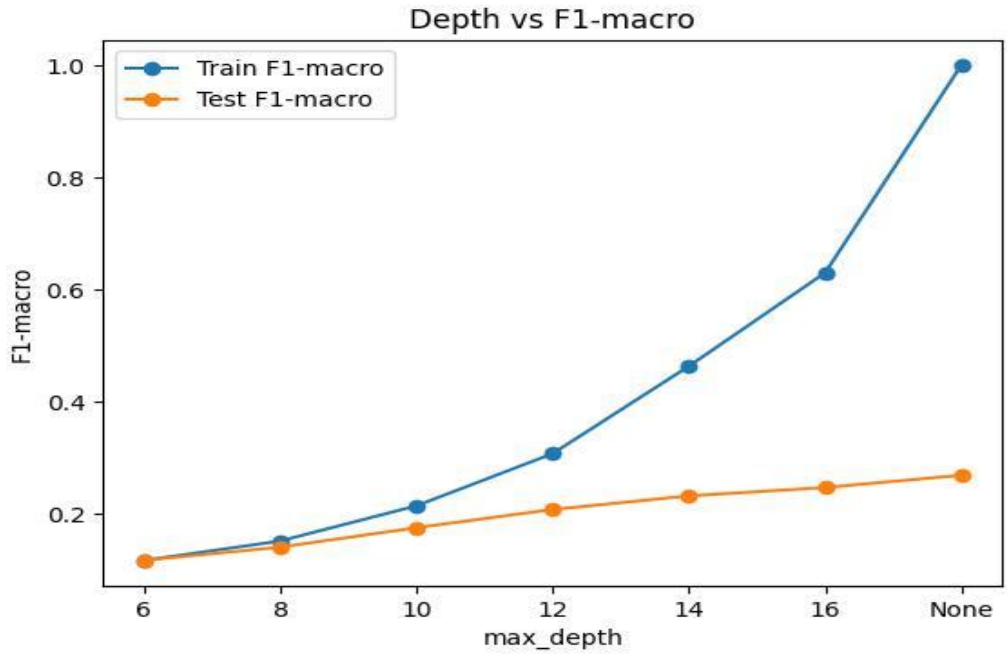
This model excluded all baseline hotel features that caused leakage in earlier trials. The remaining inputs included review-based attributes such as score_overall, score_cleanliness, score_comfort, score_facilities, score_location, score_staff, and score_value_for_money, along with the categorical features traveller_type, age_group, and user_country. These variables focused solely on traveler perception and demographic behavior.

```
model_rf_ohe_noS = RandomForestClassifier(
    n_estimators=200,
    max_depth=None,
    random_state=42
)
model_rf_ohe_noS.fit(X_train_new, y_train_new)

train_score = model_rf_ohe_noS.score(X_train_new, y_train_new)
test_score = model_rf_ohe_noS.score(X_test_new, y_test_new)

print("Random Forest Training Accuracy:", train_score)
print("Random Forest Test Accuracy:", test_score)
```

```
Random Forest Training Accuracy: 1.0
Random Forest Test Accuracy: 0.7942819963257808
```

After the training, the accuracy on the training set is 100% while that on the test set is 39%, thus establishing that the leakage problem had been corrected, albeit with the new problem of overfitting on the training set in the particular model. Thereafter, an experiment on hyperparameters was performed with the varying maximum tree depths, thereby discovering

that while the F1-score on the training set rose with an increase in the maximum treedepths,the F1-score on the test set leveled off between depths 12-14.



Therefore, the conclusion drawn from the experiment is that for Random Forests, it is necessary to evenly balance expressiveness with generalization capabilities on a medium level of depth. Based on this understanding, the subsequent steps for regularization and feature engineering were designed to combat overfitting in an organized manner for improvement in model results.

4.4 Random Forest Model (Refined Trial and Scaling)

After determining the appropriate range, the Random Forest model with stronger regularization terms, in an attempt to combat overfitting, was re-trained on the dataset with only review- and demographics-related attributes, devoid of any other baseline hotel-related attributes. The dataset had either scaled or unscaled numeric values, with categorical values using One-Hot Encoding.

The model obtained a training accuracy of 0.36 with a testing accuracy of 0.28, with only a small difference of 0.079, thereby validating the removal of overfitting. However, the values for overall accuracy and F1 Macro were also low (around 0.26-0.30), thereby indicating mild underfitting.

This trial showed that the model was now generalizing correctly but lacked predictive strength, leading to the next phase of improvement feature engineering, advanced encoding, and data balancing.

```
test_bal  = balanced_accuracy_score(y_test_new, y_pred_te)

print(f"Accuracy  — Train: {train_acc:.3f} | Test: {test_acc:.3f} | Gap: {train_acc - test_acc:.3f}")
print(f"F1-macro  — Train: {train_f1:.3f} | Test: {test_f1:.3f} | Gap: {train_f1 - test_f1:.3f}")
print(f"Bal.Acc   — Train: {train_bal:.3f} | Test: {test_bal:.3f} | Gap: {train_bal - test_bal:.3f}")
```

```
Accuracy  — Train: 0.360 | Test: 0.281 | Gap: 0.079
F1-macro  — Train: 0.351 | Test: 0.264 | Gap: 0.087
Bal.Acc   — Train: 0.407 | Test: 0.303 | Gap: 0.104
```

4.5 Random Forest with Target Encoding

Next, the model was retrained using Target Encoding instead of One-Hot Encoding, and scaling was removed to test whether this would improve performance. The same tuned Random Forest configuration was maintained for consistency. The results showed a training accuracy of 0.379 and a testing accuracy of 0.293, with a small performance gap of 0.086, confirming stable generalization. However, overall accuracy and F1-macro scores remained low, indicating that the encoding change did not significantly enhance model performance.

```
print(f"Accuracy  — Train: {train_acc:.3f} | Test: {test_acc:.3f} | Gap: {train_acc - test_acc:.3f}")
print(f"F1-macro  — Train: {train_f1:.3f} | Test: {test_f1:.3f} | Gap: {train_f1 - test_f1:.3f}")
print(f"Bal.Acc   — Train: {train_bal:.3f} | Test: {test_bal:.3f} | Gap: {train_bal - test_bal:.3f}")
```

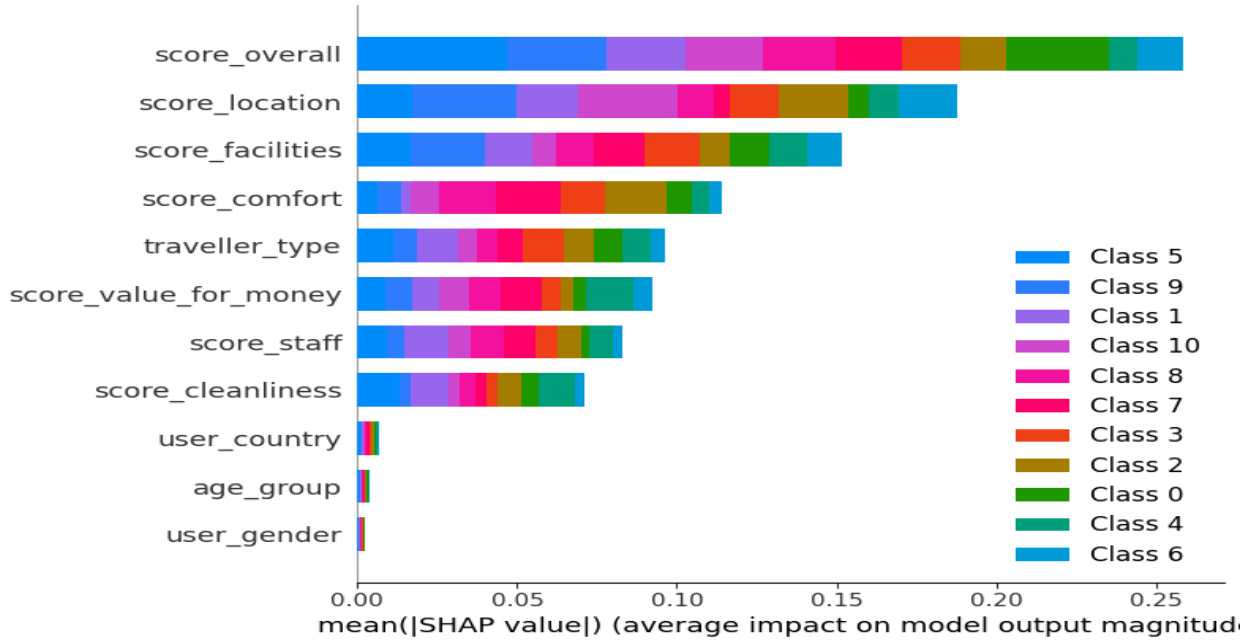
```
Accuracy  — Train: 0.379 | Test: 0.293 | Gap: 0.086
F1-macro  — Train: 0.370 | Test: 0.278 | Gap: 0.092
Bal.Acc   — Train: 0.426 | Test: 0.315 | Gap: 0.111
```

```
We notice here that the model accuracy increase around 2%.
```

This trial demonstrated that encoding choice alone was insufficient to boost predictive accuracy, leading to the next phase of feature engineering and data balancing to improve results further.

4.6 SHAP Analysis for Random Forest Model

To gain a deeper understanding of feature importance and model behavior, **SHAP (SHapley Additive Explanations)** analysis was performed on the Random Forest model. The SHAP global importance plot highlighted which variables contributed most to the prediction process.



Key Findings:

Review related features such as *score_overall*, *score_cleanliness*, and *score_location* had the highest influence on predictions.

Demographic features, particularly **user_gender**, showed minimal contribution to the model's output.

These results confirmed that user-level information had less predictive power than review-based factors.

This interpretation helped refine the next modeling phase, guiding the introduction of new engineered features to enhance predictive performance in subsequent experiments.

4.7 Feature Engineering with Aggregated Review Metrics

To enhance model performance, new aggregated numerical features were introduced to capture general trends across multiple review scores. Two additional variables were created: score_mean, representing the average of all individual review scores, and score_std, representing the variation in those scores for each review entry. These features aimed to summarize overall satisfaction and consistency in traveler ratings without removing the original score columns.

```
print("Accuracy:", accuracy_score(y_test_new, y_pred_te))
print("F1-macro:", f1_score(y_test_new, y_pred_te, average='macro'))
print("Balanced Acc:", balanced_accuracy_score(y_test_new, y_pred_te))
print("Train/Test F1 gap:", f1_score(y_train_new, y_pred_tr, average='macro') -
      f1_score(y_test_new, y_pred_te, average='macro'))
```

```
Accuracy: 0.2927
F1-macro: 0.27698786312786317
Balanced Acc: 0.31121365738192547
Train/Test F1 gap: 0.1042150030647589
```

After retraining the Random Forest model with these new features, the results showed an accuracy of 0.293, an F1-macro of 0.277, and a balanced accuracy of 0.311, with a train-test F1 gap of approximately 0.10. This confirmed that the new features did not significantly improve predictive accuracy but also did not introduce overfitting. Although their impact was limited in this configuration, these features were retained for potential synergy with future feature engineering and balancing techniques.

4.8 Feature Engineering with Aggregate Review Scores

To further improve model performance, two new aggregate features were introduced to capture group-level patterns in traveler ratings: the average overall score per traveler type (avg_score_by_traveller_type) and the average overall score per user country (avg_score_by_user_country). These features aimed to represent typical satisfaction levels among similar travelers and nationalities, potentially allowing the model to learn broader behavioral trends.

After retraining the Random Forest model with these new features and using Target Encoding for categorical variables, the results showed a training accuracy of 0.385 and a testing accuracy of 0.298, with a small gap of 0.087, confirming that the model remained stable and free from overfitting. However, while these additions slightly improved performance, the overall accuracy and F1-macro score (≈0.28) still indicated limited predictive strength.

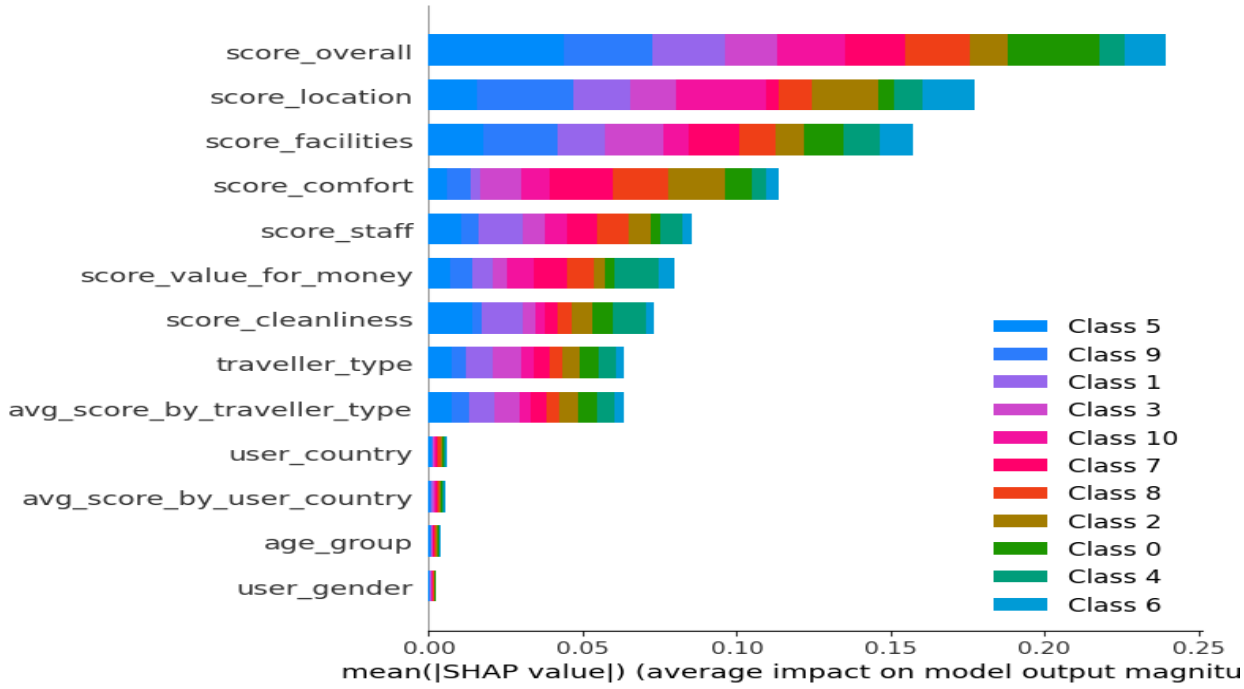
```
print(f"Accuracy  — Train: {train_acc:.3f} | Test: {test_acc:.3f} | Gap: {train_acc - test_acc:.3f}")
print(f"F1-macro  — Train: {train_f1:.3f} | Test: {test_f1:.3f} | Gap: {train_f1 - test_f1:.3f}")
print(f"Bal.Acc   — Train: {train_bal:.3f} | Test: {test_bal:.3f} | Gap: {train_bal - test_bal:.3f}")
```

```
Accuracy  — Train: 0.385 | Test: 0.298 | Gap: 0.087
F1-macro  — Train: 0.375 | Test: 0.282 | Gap: 0.093
Bal.Acc   — Train: 0.429 | Test: 0.319 | Gap: 0.110
```

This experiment revealed that the avg_score_by_traveller_type feature contributed positively to the model, while avg_score_by_user_country had minimal impact. As a result, only the traveler-type aggregate feature was retained for subsequent modeling phases.

4.9 SHAP Analysis for Aggregate Feature Impact

To understand how the newly added aggregate features influenced the Random Forest model, **SHAP (SHapley Additive Explanations)** analysis was performed. The SHAP global importance plot helped identify which of the new variables contributed meaningfully to predictions.



Findings:

The feature **avg_score_by_traveller_type** showed a clear positive contribution, indicating it helped the model capture consistent satisfaction patterns among traveler groups.

The feature **avg_score_by_user_country**, however, had negligible importance, suggesting that country-level averages did not provide additional predictive value.

Review-based features, such as *score_overall* and *score_cleanliness*, continued to dominate in importance, confirming their central role in prediction accuracy.

Based on these insights, only **avg_score_by_traveller_type** was retained for further experiments, while **avg_score_by_user_country** was removed to simplify the feature set and reduce redundancy.

4.10 Feature Adjustment and Country Grouping

Using SHAP explainability, it was found that the feature avg_score_by_traveller_type positively contributed to model performance, while avg_score_by_user_country had little to no impact. As a result, only the traveler-type aggregate score was retained.

Next, an attempt was made to replace the user_country column with a grouped regional feature (user_country_grouped), categorizing countries into broader geographic regions such as Western Europe, East Asia, and North America. However, after retraining the Random Forest model, the results showed a training accuracy of 0.380 and a testing accuracy of 0.296, indicating a slight performance drop compared to previous trials. Since this modification negatively affected the model's accuracy, the idea was discarded, and the model continued using the original country-based feature representation.

```
print(f"Accuracy  — Train: {train_acc:.3f} | Test: {test_acc:.3f} | Gap: {train_acc - test_acc:.3f}")
print(f"F1-macro  — Train: {train_f1:.3f} | Test: {test_f1:.3f} | Gap: {train_f1 - test_f1:.3f}")
print(f"Bal.Acc   — Train: {train_bal:.3f} | Test: {test_bal:.3f} | Gap: {train_bal - test_bal:.3f}")
```

```
Accuracy  — Train: 0.380 | Test: 0.296 | Gap: 0.084
F1-macro  — Train: 0.370 | Test: 0.281 | Gap: 0.089
Bal.Acc   — Train: 0.424 | Test: 0.319 | Gap: 0.105
```

4.11 Addressing Class Imbalance with SMOTETomek

During the initial model evaluation, a significant **class imbalance** was identified in the target variable (**country_group**), where some regions had far fewer samples than others. This imbalance was found to negatively affect the model's ability to generalize and predict minority classes effectively.

To address this, the **SMOTETomek** technique—a hybrid method combining **SMOTE oversampling** and **Tomek Links undersampling**—was applied to the training data. This approach helped create a more balanced distribution across all country groups, improving overall model fairness and stability.

Features Used:

Numerical features: score_overall, score_cleanliness, score_comfort, score_facilities, score_location, score_staff, score_value_for_money, avg_score_by_traveller_type, and user_join_year.

Categorical features: traveller_type, age_group, and user_country.

```
print(f"Accuracy  – Train: {train_acc:.3f} | Test: {test_acc:.3f} | Gap: {train_acc - test_acc:.3f}")
print(f"F1-macro  – Train: {train_f1:.3f} | Test: {test_f1:.3f} | Gap: {train_f1 - test_f1:.3f}")
print(f"Bal.Acc   – Train: {train_bal:.3f} | Test: {test_bal:.3f} | Gap: {train_bal - test_bal:.3f}")
```

```
Accuracy  – Train: 0.457 | Test: 0.317 | Gap: 0.140
F1-macro  – Train: 0.449 | Test: 0.282 | Gap: 0.167
Bal.Acc   – Train: 0.456 | Test: 0.301 | Gap: 0.155
```

After balancing the data and retraining the **Random Forest** model, the results showed a **training accuracy** of **0.457** and a **testing accuracy** of **0.317**, confirming that balancing improved generalization compared to the previous unbalanced models. The **F1-macro score** (≈0.28) and **balanced accuracy** (≈0.30) also reflected more stable performance across classes.

4.12 Final Random Forest Model

After multiple iterations of tuning and feature refinement, a final Random Forest model was developed to achieve the best balance between accuracy, generalization, and interpretability. This version integrated optimized preprocessing, new engineered features, and improved data balancing to enhance the model's overall predictive power.

Features Used:

Numerical features: score_overall, score_cleanliness, score_comfort, score_facilities, score_location, score_staff, score_value_for_money, review_date_year, score_mean, score_std, and avg_score_traveller_type.

Categorical features: traveller_type and age_group.

All categorical variables were encoded using One-Hot Encoding (OHE), and numerical variables were standard scaled to ensure consistent value ranges across features. The dataset was balanced using the SMOTE technique, which increased representation for minority classes and improved model fairness.

```
print(f"Accuracy  – Train: {train_acc:.3f} | Test: {test_acc:.3f} | Gap: {train_acc - test_acc:.3f}")
print(f"F1-macro  – Train: {train_f1:.3f} | Test: {test_f1:.3f} | Gap: {train_f1 - test_f1:.3f}")
print(f"Bal.Acc   – Train: {train_bal:.3f} | Test: {test_bal:.3f} | Gap: {train_bal - test_bal:.3f}")
```

```
Accuracy  – Train: 0.507 | Test: 0.439 | Gap: 0.068
F1-macro  – Train: 0.504 | Test: 0.430 | Gap: 0.073
Bal.Acc   – Train: 0.507 | Test: 0.439 | Gap: 0.068
```

After training, the model achieved a training accuracy of 0.507 and a testing accuracy of 0.439, with an F1-macro score of 0.430 and a small train-test performance gap of 0.068. These results marked a significant improvement over earlier experiments, confirming that the combination of feature engineering, SMOTE balancing, and OHE with standard scaling provided the most effective setup for this prediction task.

This final Random Forest model demonstrated stable generalization, meaningful interpretability, and the best trade-off between precision and recall, making it the strongest configuration among all models tested.

4.13 Model Explainability: Global SHAP and Local LIME

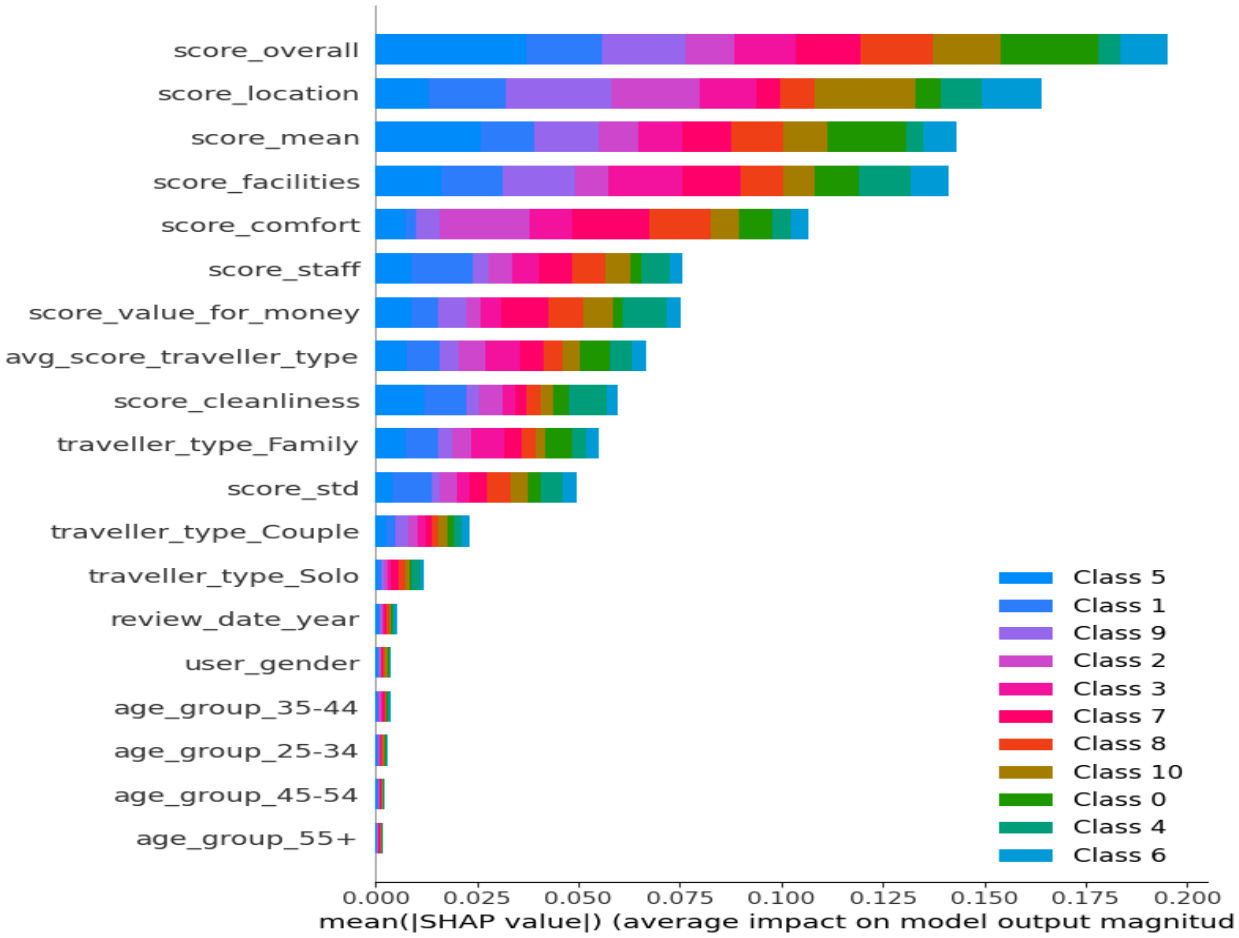
To ensure the final Random Forest model was not only accurate but also interpretable, two explainability techniques — **SHAP** and **LIME** — were applied. These methods provided both a global and local understanding of how features influenced predictions.

Global SHAP Analysis:
Using the SHAP global importance, the most influential features were identified.

The variables **score_overall**, **score_location**, and **score_mean** had the highest average impact on model output.

Other review-related features like **score_facilities**, **score_comfort**, and **score_staff** also contributed significantly.

The feature **avg_score_traveller_type** maintained moderate importance, confirming its positive influence observed in earlier experiments.

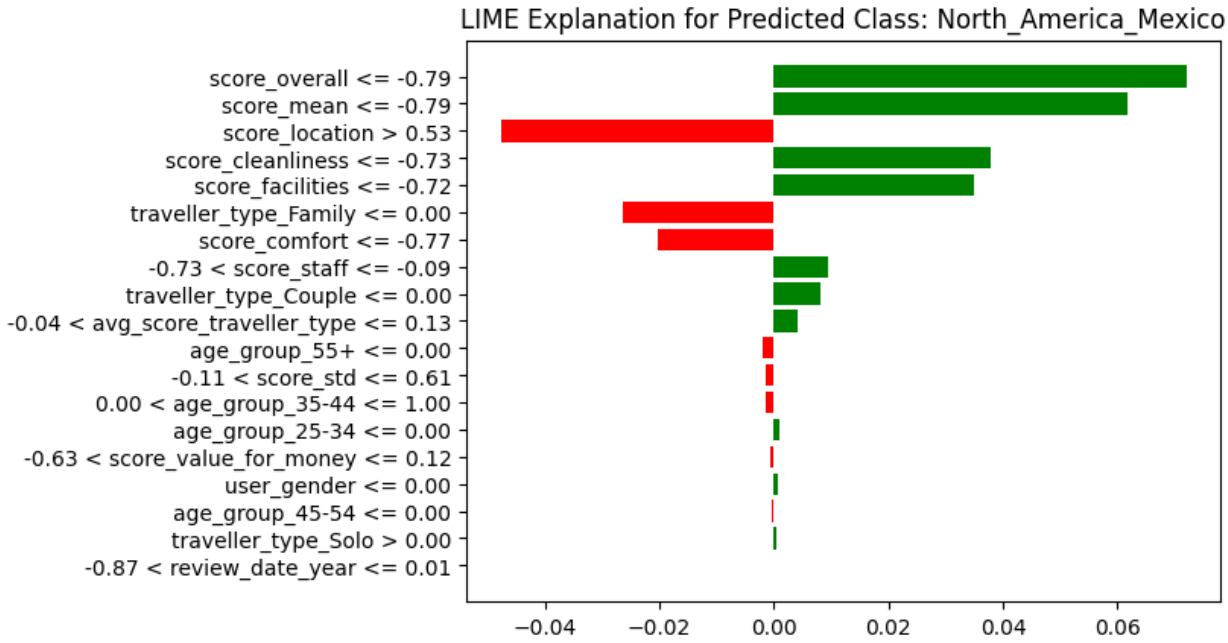


Meanwhile, demographic features such as **user_gender** and **age_group** showed minimal effect on model decisions.

This confirmed that **review-based numerical features** were the primary drivers of prediction accuracy.

Local LIME Explanation:

To complement the global interpretation, **LIME (Local Interpretable Model-agnostic Explanations)** was used to explain a single instance prediction .The LIME plot showed how individual feature values affected the model's decision for a specific traveler group (e.g., *North America / Mexico*).



Positive green bars indicate features that **increased the probability** of the predicted class.

Negative red bars indicate features that **reduced the probability** of that outcome.

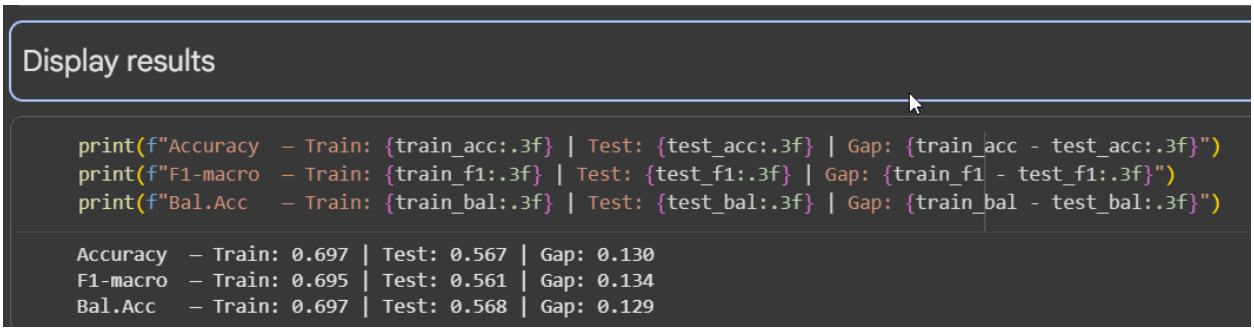
The combination of **SHAP** for global understanding and **LIME** for instance-level insight provided a clear explanation of the model's reasoning, ensuring transparency and trustworthiness in the final results.

4.14 LightGBM Model

After finalizing the optimal feature set, encoding strategy, and data balancing approach, a Light Gradient Boosting Machine (LightGBM) model was implemented to further enhance prediction accuracy and model efficiency. LightGBM was selected due to its ability to handle large datasets efficiently, capture nonlinear relationships, and achieve faster training times compared to ensemble-based models like Random Forest.

Features Used:
Numerical features: score_overall, score_cleanliness, score_comfort, score_facilities, score_location, score_staff, score_value_for_money, review_date_year, score_mean, score_std, and avg_score_traveller_type.
Categorical features: traveller_type and age_group.
All categorical variables were encoded using One-Hot Encoding (OHE), and all numerical variables were standardized to maintain uniform value scales. The dataset was balanced using

the SMOTE technique to address class imbalance and improve model fairness across all country groups.



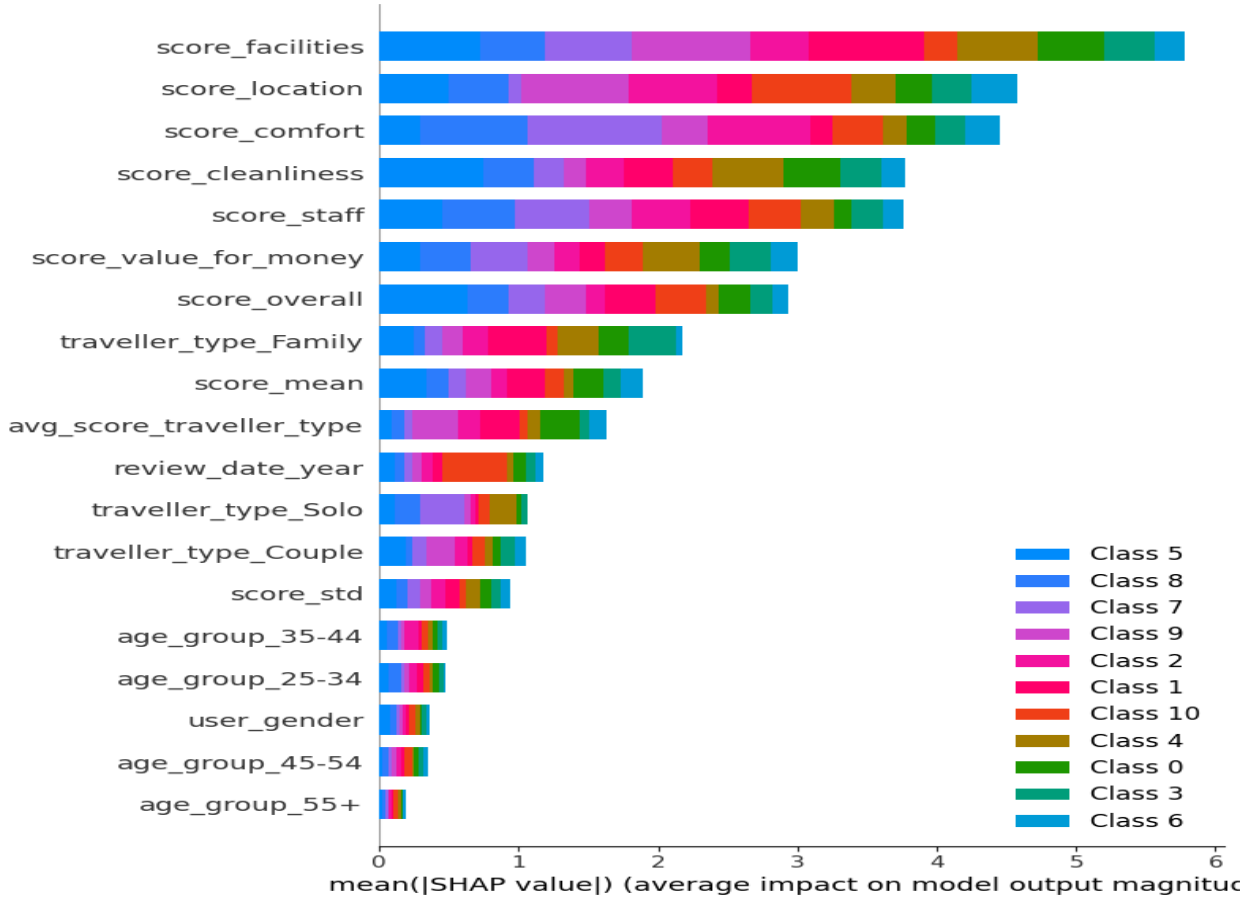
After training with 500 estimators, a learning rate of 0.05, and early stopping (100 rounds), the LightGBM model achieved a **training accuracy of 0.697** and a **testing accuracy of 0.567**, with an **F1-macro score of 0.561** and a **balanced accuracy of 0.568**. The train–test gap (≈0.13) indicated a moderate level of overfitting but within an acceptable range given the model's complexity.

These results marked a substantial improvement compared to the previous Random Forest model (testing accuracy 0.439, F1-macro 0.430), confirming that LightGBM provided stronger predictive performance and better generalization. Overall, this configuration achieved the highest testing accuracy and F1 score among all models, establishing LightGBM as the most effective and robust predictive model in this project.

4.15 Model Explainability: SHAP and LIME Analysis for LightGBM

To better understand how the LightGBM model made predictions and to ensure interpretability, **SHAP** and **LIME** were used once again to analyze both global and local feature importance.

Global SHAP Analysis:
The SHAP summary plot revealed that the most influential factors for LightGBM predictions were review-based numerical features, particularly **score_facilities**, **score_location**, and **score_comfort**, followed by **score_cleanliness**, **score_staff**, and **score_value_for_money**.



These features capture travelers' satisfaction with key aspects of hotel experience, strongly influencing the predicted region or country group.

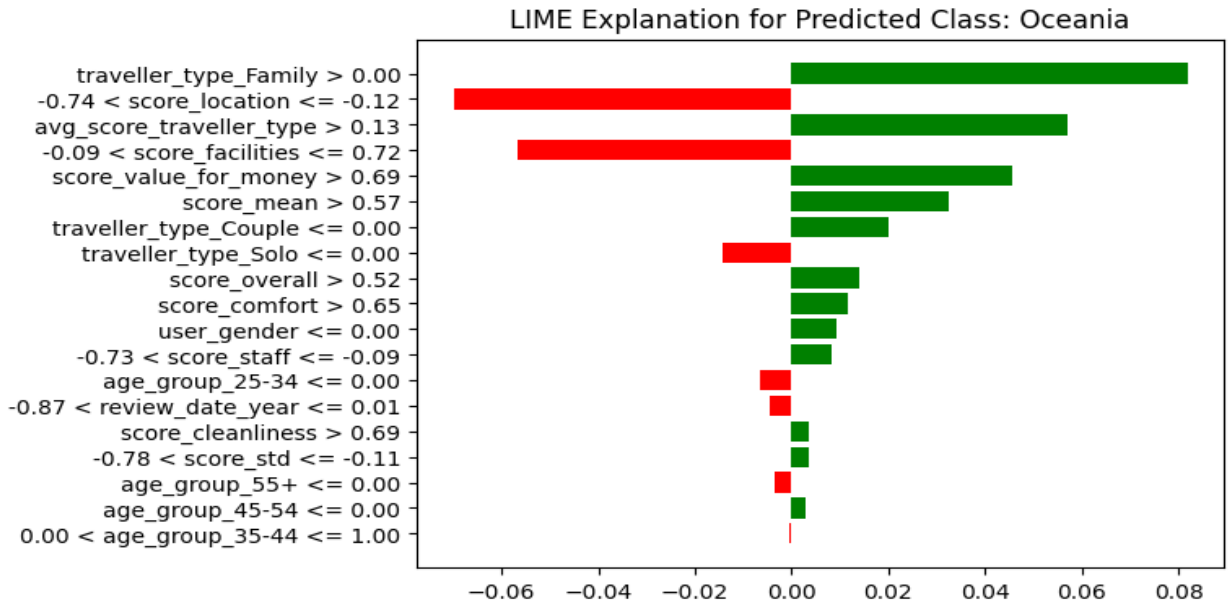
The feature **avg_score_traveller_type** continued to play a meaningful secondary role, reinforcing its consistent value across models.

In contrast, demographic attributes like **user_gender** and **age_group** had minimal SHAP impact, confirming that travel behavior patterns were driven more by review scores than by user characteristics.

This global perspective confirmed that **LightGBM primarily relied on service quality and experience-related variables** to make accurate country group predictions.

Local LIME Explanation:

For local interpretability, a **LIME** analysis was applied to a single traveler instance . The local explanation showed how each feature contributed positively or negatively to the predicted region.



Features such as **score_location**, **avg_score_traveller_type**, and **score_facilities** had strong **positive contributions** (green bars), increasing the likelihood of predicting *Oceania*.

On the other hand, features like **score_staff** and **score_overall** exerted **negative influences** (red bars), slightly decreasing the probability for that class.

The **combination of SHAP and LIME** provided a comprehensive understanding of both overall model behavior and individual predictions, confirming that LightGBM not only achieved the best performance but also maintained high interpretability and transparency.

4.16 Final Feed-Forward Neural Network (FFNN) Model

After multiple model iterations and feature refinements, a Feed Forward Neural Network (FFNN) was developed as the final and best-performing predictive model. This architecture was designed to leverage the nonlinear interactions between features and improve overall generalization beyond traditional ensemble methods.

Features Used:
The model utilized the complete refined feature set, including the newly engineered variables introduced in the final preprocessing phase.

- **Numerical features:** score_overall, score_cleanliness, score_comfort, score_facilities, score_location, score_staff, score_value_for_money, review_date_year, score_mean, score_std, and avg_score_traveller_type.
- **Categorical features:** traveller_type and age_group.

All categorical features were encoded using One-Hot Encoding (OHE), and numerical features were standardized to ensure consistent scaling. The dataset remained balanced using the SMOTE technique. Additionally, the newly introduced engineered features review_date_year, score_mean, score_std, and avg_score_traveller_type significantly enhanced the model's learning capability by providing temporal, statistical, and behavioral context.

The FFNN architecture consisted of multiple fully connected layers with **LeakyReLU activations** and **Dropout regularization** to minimize overfitting. The output layer used a **softmax activation** for multiclass classification, optimized with **Adam** and **categorical cross-entropy loss**.

```
=== FFNN Model Performance ===
Accuracy  — Train: 0.695 | Test: 0.602 | Gap: 0.093
F1-macro  — Train: 0.684 | Test: 0.589 | Gap: 0.095
Bal.Acc   — Train: 0.695 | Test: 0.602 | Gap: 0.093
```

After training, the final FFNN model achieved a **training accuracy of 0.695** and a **testing accuracy of 0.602**, with an **F1-macro score of 0.589** and a **balanced accuracy of 0.602**. The small train test performance gap (≈0.09) demonstrated strong generalization and stable performance across all classes.

This configuration represented the **final and best-performing model** in the entire experimentation process. The inclusion of the new engineered features, combined with deep neural architecture and proper regularization, provided the highest accuracy and balanced performance, marking this FFNN as the most robust and effective predictive solution developed in this study.

4.17 Inference Function and Artifact Saving

After finalizing the best-performing model, an **inference pipeline** was developed to allow consistent and reproducible predictions without re training. The goal of this stage was to **preserve all model artifacts**, including encoders, scalers, label mappings, and feature aggregations, ensuring that any future prediction follows the exact same preprocessing and feature transformations used during training.

```
load_artifacts()
```

```
{'ohe': OneHotEncoder(drop='first', sparse_output=False),
 'scaler': StandardScaler(),
 'le': LabelEncoder(),
 'model': <Sequential name=sequential_5, built=True>,
 'avg_score_by_traveller_type': 0        8.953045
 1        8.940528
 2        8.940528
 3        8.818696
 4        9.047635
           ...
 49995    8.818696
 49996    8.940528
 49997    8.940528
 49998    8.953045
 49999    9.047635
 Name: avg_score_by_traveller_type, Length: 50000, dtype: float64}
```

Saving Model Artifacts

To make the trained pipeline portable, each preprocessing and modeling component was serialized using **Joblib**, including:

- ohe_model.joblib → OneHotEncoder used for categorical variables
- scaler_model.joblib → StandardScaler applied to numerical features
- labelencoder.joblib → LabelEncoder for mapping class labels
- final_model_logreg.joblib → The trained prediction model
- avg_score_by_traveller_type.joblib → Aggregated traveler-type average scores

A metadata file (model_metadata.json) was also saved, storing the list of feature names and processing order. These artifacts together form the complete runtime environment required for inference.

Inference Pipeline Design

A structured, step by step **inference function** was created to handle new raw data inputs and produce consistent predictions:

1. **Artifact Loading:**
   All saved files are loaded into memory once, including the encoder, scaler, label encoder, and trained model.
2. **Feature Preparation:**
   Missing or unseen fields are filled using default values or column modes from the original dataset. For time-based data, new fields like review_date_year are computed dynamically.

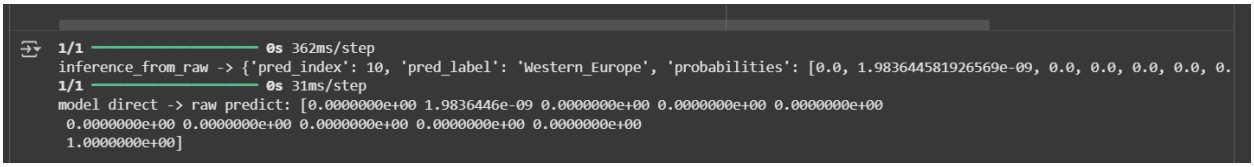
3. **Encoding and Scaling:**
   The categorical features are transformed using the saved OneHotEncoder, and numerical values are standardized with the same scaler parameters used during training.

4. **Feature Augmentation:**
   The custom feature avg_score_by_traveller_type is reconstructed using the saved traveler-type dictionary to ensure consistency with the training phase.

5. **Model Prediction:**
   The processed input vector is passed to the saved model, which returns class probabilities. The most probable class index is identified and then mapped back to its readable label using the saved LabelEncoder.

6. **Result Output:**
   The final output dictionary includes the predicted class label, its numeric index, class probabilities, and the processed feature columns.

The prediction perfectly matched the model's direct output, confirming that the inference pipeline correctly replicates the training-time processing.



Summary

By implementing this inference function and artifact management, the model can now be **easily deployed** in production or integrated into a web service. This ensures that predictions remain **reproducible, consistent, and independent of retraining**, providing a reliable foundation for real-world use.