Milestone 1 report

Team 57

DATA ANALYSIS AND CLEANING

The first step was to join the provided data to create an all encompassing dataframe to deal with all the data at once. The data was inspected for NAN or missing values using multiple functions like :

1. df.head()
2. df.info()
3. df.describe()
4. df.isnull().sum()

No missing values were found.
The dataframe was checked for duplicate rows, none were found.

DATA ENGINEERING

Q1: Best city per traveller type

Family	Dubai
Solo	Amsterdam
Couple	Amsterdam
Business	Dubai

Q2: Best value for money countries per age group

18-24	China

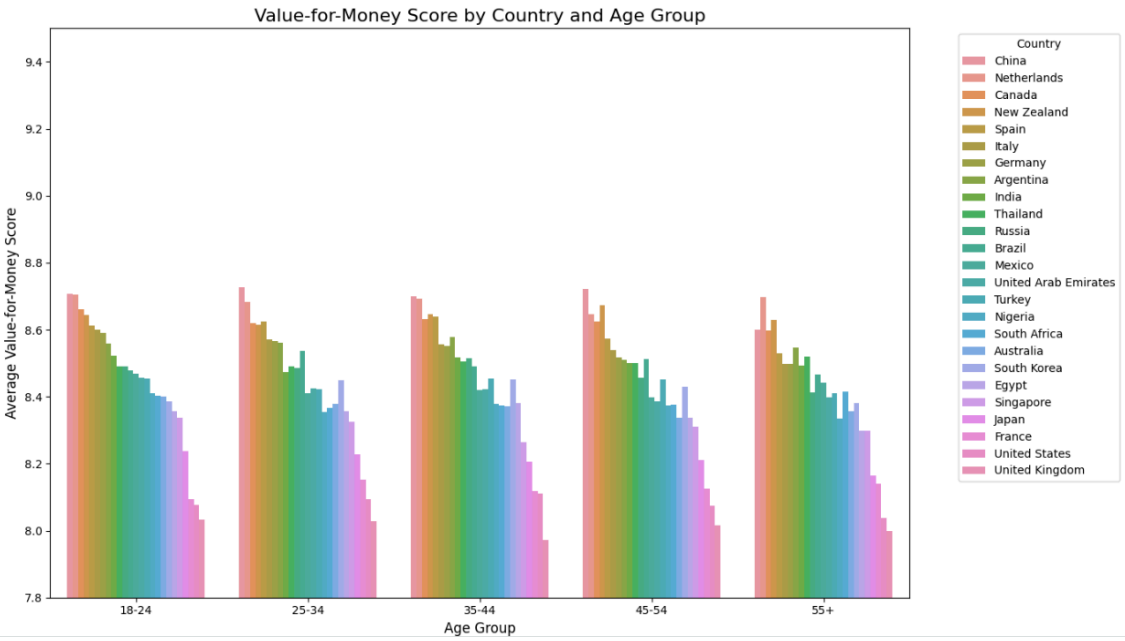
25-34	China
35-44	China
45-54	China
55+	Netherlands

Q1 steps:

I first created the avg_scores by grouping by traveller type and city and selecting the mean overall score. Simply sorting descendingly according to score then dropping duplicate traveller types provides the best city for each traveller type.

Q2 steps:

Avg_scores was created using the same approach as Q1 but grouping by age group and country and selecting the mean value for money score. This was then sorted ascendingly and descendingly according to age group and value for money score respectively. This means that within each age group (sorted ascendingly) the entries were sorted descendingly according to value for money. The first 3 entries within each age group were then selected to be printed. Since entries are sorted and we selected 3 entries per age group, every third entry is the top value for money country for the next age group.


Value-for-Money Score by Country and Age Group
Average Value-for-Money Score
9.4
9.2
9.0
8.8
8.6
8.4
8.2
8.0
7.8
18-24
25-34
35-44
45-54
55+
Age Group
Country
China
Netherlands
Canada
New Zealand
Spain
Italy
Germany
Argentina
India
Thailand
Russia
Brazil
Mexico
United Arab Emirates
Turkey
Nigeria
South Africa
Australia
South Korea
Egypt
Singapore
Japan
France
United States
United Kingdom

Predictive Task :

The goal of the predictive Task in this milestone is to predict in which country group a hotel belongs to from the information in the reviews .

Preparing the Data for the predictive Task :

In this part , we discuss all the steps done after the cleaning step leading to the creation of the predictive models

1- Creating the country_group column :
In this milestone , we are asked to use either a shallow FFNN or a Statistical ML model but both are examples of **supervised learning** which means that they train on labelled data and right now the data isn’t labelled as the target feature we want to predict (country_group) isn’t there in the data yet .

To create the target variable for our multi-class classification, we first defined the 11 regional groups based on the categories specified in the project requirements. We implemented this by building a Python dictionary to act as a lookup map, which assigned each specific country ('France' for instance) to its corresponding country group ('Western_Europe'). We then applied this dictionary to the hotel_country column (the hotel's country from the joined dataset) using the pandas **.map()** function . This operation created the new country_group column successfully translating the country for all 50,000 reviews into one of the 11 target labels and making the target variable ready for supervised learning.

```
Distribution of Country Groups:
country_group
Western_Europe          11876
Africa                   6132
East_Asia                6082
Southeast_Asia           4070
Oceania                  4014
Middle_East              3983
North_America            3962
South_America            3918
North_America_Mexico     2004
South_Asia               1989
Eastern_Europe           1970
Name: count, dtype: int64
```

A crucial finding from the data distribution of the country groups is the class **imbalance** within the country_group target variable. The Western_Europe group is a clear **majority class**, accounting for **23.7%** of the entire dataset. In stark contrast, the next largest group, Africa, makes up only **12.2%**, and the smallest of minority classes such as Eastern_Europe, fall to as low as **3.9%** of the data. This could mean that a model can develop a bias towards the majority country_groups (for instance predicting Western_Europe more than any other group) as it sees these more than any of the minority country_groups .

Choosing to balance the data or not isn't straightforward because what if the majority class really occurs more than often in real life and it isn't only the case of picking a dataset .for instance , a model that predicts what disease a patient has but what if there is a certain disease that actually many patients have so now the imbalance in the data might be a real world problem .

In our case , there are 2 cases that we have to consider :

1- what if the majority country_group isn't actually visited in real life more often than other country_groups but there are simply more examples of it in this dataset

2- what if the majority country_group is really visited in real life more often than other country_group which means that the imbalance in the data might come into play here

We couldn't make up our minds about which case we will go with as if we decide to balance the data there will be compromises so we decided that we will test the model with the data imbalanced and another time with the data balanced and the model which scores a balance between a high weighted F1 score and high accuracy is the option we will go with . We chose the F1 score to be one of the deciders as it balances between precision and recall even though Accuracy is a famous metric , it will not be sufficient here as the model can score a good accuracy if it only predicted all the answers as the majority country_group .

To test whether we will balance the data or not , we experimented (the results will be shown later) with two common resampling techniques.
The first, **Random Oversampling**, addresses the issue by randomly duplicating samples from the minority classes (like 'Eastern-Europe') and adding them to the training set until they have as many samples as the majority class ('Western-Europe'). While this forces the model to pay more attention to these rare classes, its main consequence is a high risk of **overfitting**, as the model sees the exact same minority samples multiple times and may simply memorize them instead of learning a general pattern.
The second method is **Random Undersampling** and it works in reverse by randomly deleting samples from the majority class until it

matches the size of the smallest minority class. The primary consequence of this method is the **significant loss of potentially valuable information**, which can lead to a weaker, less informed model that underfits the data.

2- dealing with non numerical features :
It is essential to preprocess all non-numerical (categorical) features before feeding them to a machine learning model as Models like Logistic Regression are mathematical and cannot perform calculations on raw text strings like 'Female', 'Solo', or '18-24'. Therefore, we must encode these text-based categories into a numerical format. For the age_group feature, we specifically chose Label Encoding. This is because age_group is an **ordinal feature**, meaning its categories have a natural and ranked order. By converting these groups into integers (0, 1, 2,4) we preserve this rank and allow the model to learn patterns related to the *progression* of age, which was our goal. This strategy contrasts with our other categorical features like user_gender and traveller_type, which are **nominal** (having no natural order). These will be handled separately using One-Hot Encoding during the final model preparation .

```
The unique categories for the 'age_group' column are:
['25-34' '55+' '18-24' '35-44' '45-54']
```

```
--- Original vs. Encoded Column ---
  age_group  age_group_encoded
0     25-34                  1
1       55+                  4
2     25-34                  1
3     18-24                  0
4     35-44                  2
5     45-54                  3
6     18-24                  0
7       55+                  4
8     35-44                  2
9     25-34                  1
```

3- choosing the features I will use for the prediction model :
The first feature we will discard is the star_rating as when we investigated the columns in the hotels table , we found out that there is only one value in that column which is (5) which means there is no difference between all hotels in this column so this feature has no significance in prediction .

All IDs will be discarded as they are arbitrary labels used for database indexing, not predictive features. A model would just memorize them, leading to severe overfitting.

Review_text will also be discarded as This feature would require Natural Language Processing (NLP) techniques and It cannot be used directly by a statistical model like Logistic Regression.

When it comes to join_date and review_date , we have observed a very strange observation . we have found out there are some rows where the review_date is before the join_date and this could mean 2 scenarios :
1- that those rows are spam reviews and they somehow became attached to users because of an error in the booking system
2- that the app allows any guest to review a hotel , and when the guest signs up for the app , the review he made before is linked to his account later

	review_date	join_date
0	2022-10-07	2021-03-21
1	2021-01-18	2023-10-05
2	2020-12-04	2021-12-18
3	2023-06-17	2023-02-02
4	2020-09-06	2024-12-10
...	...	...
49995	2020-12-26	2023-09-05
49996	2020-06-14	2020-09-14
49997	2024-04-18	2022-07-12
49998	2025-08-06	2022-06-13
49999	2023-12-14	2024-02-09

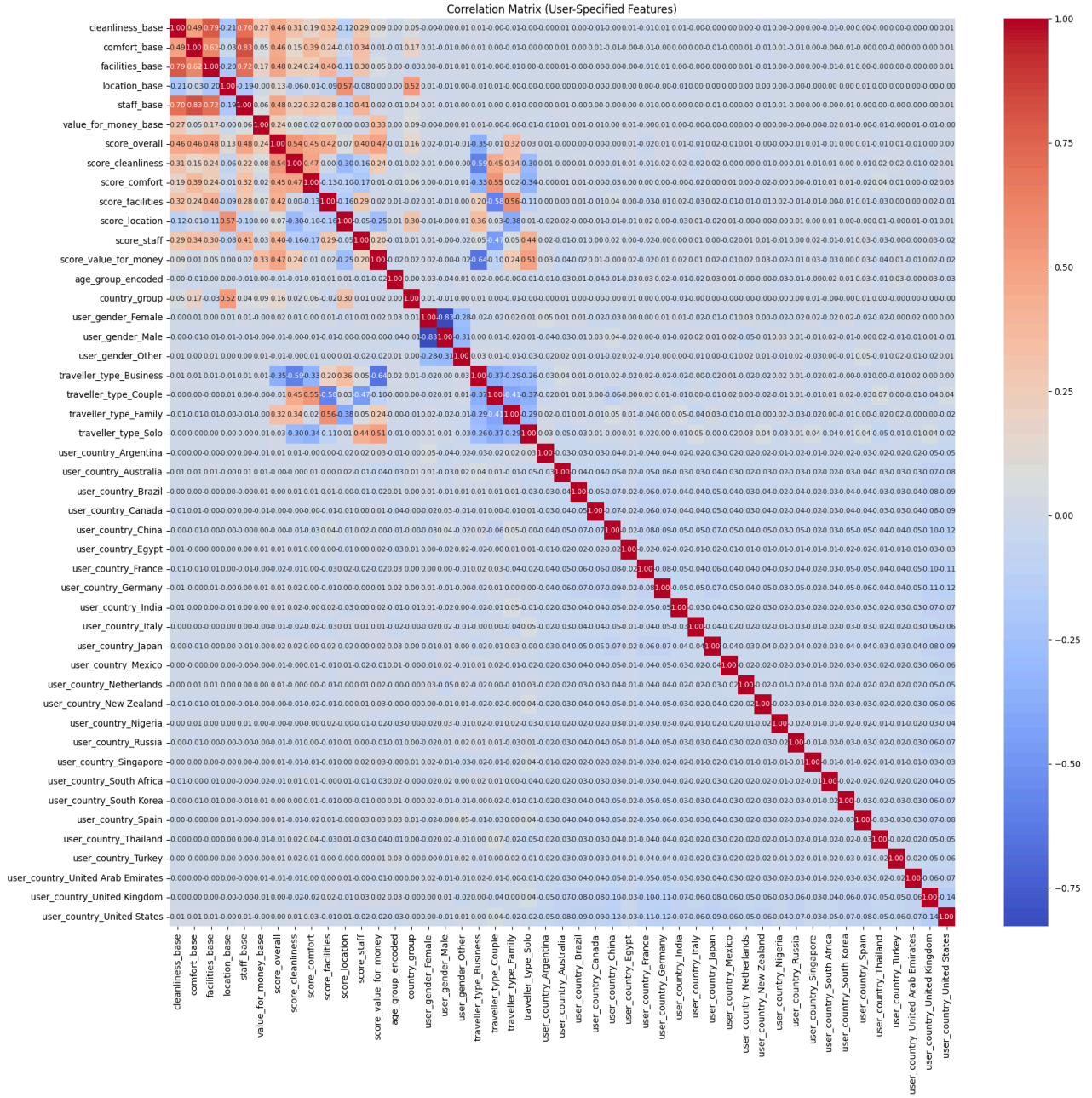
Another problem with using the date features is that they cannot be used directly and I have to make 3 columns for each date to represent day , month and year and this will add so much dimensionality only to represent dates
For the mentioned 2 reasons , we decided we will discard the dates completely .

We also decided not to take any location based feature of the hotel (hotel_country , city , lon , lat) because they are directly related to the hotel's geographic location (like city, hotel_country, lat, and lon) and this may cause data leakage. The model's objective is to predict the country_group, which is itself a location label. Using features like city or hotel_country to predict country_group isn't a real prediction; it's essentially giving the model the answer key as the model can just memorize that france is located in western_Europe .

We decided not to use any feature from the hotel table as there are only 25 hotels but there are 50k review divided on those hotels and this means that any feature unique to a hotel can be treated as a unique identifier to the hotel so the model will just memorize the identifiers of the 25 hotels instead of using information from reviews to predict .

We later proved that those features caused data leakage and overfitting happened (will show it when we talk about the results of the models)

Okay now that we ruled out some of the features , the last thing to do to determine what features to take is a correlation matrix (heatmap) to see which features are correlated to the country group. We also tried to use this heatmap to see if we can spot the possible data leakage as if the features taken from the hotel table have a very high correlation with the country group column then this would mean they will cause a data leakage.



Here we didn't want to know the correlation between each feature and each country group , we just wanted to know the correlation of each feature with the country group column overall so that's why we didn't one hot encode the country group but we label encoded it.

```
--- CORRELATION WITH TARGET (country_group) ---
country_group                          1.000000
location_base                          0.519075
score_location                         0.302788
comfort_base                           0.167844
score_overall                          0.160780
value_for_money_base                   0.091905
score_comfort                          0.061909
cleanliness_base                       0.049437
staff_base                             0.044185
score_value_for_money                  0.021236
score_cleanliness                      0.016984
score_staff                            0.010192
traveller_type_Business                0.008706
user_country_Australia                 0.008706
user_country_Spain                     0.008501
user_gender_Female                     0.005917
user_country_Singapore                 0.005835
user_country_Egypt                     0.005297
age_group_encoded                      0.004409
user_country_France                    0.004399
user_country_Thailand                  0.004366
user_country_Japan                     0.004302
user_gender_Other                      0.004295
user_country_Brazil                    0.003512
traveller_type_Couple                  0.002906
user_country_United Arab Emirates      0.002232
user_country_Germany                   0.000648
user_country_Russia                    0.000223
user_country_South Africa             -0.000217
user_country_Netherlands              -0.000985
user_country_New Zealand              -0.001149
user_country_Turkey                   -0.001532
user_country_Mexico                   -0.001829
user_country_South Korea              -0.001855
user_country_Canada                   -0.002087
user_country_Italy                    -0.002966
user_country_Argentina                -0.002993
user_country_Nigeria                  -0.003307
user_country_United States            -0.003511
user_country_China                    -0.003913
user_country_United Kingdom           -0.004683
traveller_type_Solo                   -0.004933
traveller_type_Family                 -0.006814
user_gender_Male                      -0.008387
user_country_India                    -0.008633
score_facilities                      -0.018529
facilities_base                       -0.034459
Name: country_group, dtype: float64
```

The numbers on the right represent the linear correlation between the features and the country_group column but there is a very important thing to understand and that these numbers represent only the linear relationships so a low correlation number doesn't mean that there isn't relationship but the relationship isn't linear and can't be computed using the heatmap .

What can be seen from the results that most of the correlations aren't high enough to consider dangerous except (location_base) Which is also isn't high enough but it is definitely higher than the other features which might hint at the data leakage problem that we might face .

Seeing that most of the features don’t have a high correlation as most of them are lower than 0.01 and even the features with negative correlation are higher than 0.0.1 so that might mean 2 things :
1- the correlation matrix cannot compute the correlation because it isn’t linear and it is more complex than that
2- all the features contribute a little part to the final prediction so that means I cannot discard the features that score a small number as most of them scored a small number
We decided that we will take any feature that doesn’t score a direct zero as all of them scored a small number and we will try another way to prove the data leakage so the feature list for now will be :
'cleanliness_base', 'comfort_base', 'facilities_base',
'location_base', 'staff_base', 'value_for_money_base',
'score_overall', 'score_cleanliness', 'score_comfort',
'score_facilities', 'score_location', 'score_staff',
'score_value_for_money','user_country',
'user_gender', 'traveller_type', 'age_group_encoded'

Predictive models :

We now move to the part about creating the model we want .
Our final choice was that we will choose a FFNN as our model (which we will talk about later) but to get to that point we had to make a comparison between FFNN and logistic regression model .

The tests we made :
1- test both models with all the (_base) features (feature set 1) to test if there is a data leakage
2- test both models with a feature set (feature set 2) without the (_base) features to see confirm if there was a data leakage or not
3- test both models with a feature set (feature set 2) and leave it imbalanced .

4- test both models with a feature set (feature set 2) and balance it one time using random undersampling and another time using random oversampling .
5- use feature engineering to make the models perform better

Logistic regression model :
In each test we will do next , we scaled the input data before giving it to the model because as was said in lab 3 that it is a good practice to scale all the features so that features with different ranges or units be treated differently . For instance we have the age_group feature which is label encoded to be from 0 to 4 and every other rating feature is from 0 to 10 so without scaling the model might think that the ratings are more important than the age_group only because the range .
The first time I ran the model , I got a convergence error and it was suggested in the error to scale the data to solve this error and when I did , the problem was solved so that was another reason why I scaled the data

We also splitted the dataset into a train set and a test set .
The train set is 80% of the original dataset and will be used to training the model .
The test set is 20%of the original dataset and will be used to test and evaluate the model

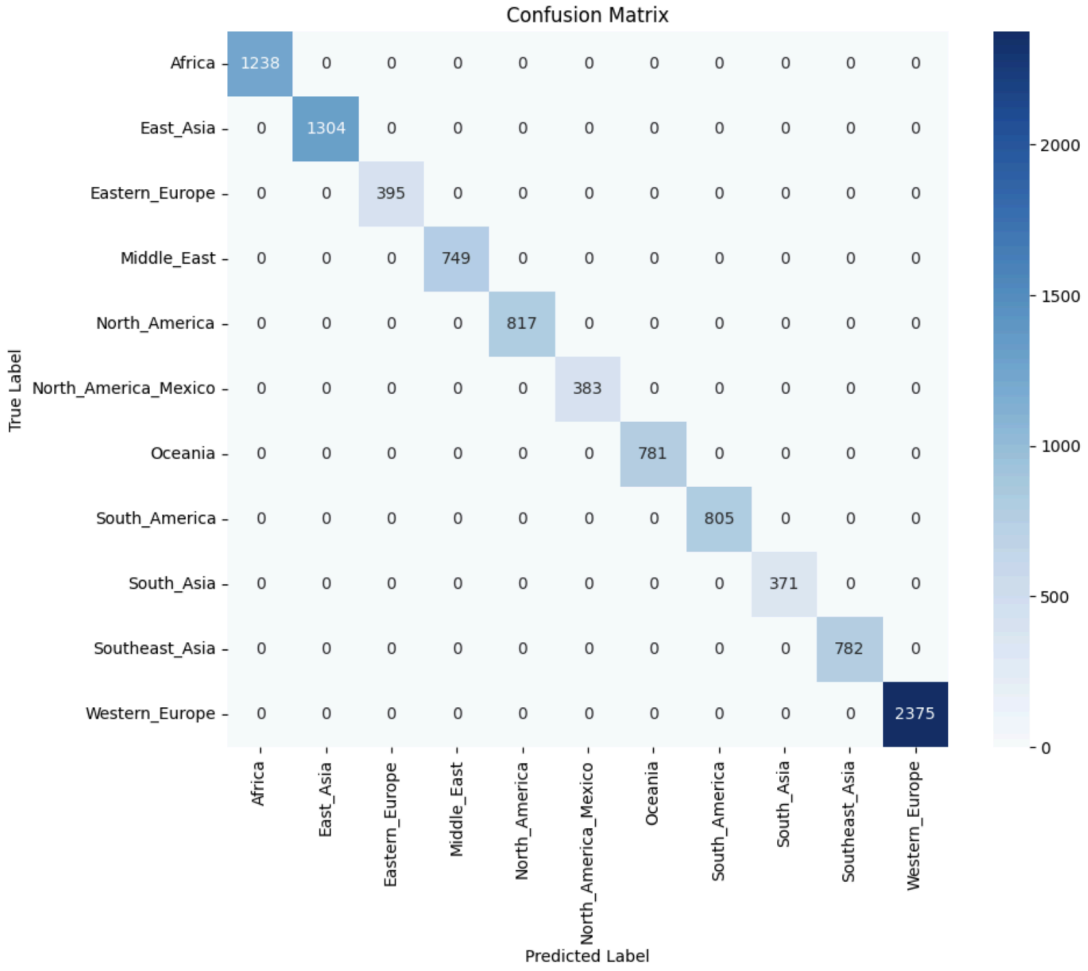
Test 1 :
In this test we tried to train a logistic regression model with the following feature list

```
feature_list = [
    'cleanliness_base', 'comfort_base', 'facilities_base',
    'location_base', 'staff_base', 'value_for_money_base',
    'score_overall', 'score_cleanliness', 'score_comfort',
    'score_facilities', 'score_location', 'score_staff',
```

'score_value_for_money','user_country',
'user_gender', 'traveller_type', 'age_group_encoded'
]

We left all _base features to check if there is a data leakage or not

```
--- Accuracy Check ---
   Training Accuracy: 1.0000
   Testing Accuracy:  1.0000
--------------------------

--- Full Test Set Report ---
                      precision    recall  f1-score   support

              Africa       1.00      1.00      1.00      1238
           East_Asia       1.00      1.00      1.00      1304
      Eastern_Europe       1.00      1.00      1.00       395
         Middle_East       1.00      1.00      1.00       749
       North_America       1.00      1.00      1.00       817
North_America_Mexico       1.00      1.00      1.00       383
             Oceania       1.00      1.00      1.00       781
       South_America       1.00      1.00      1.00       805
          South_Asia       1.00      1.00      1.00       371
      Southeast_Asia       1.00      1.00      1.00       782
      Western_Europe       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
```



From the results of the classification report on the predicted Y values from the test set we can see that the model scored a perfect 100% in every metric which means that the model didn't even make 1 mistake . we can also see from the confusion matrix that we have a perfect diagonal so each country_group was predicted correctly . These results mean that there is data leakage which caused the model to memorize every country_group (overfitting) .

Test 2 (this represents test 2 and 3 from the list) :

In this test , we tried a logistic regression model with the following feature list

```
feature_list_2 = [
    'score_overall', 'score_cleanliness', 'score_comfort',
    'score_facilities', 'score_location', 'score_staff',
    'score_value_for_money','user_country',
    'user_gender', 'traveller_type', 'age_group_encoded'
]
```

This feature list removes all the (_base) features to confirm the data leakage

```
--- Model 2: Accuracy comparison ---
   Training Accuracy: 0.3420
   Testing Accuracy:  0.3454
--------------------------------

--- Model 2: Full Test Set Report ---
                      precision    recall  f1-score   support

              Africa       0.30      0.48      0.37      1238
           East_Asia       0.37      0.45      0.41      1304
      Eastern_Europe       0.27      0.04      0.07       395
         Middle_East       0.21      0.05      0.09       749
       North_America       0.20      0.08      0.12       817
North_America_Mexico       0.31      0.09      0.14       383
             Oceania       1.00      0.00      0.00       781
       South_America       0.33      0.22      0.26       805
          South_Asia       0.33      0.00      0.01       371
      Southeast_Asia       0.38      0.18      0.25       782
      Western_Europe       0.37      0.76      0.50      2375

            accuracy                           0.35     10000
           macro avg       0.37      0.21      0.20     10000
        weighted avg       0.37      0.35      0.28     10000
```

This time we can see that the performance of the model decreased significantly which confirms that the (_base) features caused data leakage and overfitting .

When we compare the accuracy in training and testing we can see that they are almost the same which means that there is no overfitting because for overfitting to happen , the accuracy during training has to be very high and marginally higher than the testing accuracy .
There is something that we have to keep in mind which is that the data is imbalanced and this is very apparent in the confusion matrix. If we look at the last column on the right of the matrix , we can see that the model keeps predicting other country groups as western europe as this is the most major country group in the dataset .

This means that the model can get a good accuracy by only predicting Western Europe so going forward we will not compare the models based only on the accuracy .

The **Precision** metric here means when the model predicts a certain country group , what was the percentage of that country group to be correct and we can see that the model scored 37% . This means that 37% of the times the model predicted a certain country group , it was the actual group .

The **Recall** metric tells us out of all the actual reviews belonging to a hotel that is in a specific country group, what percentage did the model successfully identify ? Our model's weighted average recall is **35%**. This means that, on average, the model only identified 35% of the true reviews for each country group present in the test set, failing to find the remaining 65%.

The **F1-Score** combines Precision and Recall into a single metric providing a balanced measure of the model's performance and we can see that the model scored 28% in the weighted average

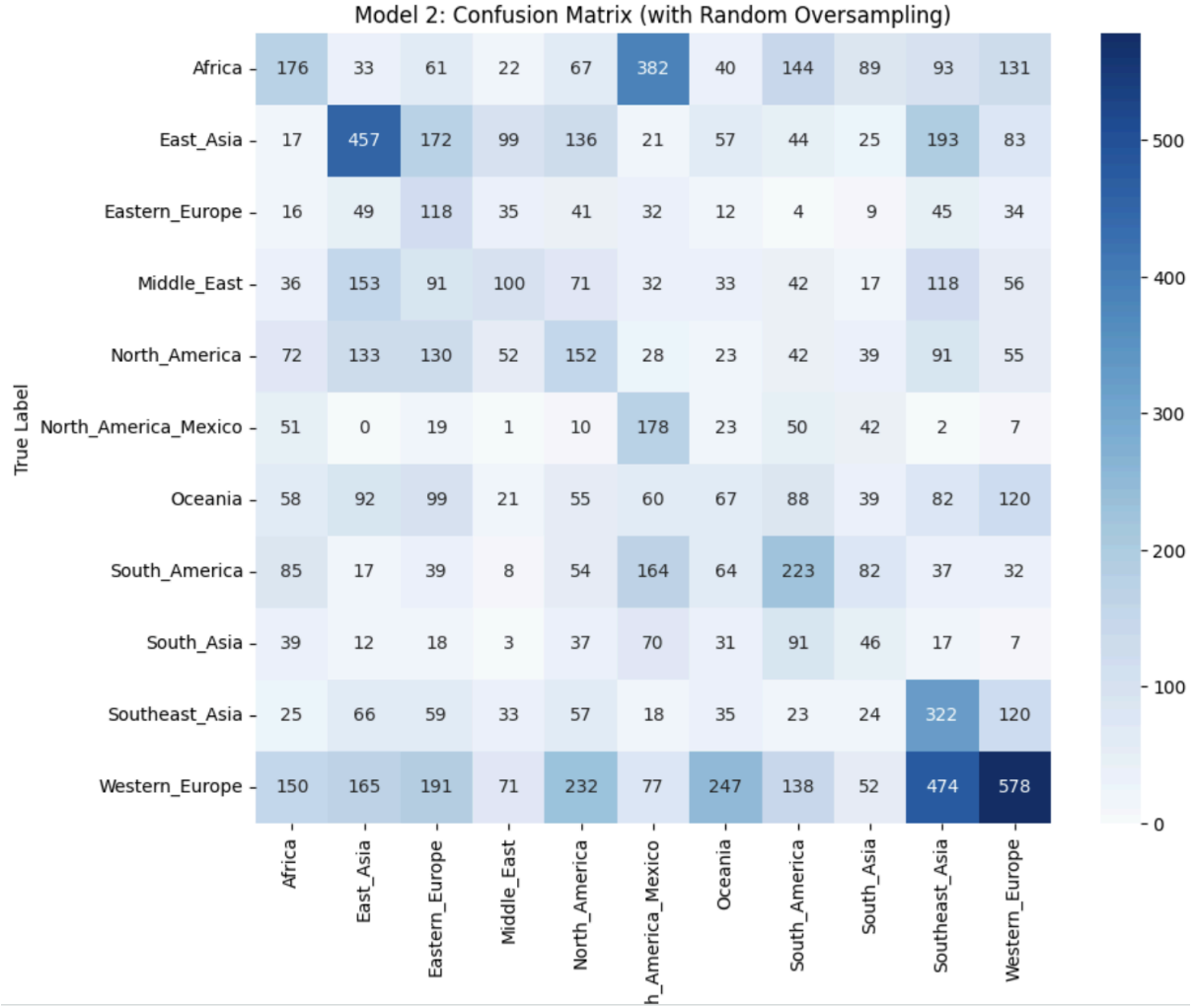
Test 3 (this is test 4 according to the list of tests) :

In this test , we are going to test the same feature list (feature list 2) but this time with Random oversampling done to it and this will randomly remove some rows that belong to the majority country groups like western Europe which will cause the data to be balanced .

We wanted to see if the model will perform better with the data being balanced or not .

--- Model 2: Full Test Set Report (with Random Oversampling) ---

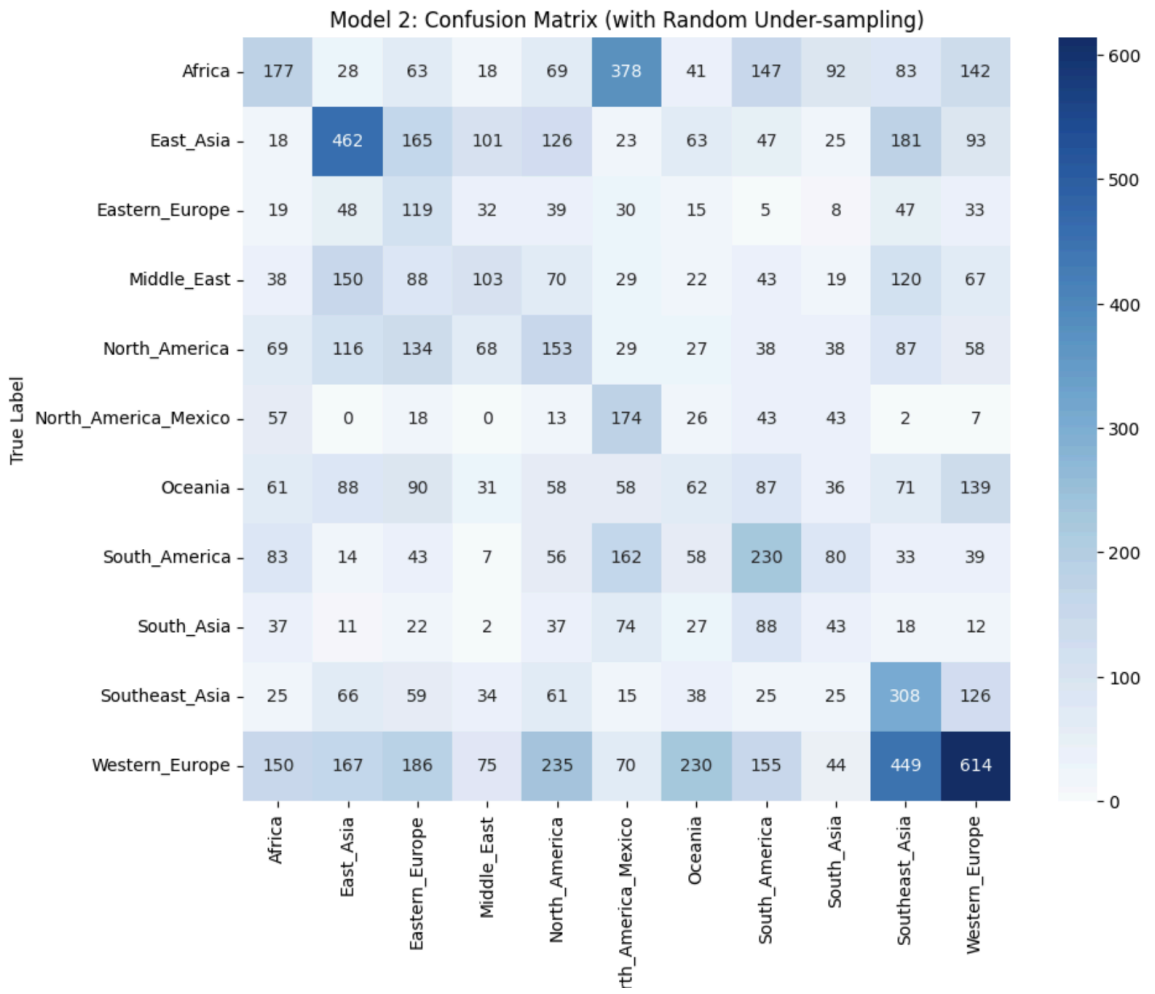
	precision	recall	f1-score	support
Africa	0.24	0.14	0.18	1238
East_Asia	0.39	0.35	0.37	1304
Eastern_Europe	0.12	0.30	0.17	395
Middle_East	0.22	0.13	0.17	749
North_America	0.17	0.19	0.18	817
North_America_Mexico	0.17	0.46	0.25	383
Oceania	0.11	0.09	0.09	781
South_America	0.25	0.28	0.26	805
South_Asia	0.10	0.12	0.11	371
Southeast_Asia	0.22	0.41	0.29	782
Western_Europe	0.47	0.24	0.32	2375
accuracy			0.24	10000
macro avg	0.22	0.25	0.22	10000
weighted avg	0.28	0.24	0.24	10000



We can see that the model performed worse in every metric that matter to us (weighted f1 and accuracy) than when the data is imbalanced so we will discard random oversampling .
The only thing that is better about this model is that it now predicts other country_groups more than before as we can see from the confusion matrix that the diagonal now is filled with more correct answers .

We will now test the same feature list (feature list 2) with random oversampling which chooses rows at random that belong to minority country groups and duplicates them until the data is balanced .

```
--- Model 2: Full Test Set Report (with Random Under-sampling) ---
                     precision    recall  f1-score   support

             Africa       0.24      0.14      0.18      1238
          East_Asia       0.40      0.35      0.38      1304
     Eastern_Europe       0.12      0.30      0.17       395
        Middle_East       0.22      0.14      0.17       749
      North_America       0.17      0.19      0.18       817
North_America_Mexico       0.17      0.45      0.24       383
            Oceania       0.10      0.08      0.09       781
      South_America       0.25      0.29      0.27       805
         South_Asia       0.09      0.12      0.10       371
     Southeast_Asia       0.22      0.39      0.28       782
     Western_Europe       0.46      0.26      0.33      2375

           accuracy                           0.24     10000
          macro avg       0.22      0.25      0.22     10000
       weighted avg       0.28      0.24      0.25     10000
```



We can see that the model performed worse in every metric that matter to us (weighted f1 and accuracy) than when the data is imbalanced so we will discard random undersampling .
The only thing that is better about this model is that it now predicts other country_groups more than before as we can see from the confusion matrix that the diagonal now is filled with more correct answers .

Test 4 :

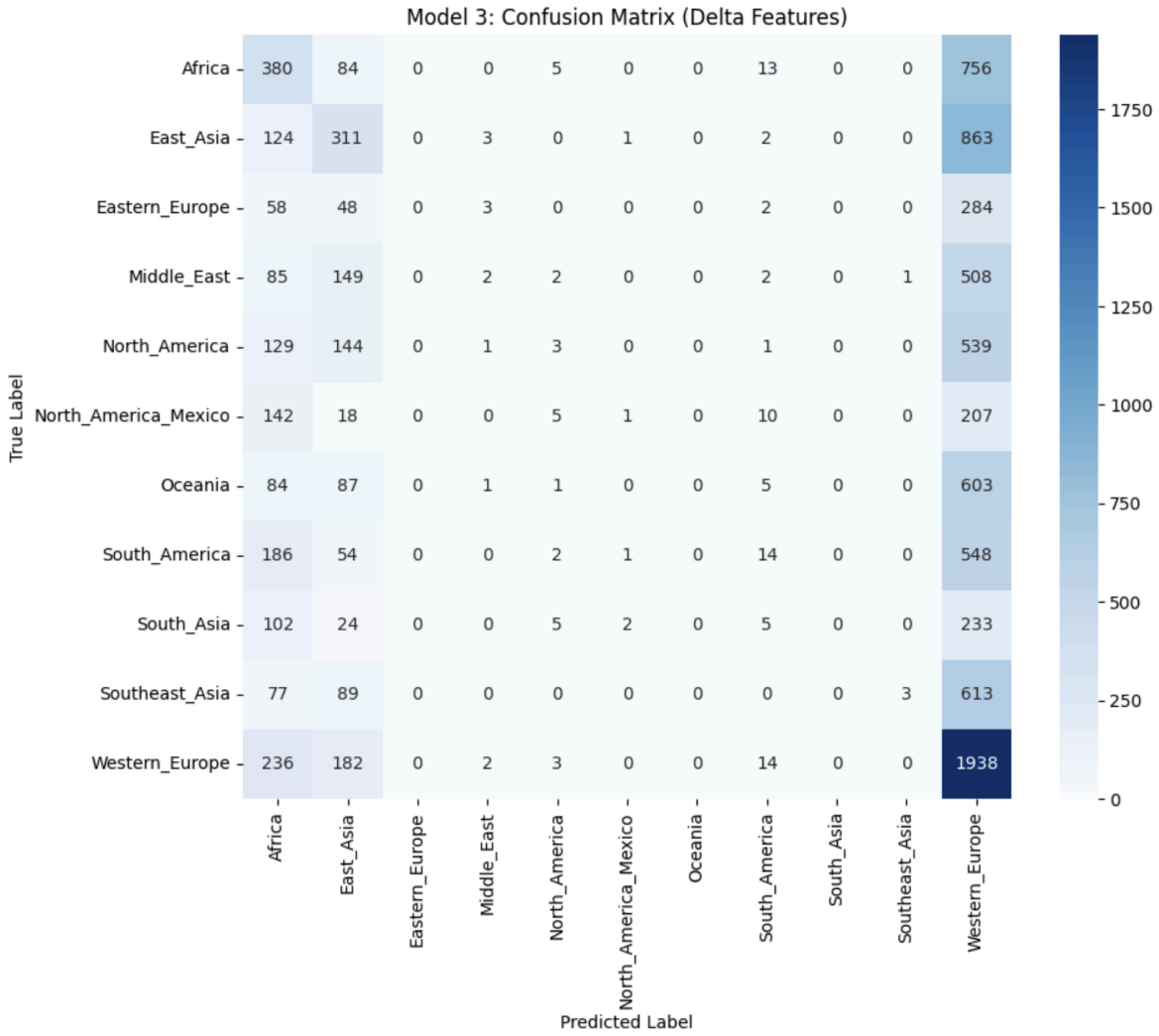
Seeing that balancing the data didn't improve the performance of the model then there is an idea we wanted to test .

What if we engineered a new feature where we get the difference between the (_base) ratings and the scores given by the reviews .

This will in theory let us embed the information from the (_base) features which we discarded and the information from the scores in one column that we can use .

```
# Create The New Delta Features
df['cleanliness_delta'] = df['score_cleanliness'] - df['cleanliness_base']
df['comfort_delta'] = df['score_comfort'] - df['comfort_base']
df['facilities_delta'] = df['score_facilities'] - df['facilities_base']
df['location_delta'] = df['score_location'] - df['location_base']
df['staff_delta'] = df['score_staff'] - df['staff_base']
df['value_for_money_delta'] = df['score_value_for_money'] - df['value_for_money_base']

print("New 'delta' features created successfully.")

# Define Feature List for Model 3
# Use only the new delta features and user demographics
feature_list_3 = [
    'cleanliness_delta', 'comfort_delta', 'facilities_delta',
    'location_delta', 'staff_delta', 'value_for_money_delta',
    'user_gender','user_country', 'traveller_type', 'age_group_encoded',
]
```

```
--- Model 3: Full Test Set Report (Delta Features) ---
                      precision    recall  f1-score   support

              Africa       0.24      0.31      0.27      1238
           East_Asia       0.26      0.24      0.25      1304
      Eastern_Europe       0.00      0.00      0.00       395
         Middle_East       0.17      0.00      0.01       749
       North_America       0.12      0.00      0.01       817
North_America_Mexico       0.20      0.00      0.01       383
             Oceania       0.00      0.00      0.00       781
       South_America       0.21      0.02      0.03       805
          South_Asia       0.00      0.00      0.00       371
      Southeast_Asia       0.75      0.00      0.01       782
      Western_Europe       0.27      0.82      0.41      2375

            accuracy                           0.27     10000
           macro avg       0.20      0.13      0.09     10000
        weighted avg       0.23      0.27      0.17     10000
```



We can see that using these new features has decreased the performance of the model where the model performed worse in f1 score and accuracy than with feature list 2

Seeing that this experiment didn’t work so we will stick with the model with feature list 2 we worked with in test 2

FFNN :

As mentioned previously, we carried out 5 tests. For all 5 tests (and the other 2 which we will mention later), we carried out the same pre-processing :-

1. Define Feature List
2. Define X Values (the defined feature list)
3. Refine Categorical Columns to be Encoded
4. Encode the X Values
5. Define the Y Values (the target)
6. Encode the Y Values
7. Split the Encoded X and Y Values to Training(80%) and Testing(20%)
8. Scale the X Values
9. Convert the Scaled X and Y Values to Arrays of float32
10. Convert Training and Testing Targets to Categorical Targets.
11. Finally, Determine the Number of Output Classes

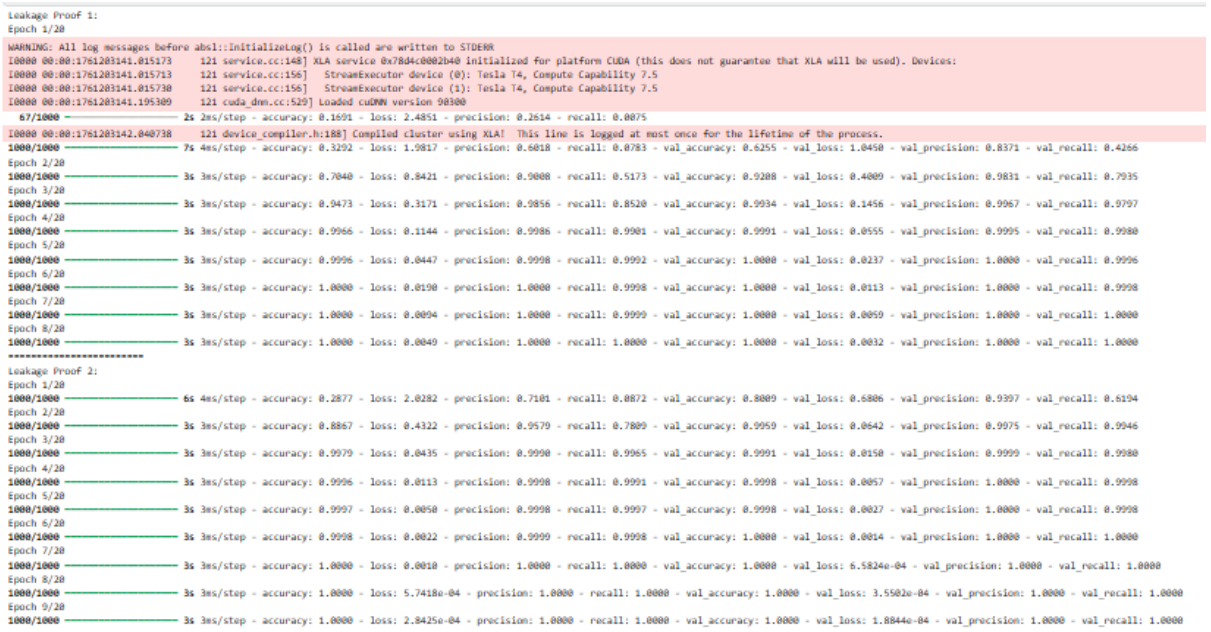
<u>Test 1:</u>

After the pre-processing, we start developing the FFNN. For the first case (proving the data leakage by using “feature_list”), we attempted to prove the data leakage by using two architectures (just to make sure). The first model consisted of the input data (training), a 16-neuron hidden layer,and the output layer (number of classes). The second model (in the same context) had an additional 16-neuron hidden layer making.

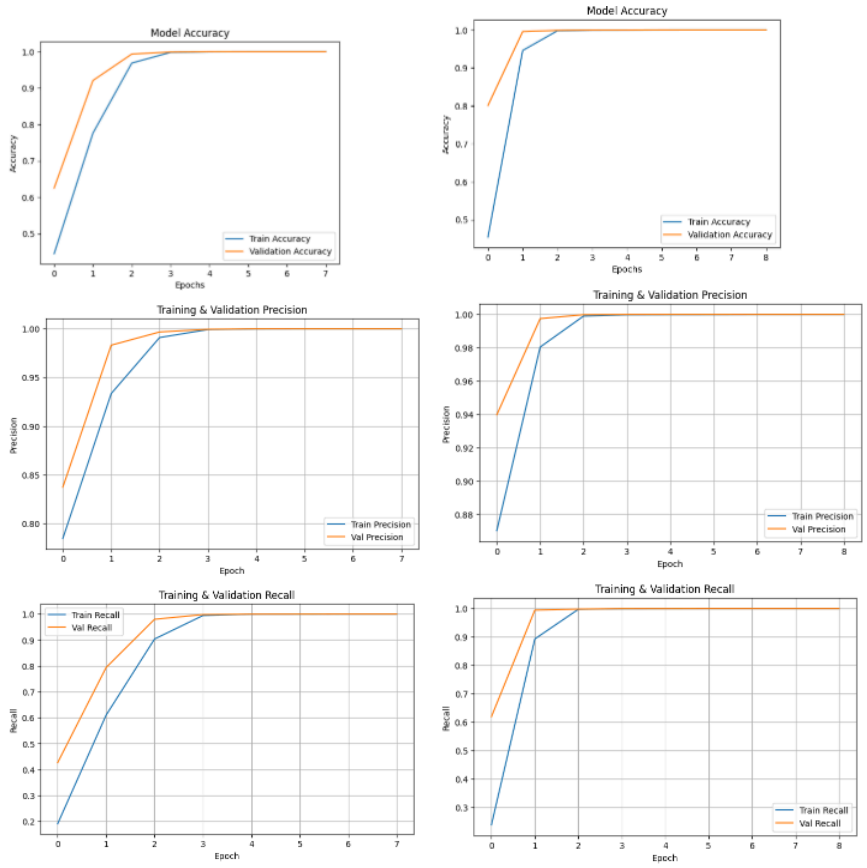
Throughout the FFNN model development, we assigned the hidden layers’ activation to have ‘relu’, while the output layers’ activation to have ‘softmax’. ‘Relu’ returns a 0 if the input is negative, positive value otherwise. This helps the model learn complex patterns. ‘Softmax’ returns a probability for each class, helping us classify the output to a certain class.

After assigning the models, we start compiling using ‘adam’ optimizer, ‘categorical_crossentropy’ loss, and we measure: [accuracy, precision, and recall].

We start training the model on the training data with a validation split of 20%, for a minimum of 20 epochs (because we know it will overfit so no need to overwork the devices). We also assign an early stop of 3 epochs so we don’t have to wait for long until the model stops training .



We then plot the accuracy, precision, and recall for each model. On the **left is model 1**, on the **right is model 2**.



As we can see, both models took less than 8 epochs and seemingly memorized the training data, thus the validation data would also show an overfit. Afterwards, we still develop a Classification report to further determine and judge the performance. On the left is model 1, the right is model 2.

Classification Report:

	precision	recall	f1-score	support
Africa	1.00	1.00	1.00	1238
East_Asia	1.00	1.00	1.00	1304
Eastern_Europe	1.00	1.00	1.00	395
Middle_East	1.00	1.00	1.00	749
North_America	1.00	1.00	1.00	817
North_America_Mexico	1.00	1.00	1.00	383
Oceania	1.00	1.00	1.00	781
South_America	1.00	1.00	1.00	805
South_Asia	1.00	1.00	1.00	371
Southeast_Asia	1.00	1.00	1.00	782
Western_Europe	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

Classification Report:

	precision	recall	f1-score	support
Africa	1.00	1.00	1.00	1238
East_Asia	1.00	1.00	1.00	1304
Eastern_Europe	1.00	1.00	1.00	395
Middle_East	1.00	1.00	1.00	749
North_America	1.00	1.00	1.00	817
North_America_Mexico	1.00	1.00	1.00	383
Oceania	1.00	1.00	1.00	781
South_America	1.00	1.00	1.00	805
South_Asia	1.00	1.00	1.00	371
Southeast_Asia	1.00	1.00	1.00	782
Western_Europe	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

Both models were able to correctly predict each class 100% of the time when tested on the scaled testing data. Thus proving the leakage.

Test 2 (feature_list_2 without resampling):

Now, we move to ‘feature_list_2’, the feature list that does not contain the “dead-giveaways” (all the _base values) as ‘feature_list’ did. Thus, we move to case/test 2.

We apply the same pre-processing as before and the first change we actually implement is within (as mentioned) the feature list being ‘feature_list_2’. We then create a FFNN similar to the first data-leakage model, having a 16-neuron hidden layer.

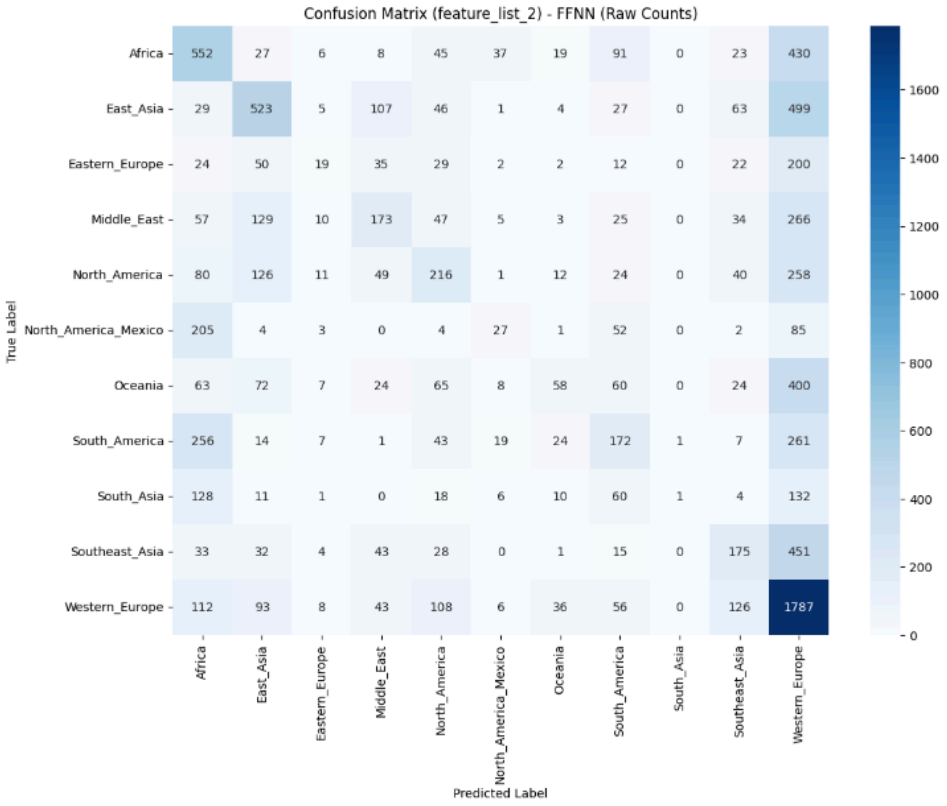
From here on forwards, we actually gave the models a chance to learn by giving them 50 epochs, while still keeping the early stop.

Results-wise, the model showed a fair 33% weighted average f1-score (as seen below).

Classification Report:

	precision	recall	f1-score	support
Africa	0.36	0.45	0.40	1238
East_Asia	0.48	0.40	0.44	1304
Eastern_Europe	0.23	0.05	0.08	395
Middle_East	0.36	0.23	0.28	749
North_America	0.33	0.26	0.29	817
North_America_Mexico	0.24	0.07	0.11	383
Oceania	0.34	0.07	0.12	781
South_America	0.29	0.21	0.25	805
South_Asia	0.50	0.00	0.01	371
Southeast_Asia	0.34	0.22	0.27	782
Western_Europe	0.37	0.75	0.50	2375
accuracy			0.37	10000
macro avg	0.35	0.25	0.25	10000
weighted avg	0.36	0.37	0.33	10000

The confusion matrix also showed the scattered predictions - which were also decent (when compared to other results from our test). The model was able to predict quite a few different country groups correctly.



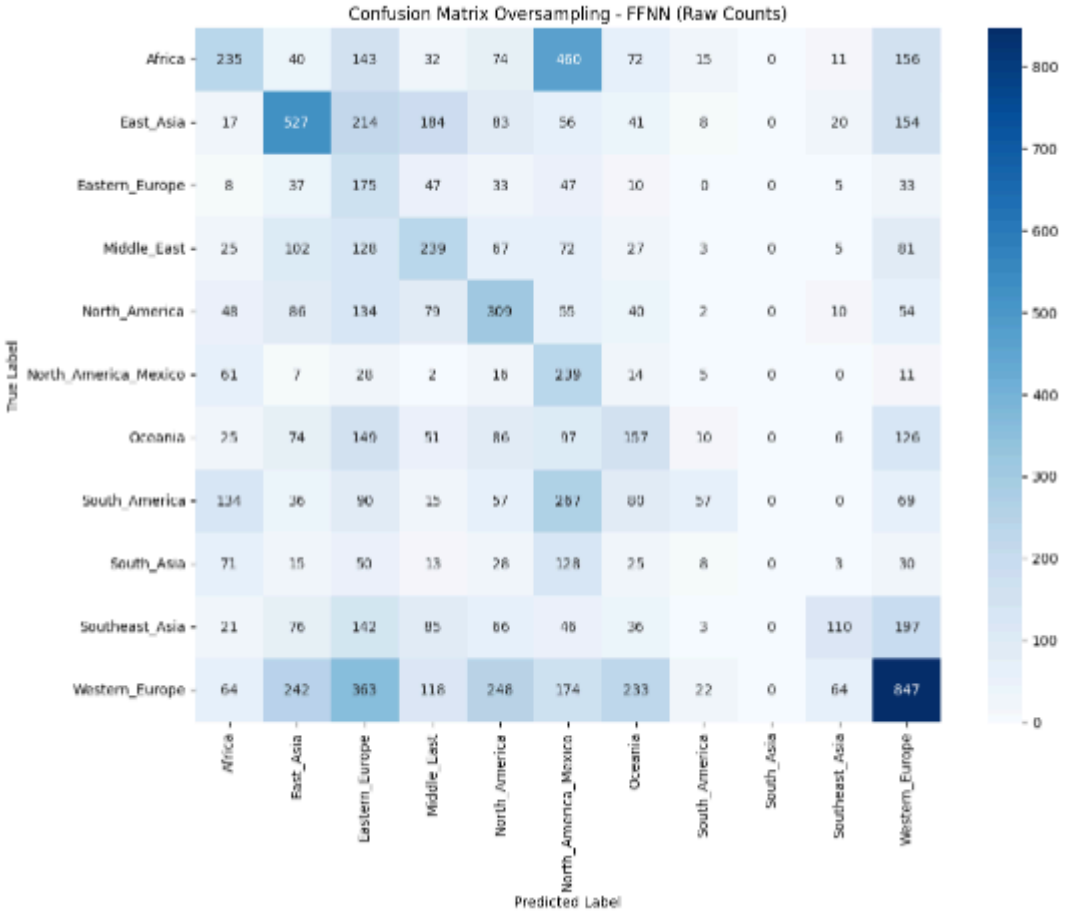
Test 3 (Over sampling):

For this test, we applied the same pre-processing but applied an oversampling of the data to address the imbalance, yet maintained the same architecture of 16-neuron, single hidden layer for fair comparison. The results were underwhelming as the weighted average score decreased to 29%.

Classification Report:

	precision	recall	f1-score	support
Africa	0.33	0.19	0.24	1238
East_Asia	0.42	0.40	0.41	1304
Eastern_Europe	0.11	0.44	0.17	395
Middle_East	0.28	0.32	0.30	749
North_America	0.29	0.38	0.33	817
North_America_Mexico	0.15	0.62	0.24	383
Oceania	0.21	0.20	0.21	781
South_America	0.43	0.07	0.12	805
South_Asia	0.00	0.00	0.00	371
Southeast_Asia	0.47	0.14	0.22	782
Western_Europe	0.48	0.36	0.41	2375
accuracy			0.29	10000
macro avg	0.29	0.28	0.24	10000
weighted avg	0.35	0.29	0.29	10000

through its confusion matrix, we can also see that in this test case the model couldn't predict a lot of correct predictions but the predictions are scattered throughout the matrix and aren't concentrated at the majority classes.



Test 4 (Undersampling):

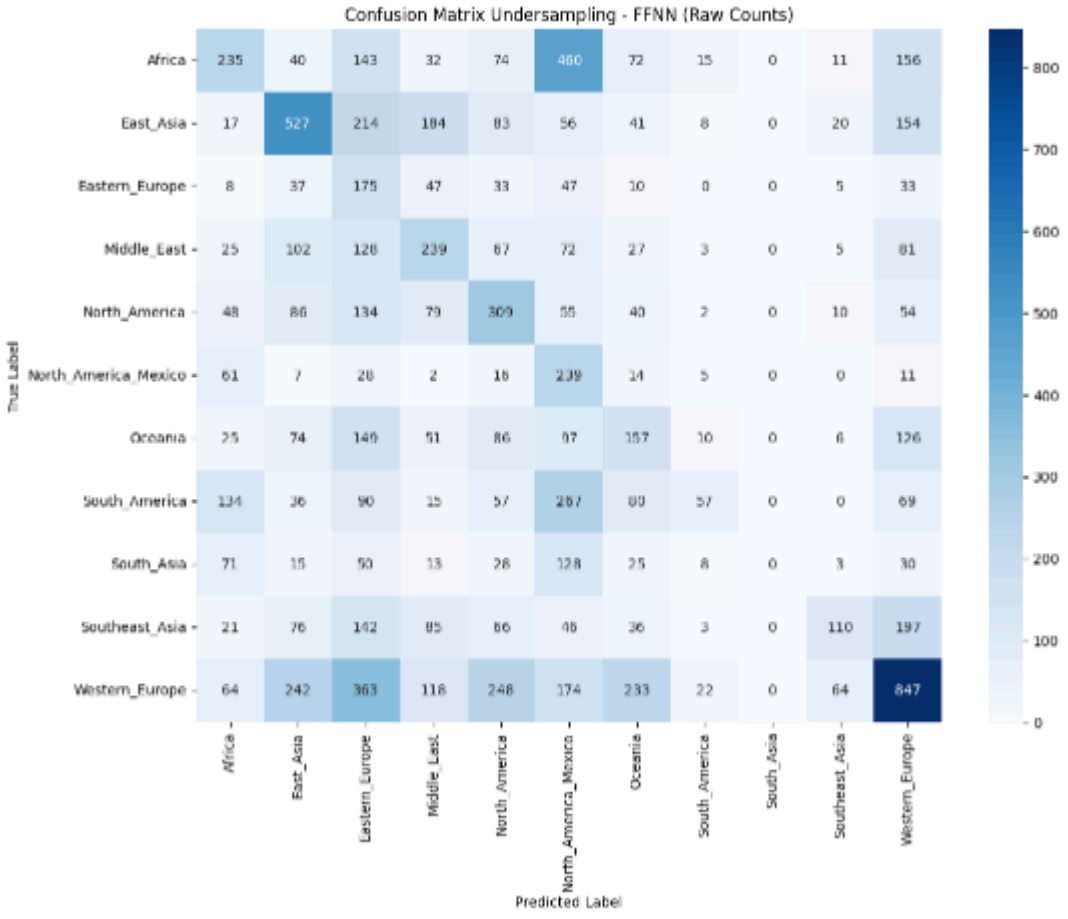
Again, we keep the same pre-processing with the addition of an undersampling of the data in order to address, again the imbalance

in the data, while also maintaining the same architecture. Again, the results are similarly underwhelming as we get the same weighted average f1-score of 29%.

Classification Report:

	precision	recall	f1-score	support
Africa	0.33	0.19	0.24	1238
East_Asia	0.42	0.40	0.41	1304
Eastern_Europe	0.11	0.44	0.17	395
Middle_East	0.28	0.32	0.30	749
North_America	0.29	0.38	0.33	817
North_America_Mexico	0.15	0.62	0.24	383
Oceania	0.21	0.20	0.21	781
South_America	0.43	0.07	0.12	805
South_Asia	0.00	0.00	0.00	371
Southeast_Asia	0.47	0.14	0.22	782
Western_Europe	0.48	0.36	0.41	2375
accuracy			0.29	10000
macro avg	0.29	0.28	0.24	10000
weighted avg	0.35	0.29	0.29	10000

The confusion matrix as well shows similar results to the oversampling test (test 3).



We now move to attempt a better change by feature engineering new columns in order to try and yield better results. Thus, we move to a newer set of inputs: ‘feature_list_3’

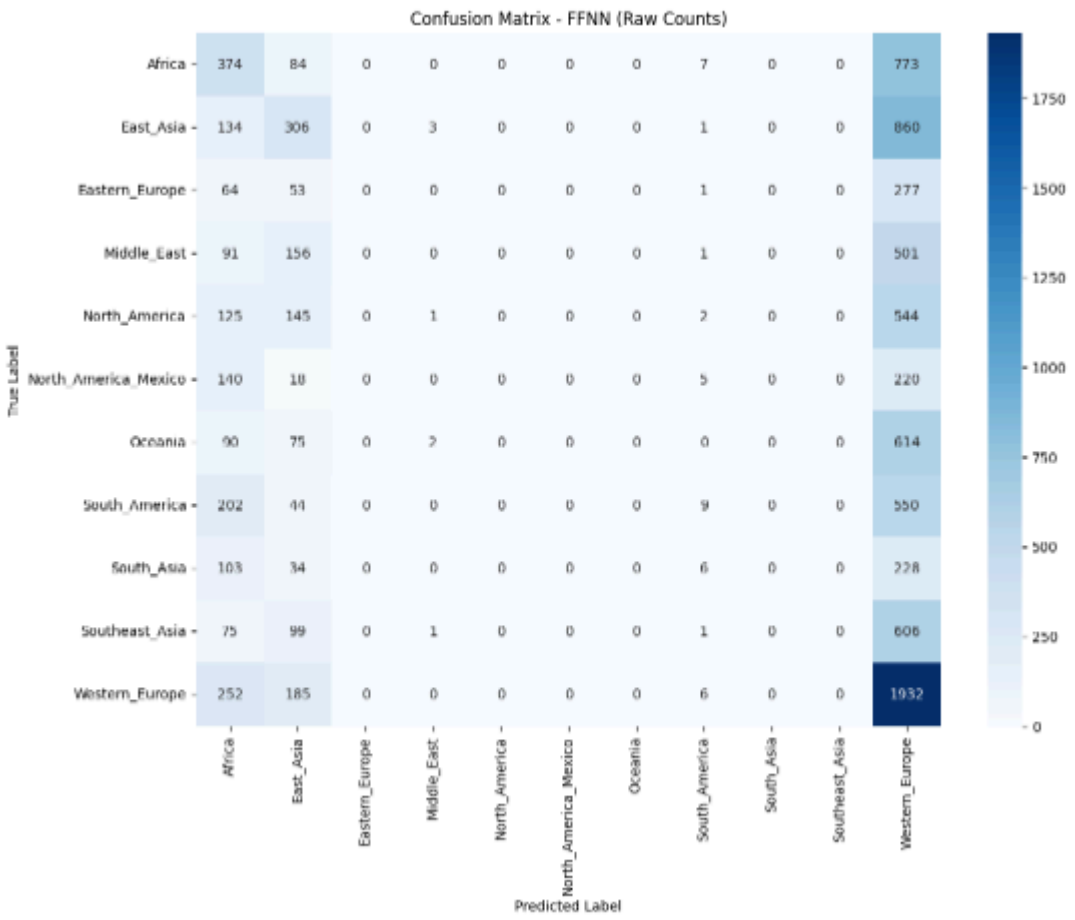
Test 5:

Though we changed the feature list (input) we are using, we still proceeded with the same pre-processing and the same architecture for the feedforward neural network. The results of this model were extremely underwhelming. We actually see a massive drop in all accuracy of f1-score.

Classification Report:

	precision	recall	f1-score	support
Africa	0.23	0.30	0.26	1238
East_Asia	0.26	0.23	0.24	1304
Eastern_Europe	0.00	0.00	0.00	395
Middle_East	0.00	0.00	0.00	749
North_America	0.00	0.00	0.00	817
North_America_Mexico	0.00	0.00	0.00	383
Oceania	0.00	0.00	0.00	781
South_America	0.23	0.01	0.02	805
South_Asia	0.00	0.00	0.00	371
Southeast_Asia	0.00	0.00	0.00	782
Western_Europe	0.27	0.81	0.41	2375
accuracy			0.26	10000
macro avg	0.09	0.12	0.08	10000
weighted avg	0.14	0.26	0.16	10000

The confusion matrix also showed how things were bad, with the model mainly focusing on a certain class to throw its predictions upon. This may occur due to the model not being able to catch the complexity needed through the 16 neurons in its hidden layer.



After carrying out 5 tests, we found that the features and architecture used in test 2 yielded the most satisfactory results from the other test in comparison. Thus, we chose to maintain the feature list but manipulate the architecture for 2 more tests to compare and decide what our final model would be.

Test 6:

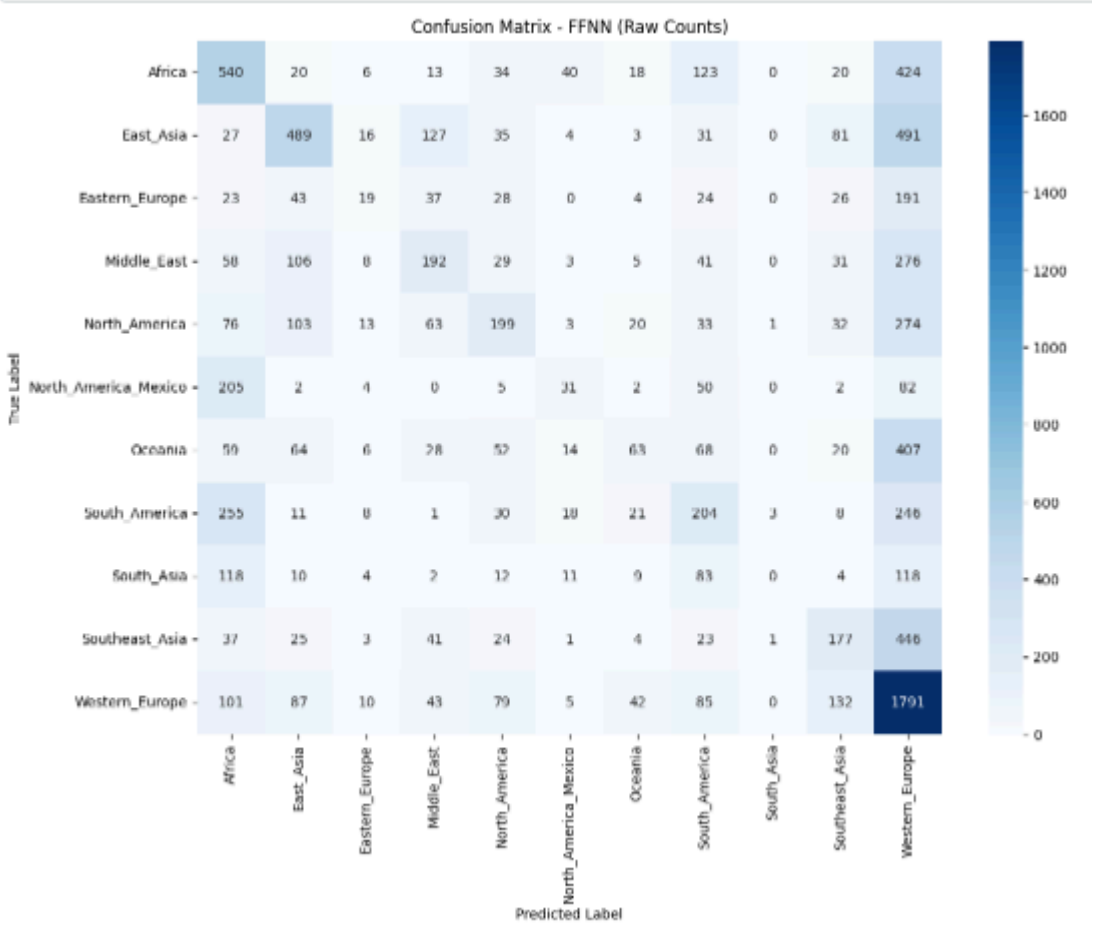
To attempt to push the limit a bit in hopes of better results, we altered the model's architecture by assigning its single hidden layer 32 neurons.

The results were extremely similar to those of test 2 with almost the same accuracy, and weighted average in f1-score.

Classification Report:

	precision	recall	f1-score	support
Africa	0.36	0.44	0.39	1238
East_Asia	0.51	0.38	0.43	1304
Eastern_Europe	0.20	0.05	0.08	395
Middle_East	0.35	0.26	0.30	749
North_America	0.38	0.24	0.30	817
North_America_Mexico	0.24	0.08	0.12	383
Oceania	0.33	0.08	0.13	781
South_America	0.27	0.25	0.26	805
South_Asia	0.00	0.00	0.00	371
Southeast_Asia	0.33	0.23	0.27	782
Western_Europe	0.38	0.75	0.50	2375
accuracy			0.37	10000
macro avg	0.30	0.25	0.25	10000
weighted avg	0.35	0.37	0.33	10000

The confusion matrix was also quite similar to the one of test 2:



To try and push the limits one last time, we carried out the last test: test 7.

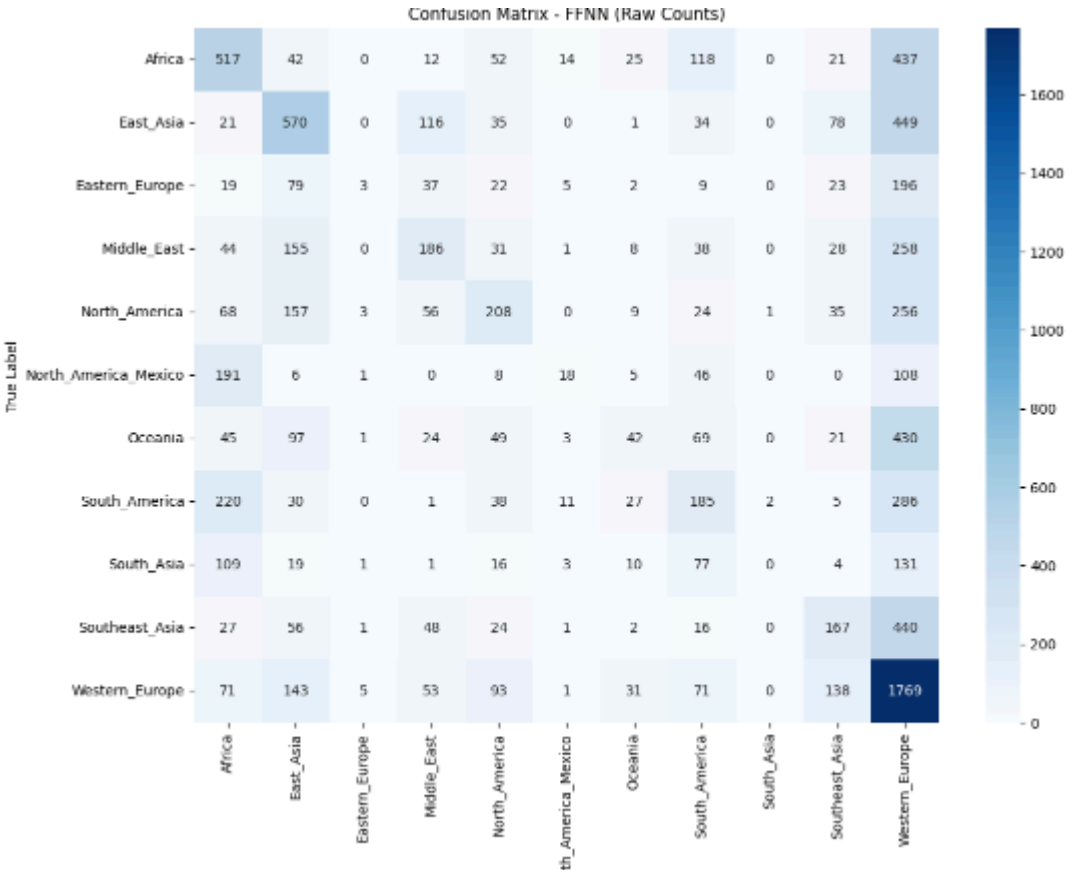
Test 7:

This time, we increased a hidden layer. The architecture test 7 used was composed of the inputs, a 32-neuron hidden layer, a 16-neuron hidden layer, and the output layer.
The results were also similar to test 6 and test 2 results, with a slight difference in the accuracy, and weighted average in f1-score.

Classification Report:

	precision	recall	f1-score	support
Africa	0.39	0.42	0.40	1238
East_Asia	0.42	0.44	0.43	1304
Eastern_Europe	0.20	0.01	0.01	395
Middle_East	0.35	0.25	0.29	749
North_America	0.36	0.25	0.30	817
North_America_Mexico	0.32	0.05	0.08	383
Oceania	0.26	0.05	0.09	781
South_America	0.27	0.23	0.25	805
South_Asia	0.00	0.00	0.00	371
Southeast_Asia	0.32	0.21	0.26	782
Western_Europe	0.37	0.74	0.50	2375
accuracy			0.37	10000
macro avg	0.30	0.24	0.24	10000
weighted avg	0.33	0.37	0.32	10000

The confusion matrix was also similar to the same test cases, but again with some slight differences (slightly worse results but not drastic).



The model we chose :

After over 10 tests (for Regression models and FFNN combined), we finally decided to proceed with the FFNN that has a single 32-neuron hidden layer and an output layer of 11 neurons because of the number of classes we have .
But before further discussion on the model, we need to elaborate on what FFNN itself is.

FFNN (FeedForward Neural Network) is an artificial neural network. It is called “FeedForward” as the information flows only in one direction, from input layer onto hidden layer(s) and then to output layer. The FFNN is straightforward when it comes to design and implementation. One downside is that a FFNN is heavily dependent on the quality of the input features. This is because FFNN models cannot automatically generate complexity themselves from bad data however FFNNs will understand complex data and thus rely on well-engineered features.

We chose the model with 1 layer and 32 neurons because when we ran the notebook many times , it scored constantly similar than the model with 2 layers (32 and 16) and it is more efficient as it uses less neurons and layers .
It also scored higher than the model with 1 layer (16) in some runs .
We also chose it for future proofing as 32 neurons would provide us with more complexity than just 16 neurons if we needed to introduce more complex problems in the future .
The model we chose also scored higher than any Logistic regression model we made .

Now that we have formally described what a FFNN is and what the chosen model’s architecture looks like, we need to move onto the explainability. FFNN in essence is easy to understand, but its predictions are not. Thus, the use of XAI is crucial to understand why the model performed either the overall predictions or performed

a certain way, or on certain/specific rows why it predicted the shown prediction.

XAI :

XAI, or explainable AI, is mainly used to describe/explain the predictions of the ‘blackbox’ models. As mentioned before, the model itself is easy to understand, its decisions aren’t. The use of XAI allows us to have an insight on the prediction made through understanding (using plots) the factors that affected the prediction. Some features may positively influence the prediction while others may negatively influence it, and that’s what we see through the lens of XAI when using tools like SHAP (for global explanation) and LIME (for local or specific predictions on specific rows/data entries).

SHAP:

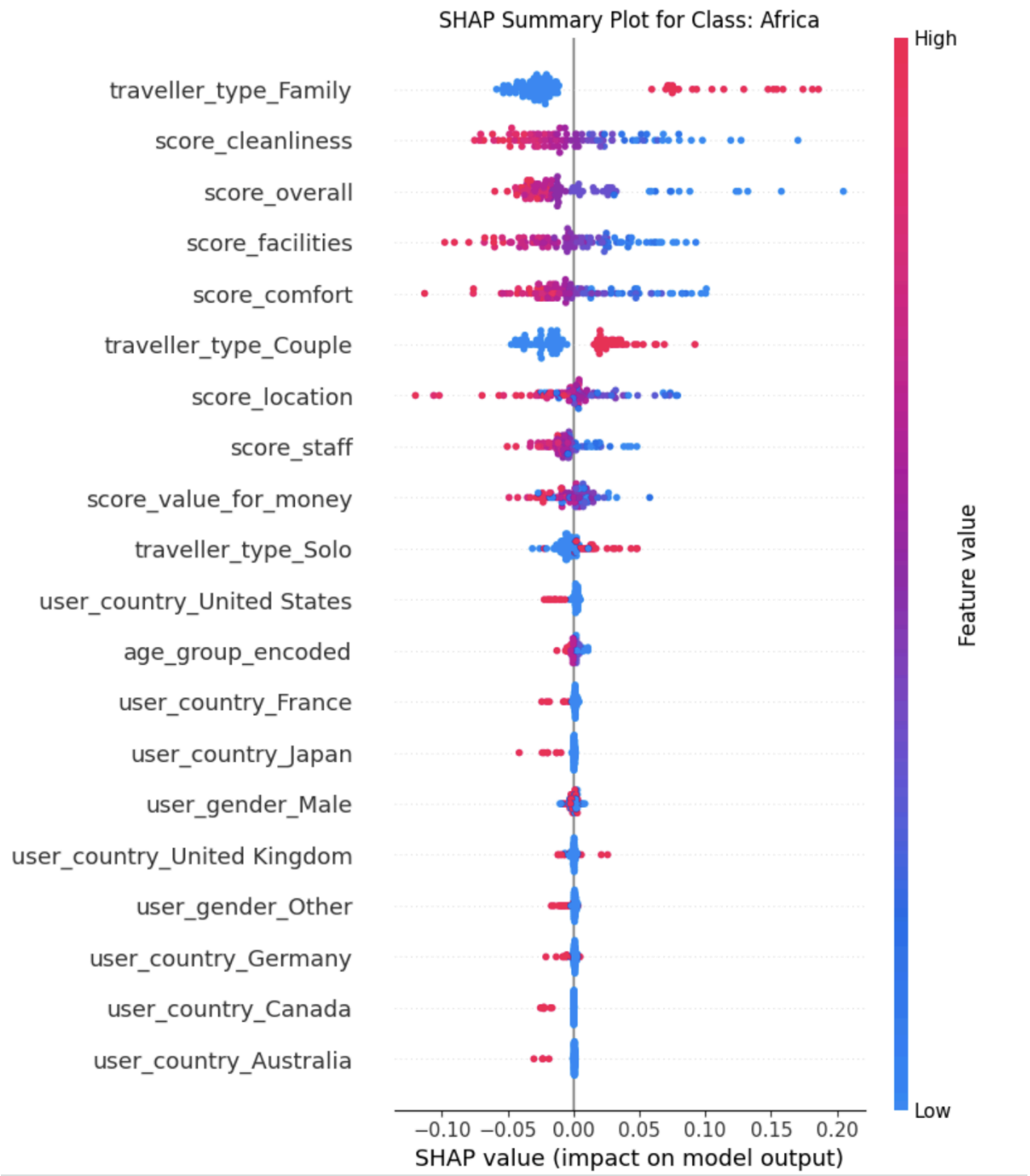
SHAP graphs/plots consist of: X-axis, Y-axis, Dots, and Colors.

- The Y-axis lists the features of the model, the order from top to bottom highlights the importance of each feature. The higher the feature, the more important it is.
- The X-axis is the SHAP value itself, how impactful a feature is to the output.
- The Dots represent rows or predictions from the dataset.
- The Colors show the actual value of the feature for a specific prediction. Blue dots indicate low feature values, Red dots indicate high feature value.

Now, we can start exploring the plots.

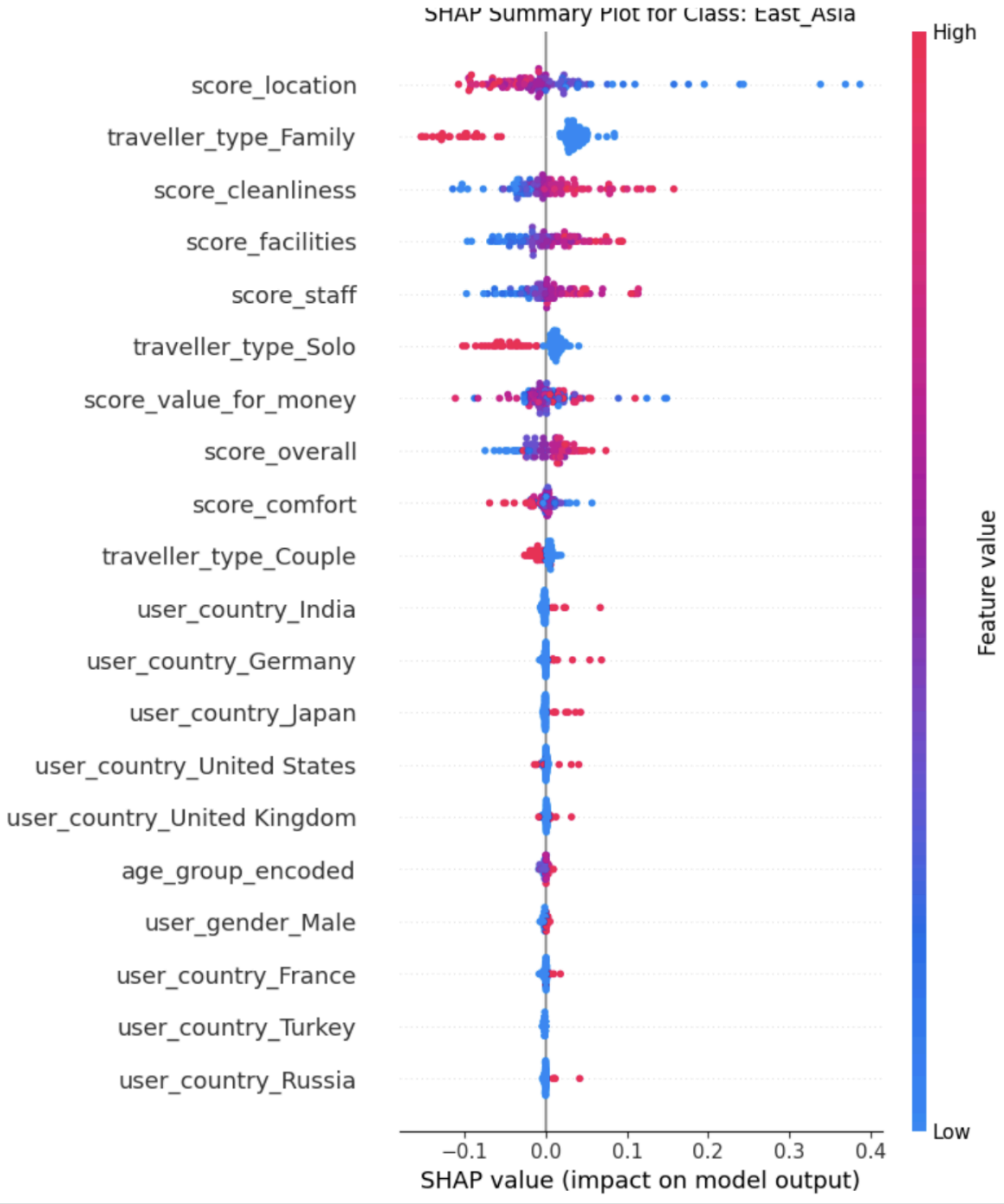
SHAP plots :

We used SHAP for global explanation to understand the influence of each feature on each class of country_groups so that’s why we will attach 11 plots as there are 11 country_groups



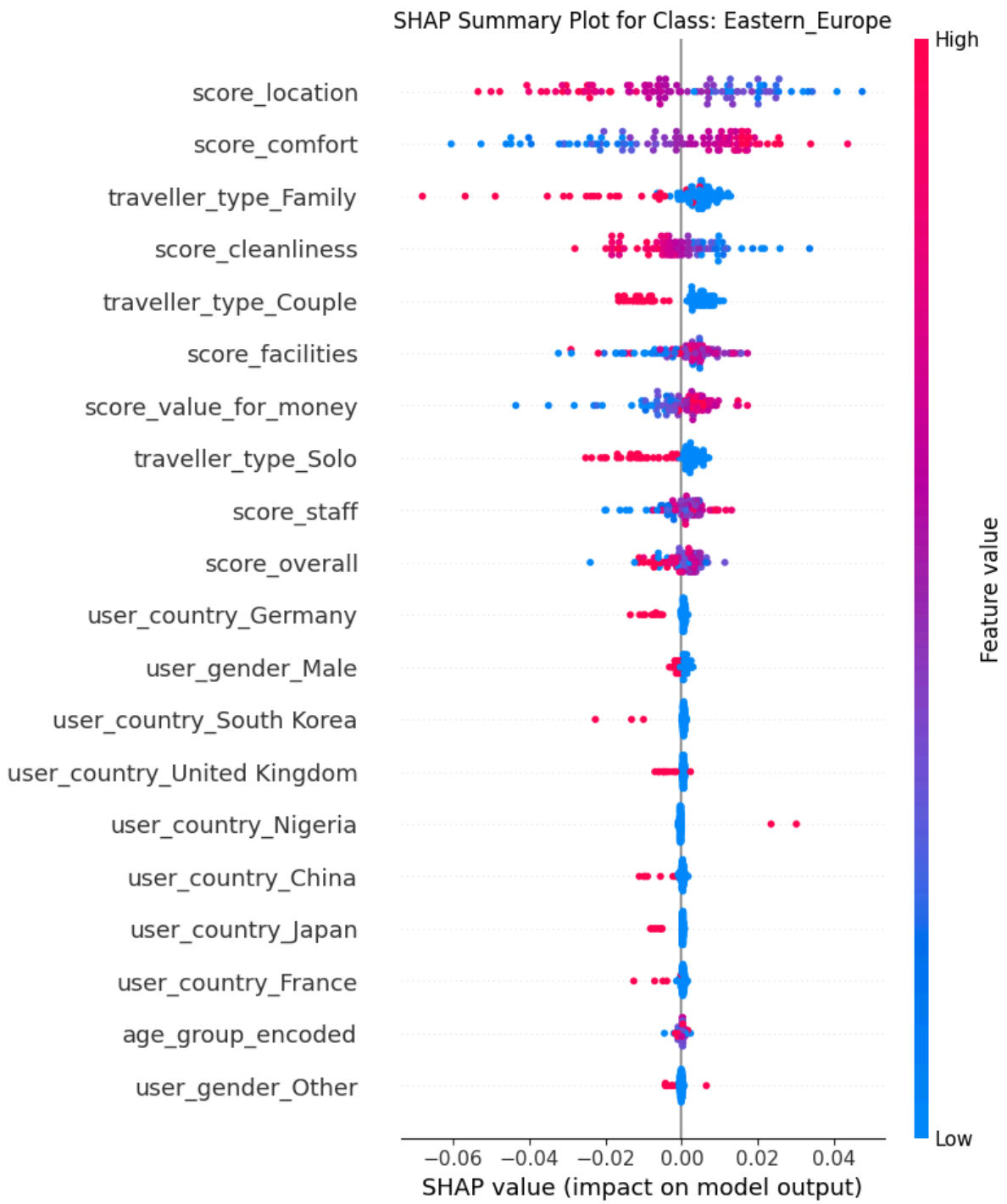
Observation and explanation of plot 1:

When the model predicts “Africa”, it was mainly affected by the traveller_type_family, score_cleanliness, and score_overall. If the model finds that the traveller type is most likely a family, it strengthens its confidence in the prediction being “Africa”. Also, if the scores of cleanliness and overall score tend to be low, the model is more likely to predict “Africa”.



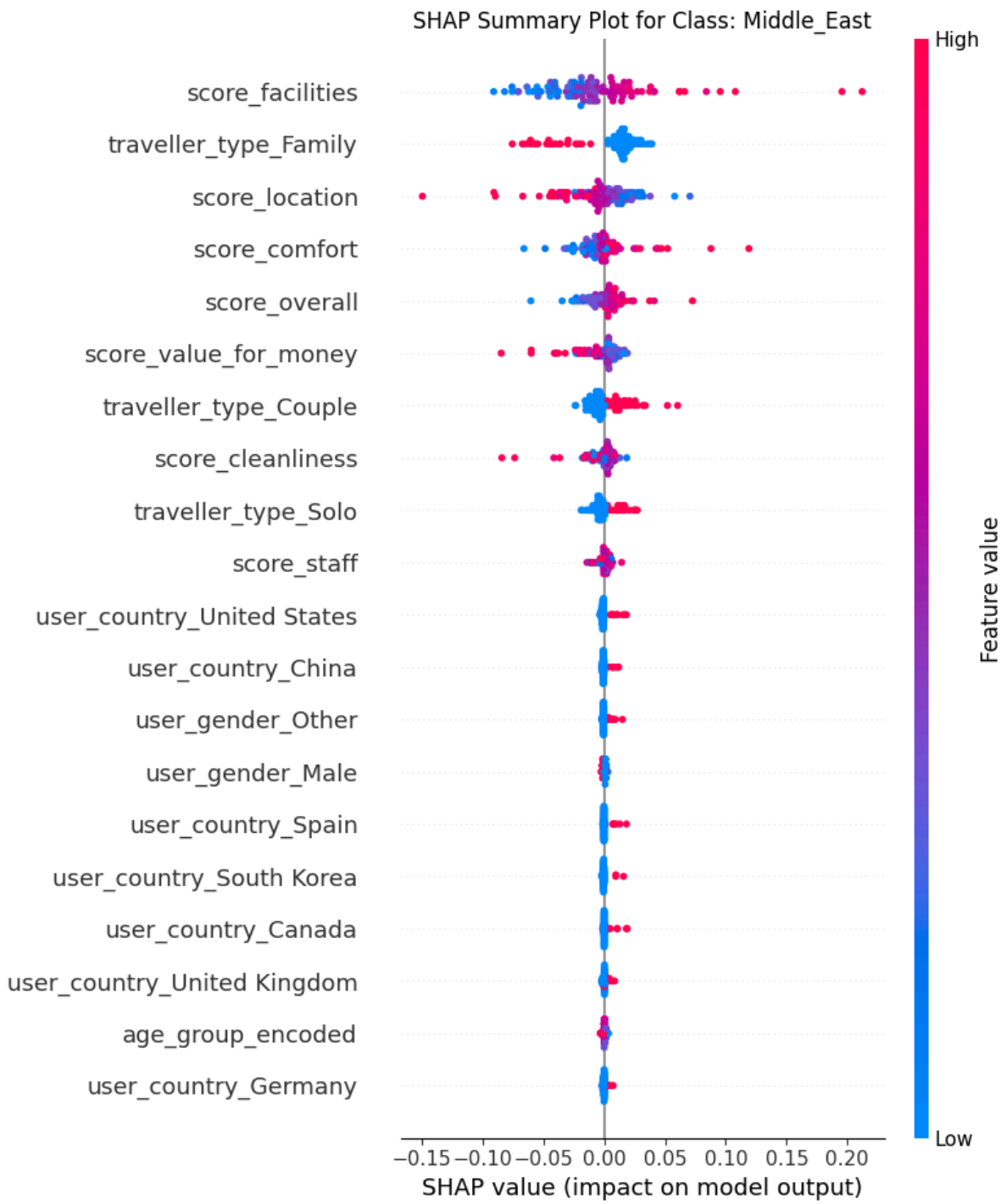
Observation and explanation of plot 2:

For “East_Asia” predictions, the plots show that the model would have a high probability of predicting it if the score_location is low, the traveller_type is not a family, and if the score_cleanliness is high, along with other features that also affect the prediction.



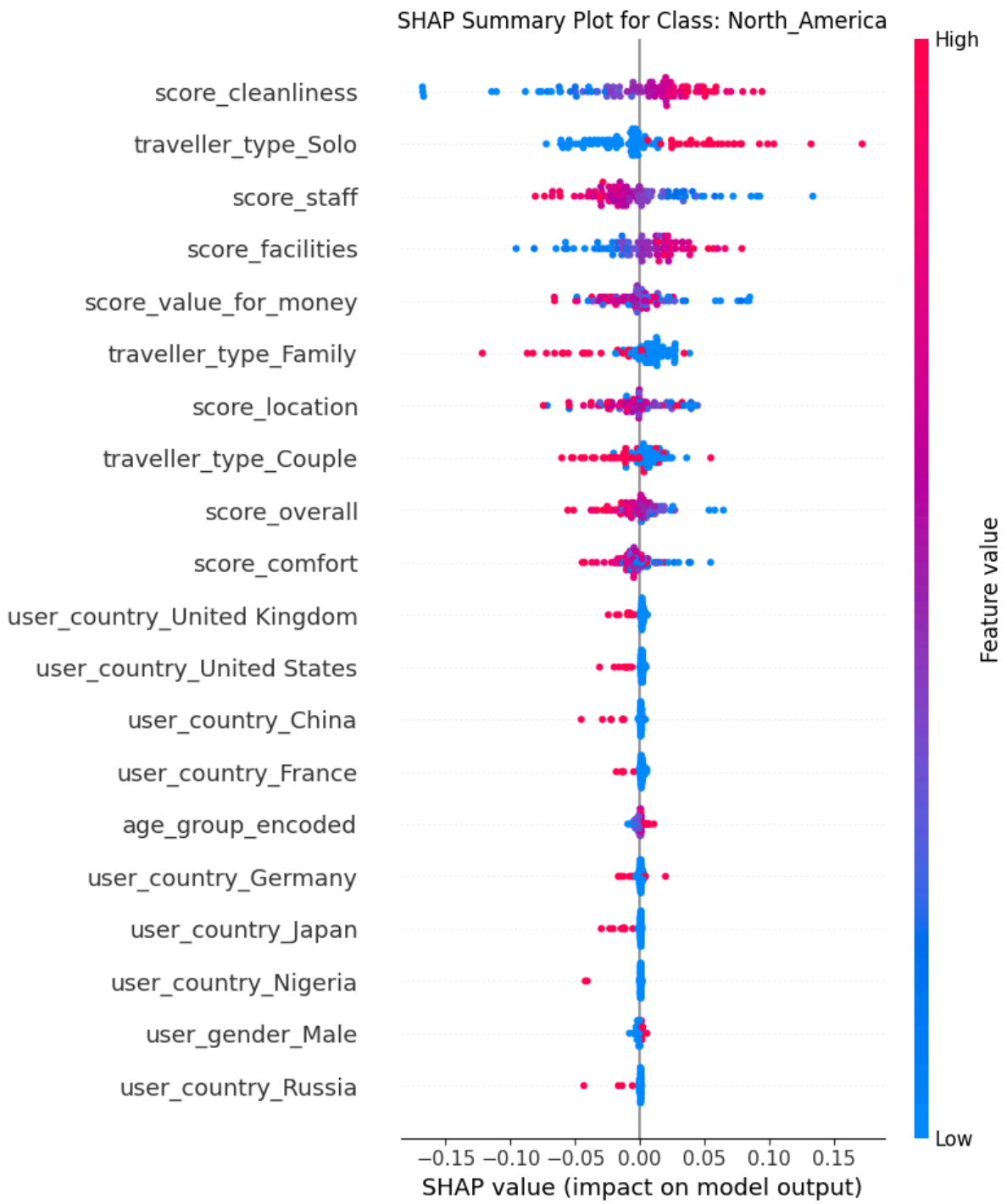
Observation and explanation of plot 3:

For “Eastern_Europe” prediction, it is more likely to get predicted if the score_location is low, the score_comfort is high, and the traveller_type is not a family.



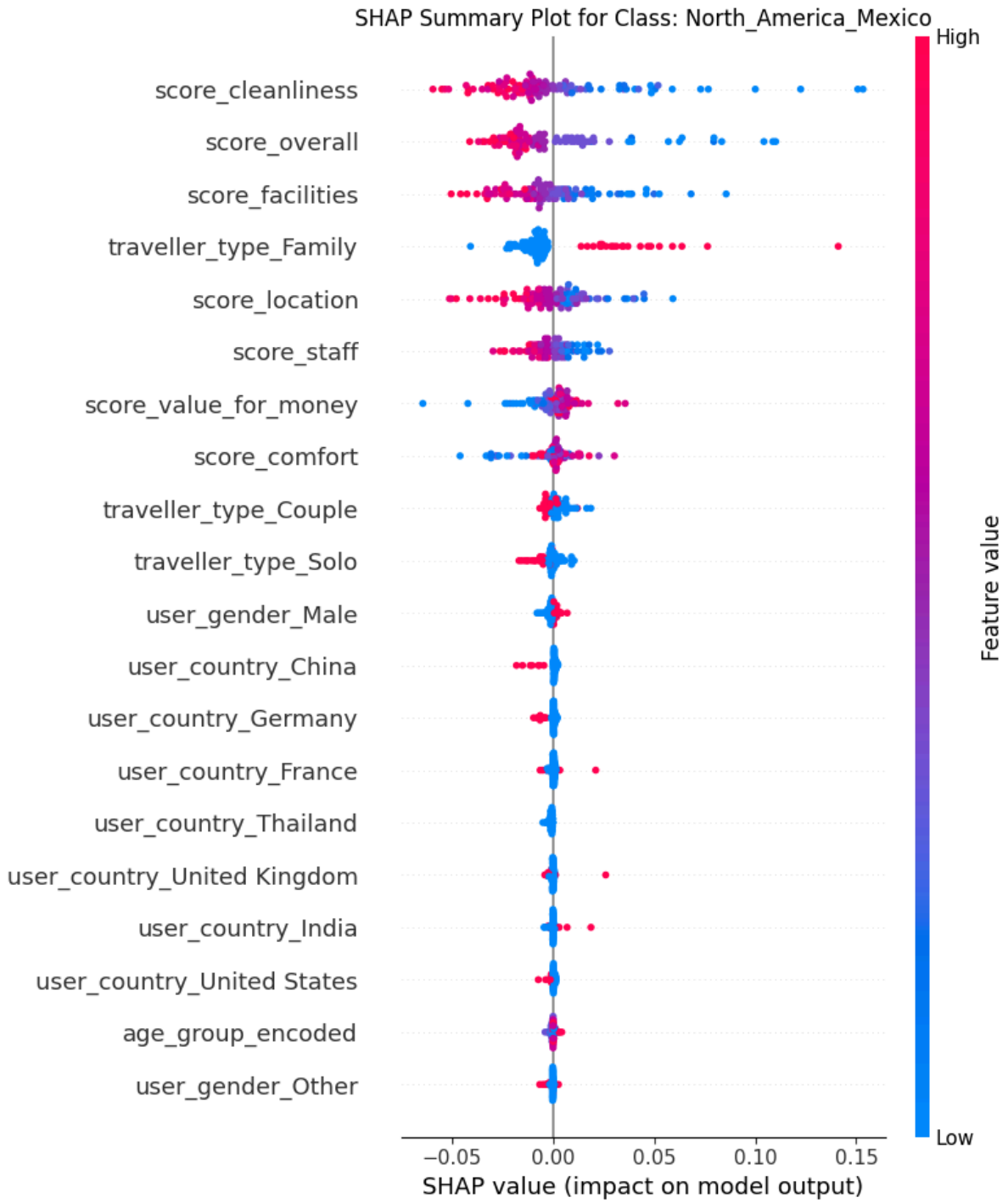
Observation and explanation of plot 4:

When the prediction is “Middle_East”, the prediction is most likely based on high scores of score_facilities, the traveller_type is not a family, and the score_location is low.



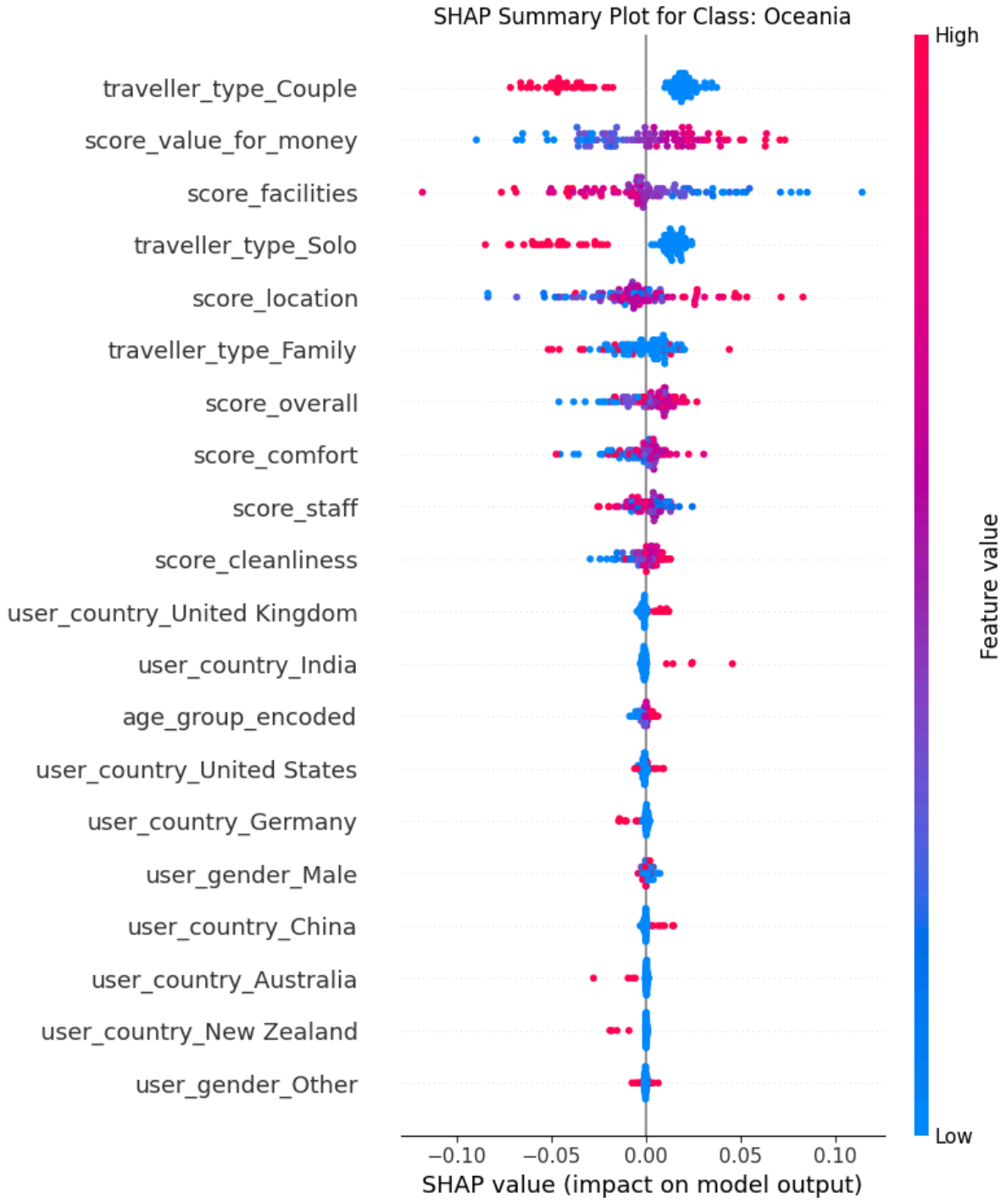
Observation and explanation of plot 5:

For “North_America” prediction, the prediction has a higher probability of being predicted if the score_cleanliness is high, the traveller_type is a Solo traveller, and the score_staff is low.



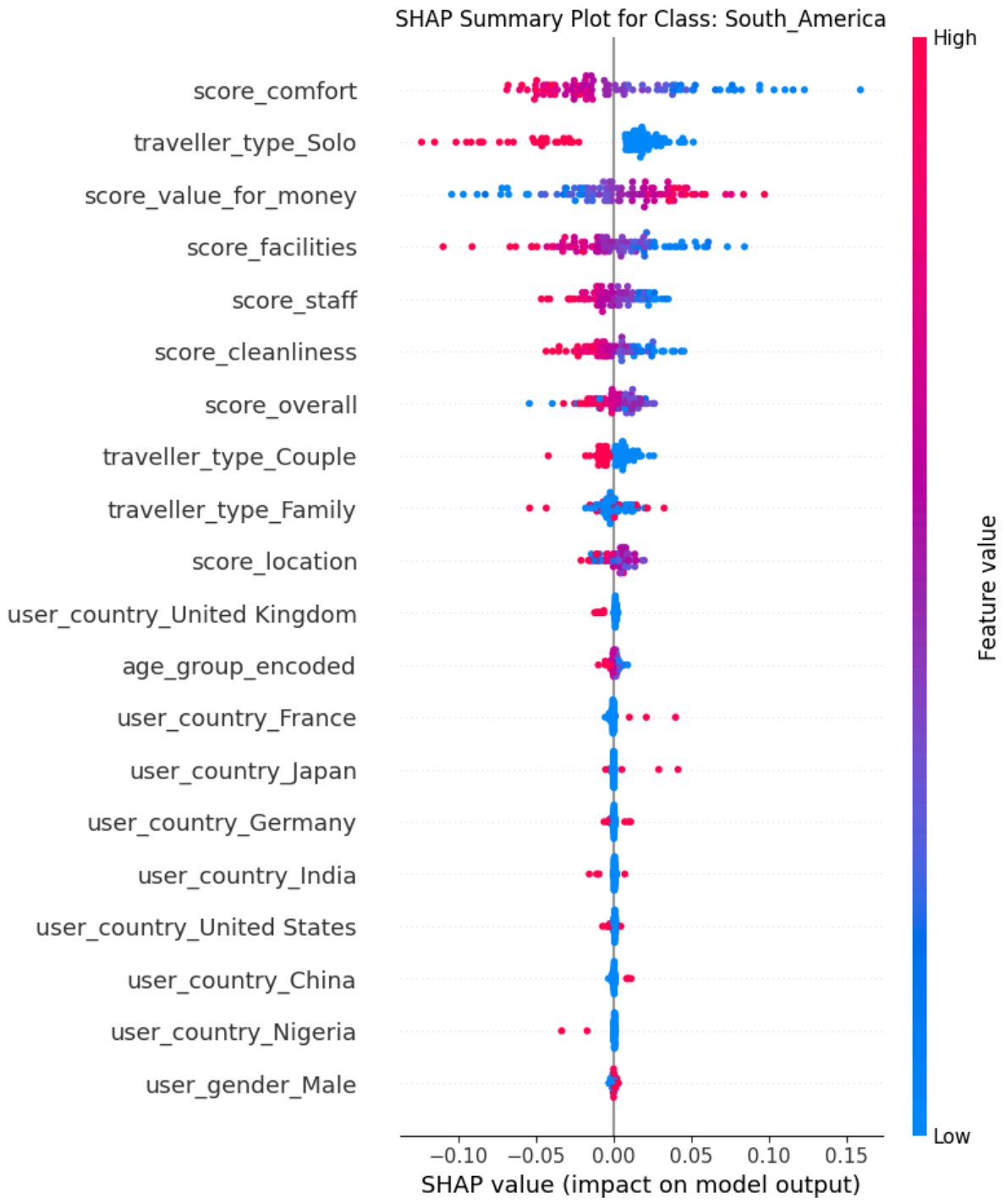
Observation and explanation of plot 6:

“North_America_Mexico” has a high chance of being predicted if score_cleanliness is low, score_overall is low, and score_facilities is also low.



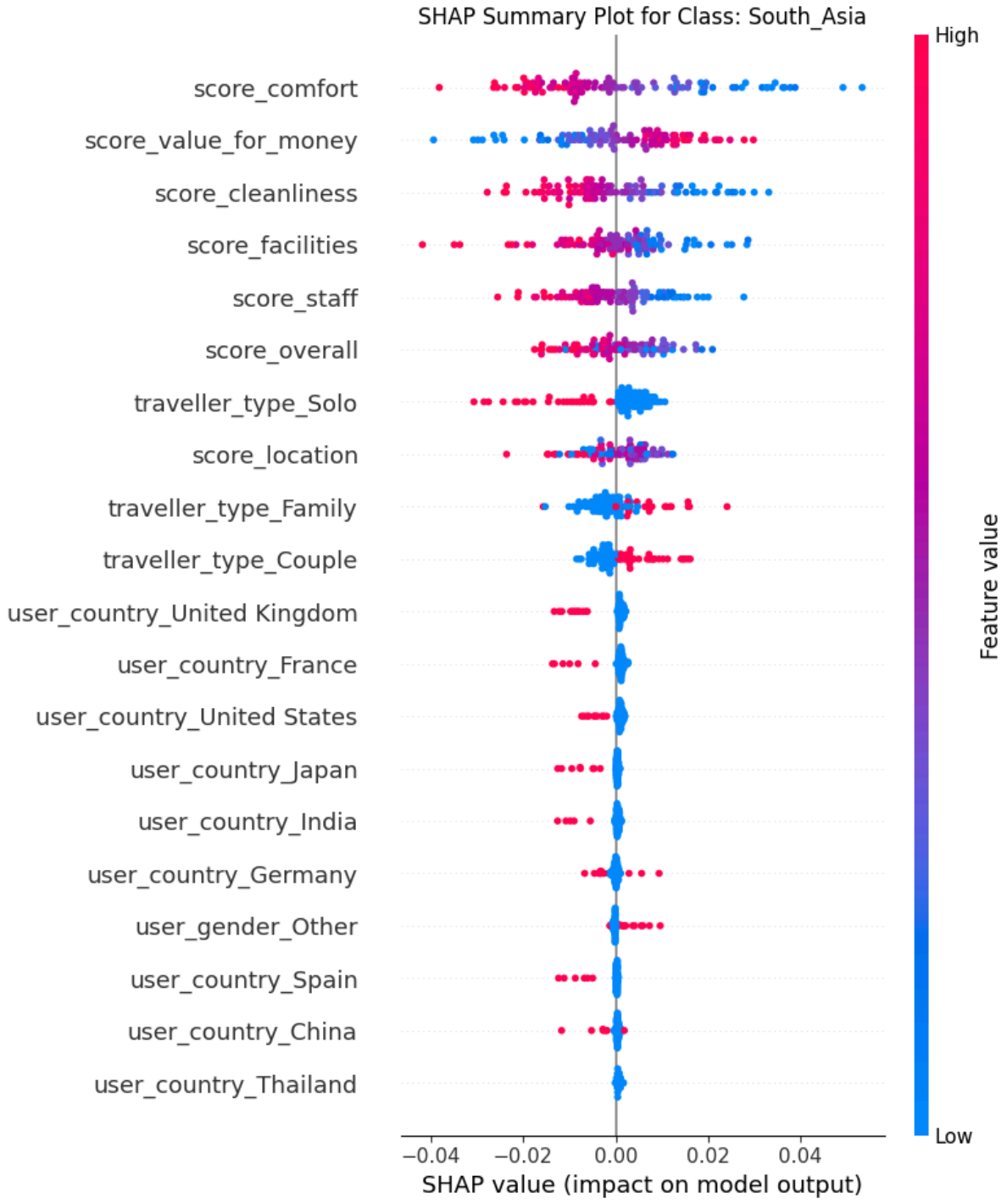
Observation and explanation of plot 7:

“Oceania” has a high chance of being predicted if the traveller_type is not a Couple, the score_value_for_money is high, and score_facilities is low.



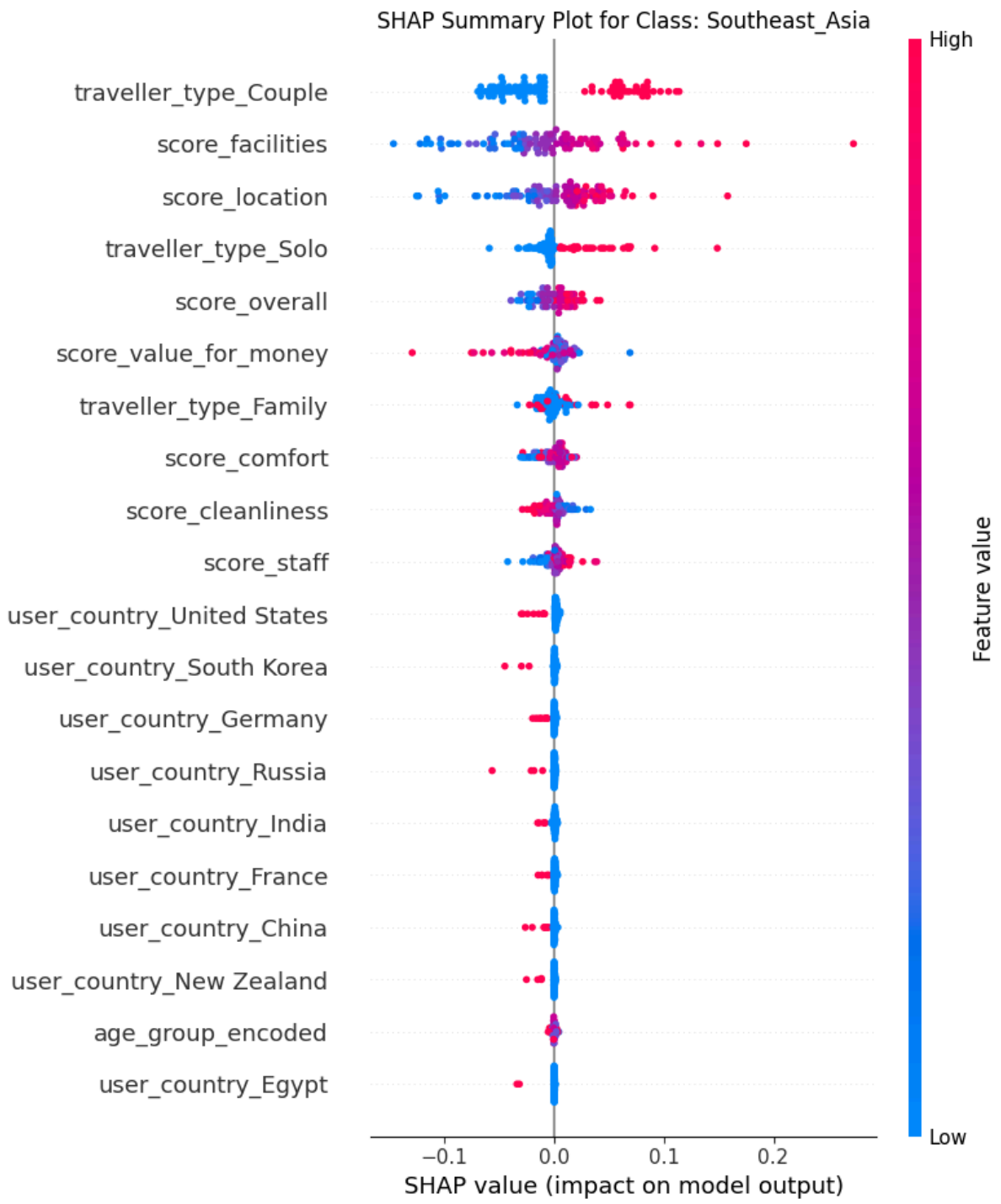
Observation and explanation of plot 8:

For “South_America”, the prediction has a higher probability of being predicted if the score_comfort is low, the traveller_type is not a Solo traveller, and the score_value_for_money is high.



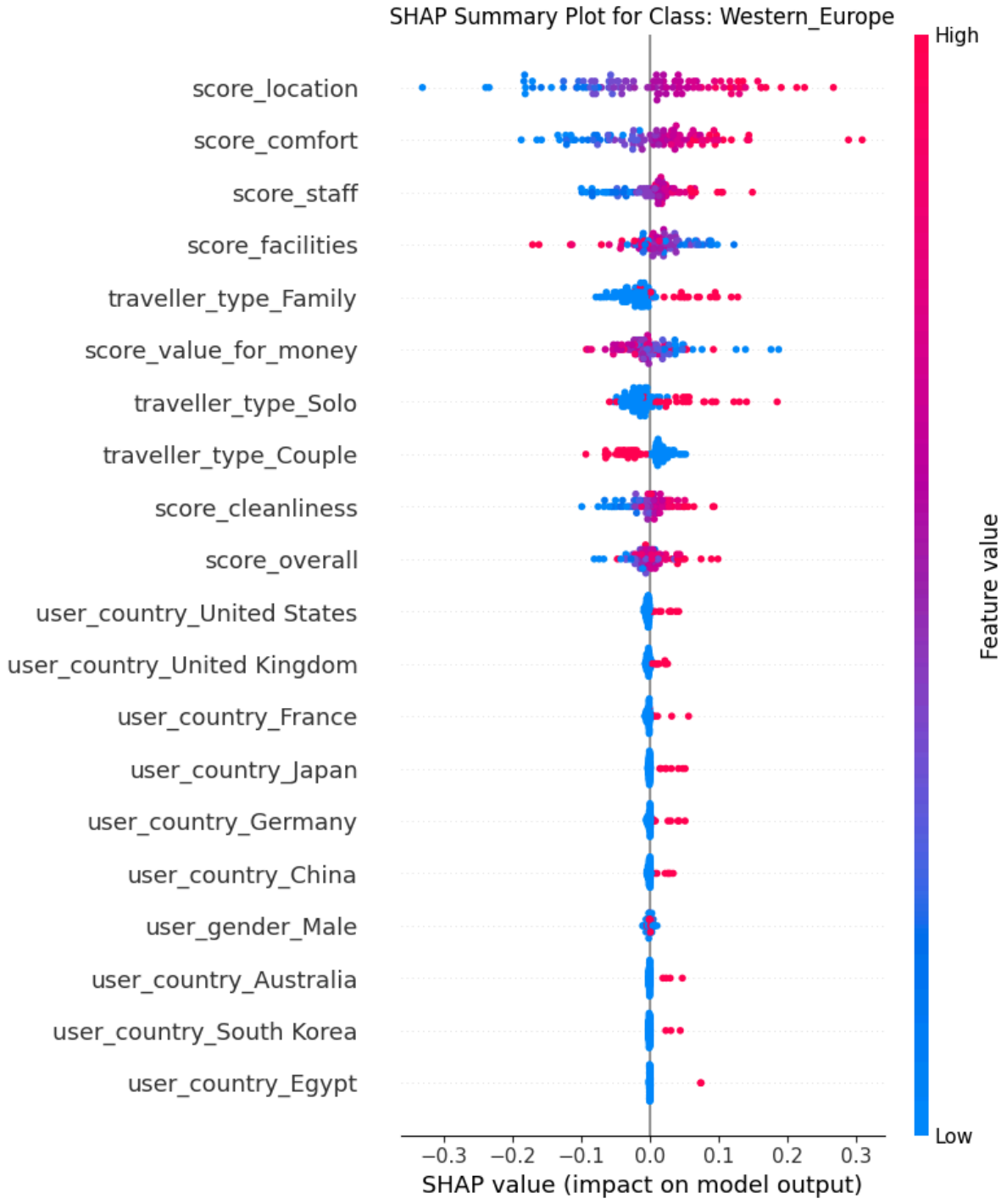
Observation and explanation of plot 9:

“South_Asia” has a higher probability of being predicted if score_comfort is low, score_value_for_money is high, and if score_cleanliness is low.



Observation and explanation of plot 10:

“Southeast_Asia” is more likely predicted if the traveller_type is a Couple, the score_facilities is high, and the score_location is also high.



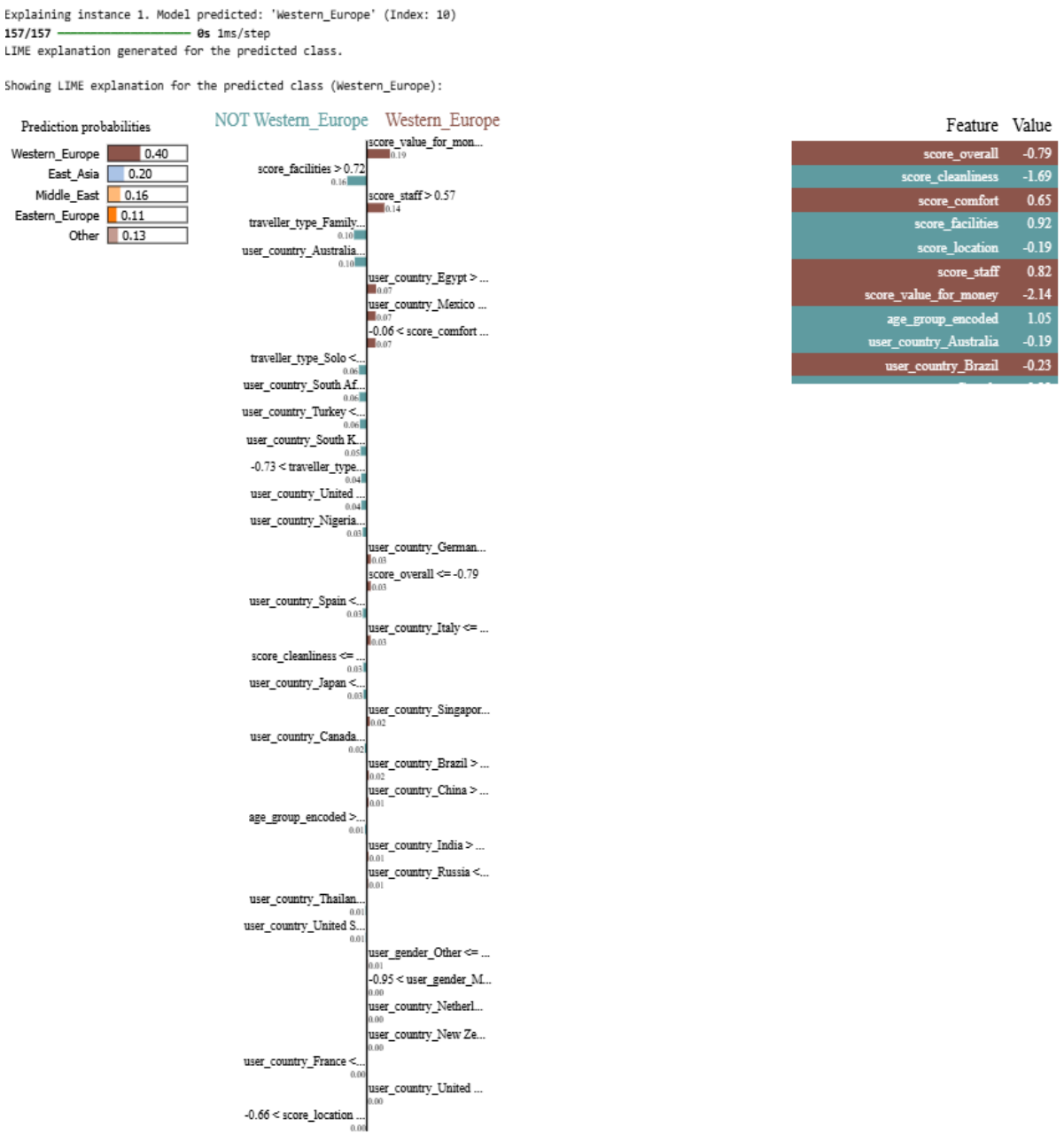
<u>Observation and explanation of plot 11:</u>

Finally, “Western_Europe” had a high probability of being predicted when score_location, score_comfort, and score_staff were all high.

The predictions were not limited to the features highlighted in the observations and explanations of the plots, the features highlighted were used as a simple example of part of the explanation.

LIME:

LIME is used (as mentioned before) to explain a certain or specific prediction, not the overall prediction criteria. It explains why this specific row had this specific prediction through some features it deems important.



Observation and explanation of LIME Plot:

For this specific row 1, the model predicted “Western_Europe”. “Western_Europe” had the highest probability, followed by (in order) “East_Asia”, “Middle_East”, “Eastern_Europe”, then the rest.

The prediction of the model was mainly affected by the low “value for money” scores by users which pushed the model to predict western_europe .

A high “score_staff” also pushed the model to predict western_europe.

The values on the left side of the vertical axis pushed the model to predict something other than “Western_Europe”.

A high “score_cleanliness“ contributed a lot in shifting the prediction of the model away from western_europe

***It happened in some runs that this prediction changed to east asia but we didn’t report on it as the last run before finalizing the report , it was predicting western-europe**

Inference function :

We wrote a function that takes as input the same variables taken for our model and predicts the country_group in text .
The function can be checked in the python notebook .

Resources:

Any links we referred to

- https://www.reddit.com/r/datascience/comments/1bdc8iy/what_are_the_general_checklist_of_data_cleaning/
- https://stackoverflow.com/questions/60796142/should-i-get-the-same-accuracy-in-the-test-set-and-the-training-set