# Machine Learning Project Report: Ranking Tourism Activities to Maximize Profit

Avram Tudor and Boldeanu Razvan

January 17, 2025

## Contents

# 1 Introduction

In this report, we address the problem of recommending a hierarchy (ranking) of tourism activity categories for a hotel chain planning to open a new hotel in a specific country. The ultimate goal is to maximize either the total revenue (`Revenue`) or the revenue per visitor (`Revenue_per_visitor`).

The dataset provided (`tourism_dataset.csv`) contains seven columns:

- **Location**: an identifier for the place;
- **Country**: which country the location is in;
- **Category**: the tourism activity category (Nature, Historical, Cultural, Beach, Adventure, Urban);
- **Visitors**: total number of visitors;
- **Rating**: average rating (1–5 scale);
- **Revenue**: total revenue from that activity;
- **Accommodation_Available**: indicates if accommodation is available (Yes/No).

We treat this as a supervised learning task to predict `Revenue_per_visitor`, then rank the categories by their predicted outcome for a chosen country.

# 2 Dataset Understanding

## 2.1 Basic Exploration and Correlations

We first explored the dataset by inspecting its structure:

- `df.head()` shows the first 5 rows.
- `df.info()` reveals there are 5989 rows (after cleaning or dropping any invalid rows).
- `df.describe()` reports summary statistics of numerical columns (`Visitors`, `Rating`, and `Revenue`).

A notable observation is the direct relationship between `Visitors` and `Revenue`, which hints that more visitors often lead to higher revenue. However, to normalize the effect of simply having a large visitor count, we introduce `Revenue_per_visitor` as our target for modeling.

## 2.2 Feature Engineering

We create a new feature,

$$\texttt{Revenue\_per\_visitor} = \frac{\texttt{Revenue}}{\texttt{Visitors}},$$

which directly measures profit per visitor. We also label-encode the categorical columns (`Country`, `Category`, `Accommodation_Available`) for input into our chosen ML algorithms.

# 3 Methodology

The main steps in our methodology are:

1. **Load and explore** the dataset (as described above).
2. **Feature engineering**: create `Revenue_per_visitor`, label-encode categories, handle missing values.
3. **Split** the data into train and test sets using an 80%-20% ratio (common practice in ML).
4. **Choose algorithms** to learn a regression model mapping $\{\text{features}\} \rightarrow$ `Revenue_per_visitor`.
5. **Evaluate** multiple candidate algorithms using MSE and $R^2$ on the test set.
6. **Select the best model** and use it to **rank the 6 categories** for a specified country (e.g., France).

# 4 Algorithms Selection and Implementation

## 4.1 Candidate Algorithms

Chosen methods:

- **Linear Regression (LR)**: a simple, interpretable baseline that assumes linear relationships among features.
- **Random Forest (RF)**: an ensemble method of decision trees, which can model complex, non-linear interactions.

## 4.2 Theoretical Justification

- **Linear Regression** often underperforms if the data's relationship is not well-modeled by linear functions. However, it is fast, interpretable, and easy to implement.
- **Random Forest** handles non-linearities and interactions among features better, potentially yielding improved accuracy. However, it is less interpretable and can be slower for very large datasets.

## 4.3 Implementation Outline

We use Python's `scikit-learn` library for model training. The main lines are:

```
lin_reg = LinearRegression()
lin_reg.fit(X_train, y_train)
y_pred_lin = lin_reg.predict(X_test)

rf = RandomForestRegressor(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)
y_pred_rf = rf.predict(X_test)
```

We then compare Mean Squared Error (MSE) and $R^2$ for each model on the test set.

# 5 Experimental Results and Comparisons

## 5.1 Train–Test Split

We use an 80%-20% split:

- Training set size: 4791 rows
- Test set size: 1198 rows

This ensures enough data for training while reserving a portion for unbiased testing.

## 5.2 Performance Metrics

We compute:

- **MSE**: Mean Squared Error between predicted and true `Revenue_per_visitor`.
- $R^2$: Coefficient of Determination.

Result on the test set:

| **Model** | **MSE** | $R^2$ |
|---|---|---|
| Linear Regression | 271.2306 | 0.0843 |
| Random Forest | 154.4139 | 0.4787 |

Table 1: Test Set Performance of LR vs. RF

Clearly, **Random Forest** achieves a higher $R^2$ and a significantly lower MSE, indicating that it predicts `Revenue_per_visitor` more accurately than Linear Regression.

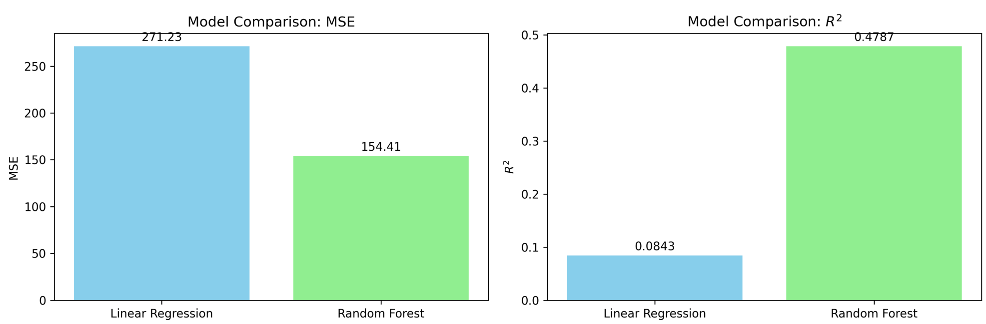


Figure 1: A sample bar chart comparing the MSE (left bars) and $R^2$ (right bars) for LR vs. RF. This visual indicates how Random Forest outperforms Linear Regression.

## 5.3 Final Model Choice

Based on these metrics, we select **Random Forest** as our final model.

# 6 Inference and Ranking

After selecting Random Forest, we rank the 6 activity categories for a fixed country (e.g., "France"). We:

1. Label-encode the country name (France) to the numerical form recognized by the model.
2. For each possible category label (0–5), we set some typical inputs (e.g., `Accommodation_Available`=Yes, `Visitors`=1000, `Rating`=3.0).
3. Predict `Revenue_per_visitor` for each category.
4. Sort the categories in descending order.

Ranking result:

```
1. Nature       -> 362.06
2. Urban        -> 354.80
3. Historical   -> 345.23
4. Beach        -> 307.76
5. Cultural     -> 307.28
6. Adventure    -> 306.88
```

Thus, **Nature** is predicted to yield the highest profit per visitor in France, followed by **Urban** and so on.

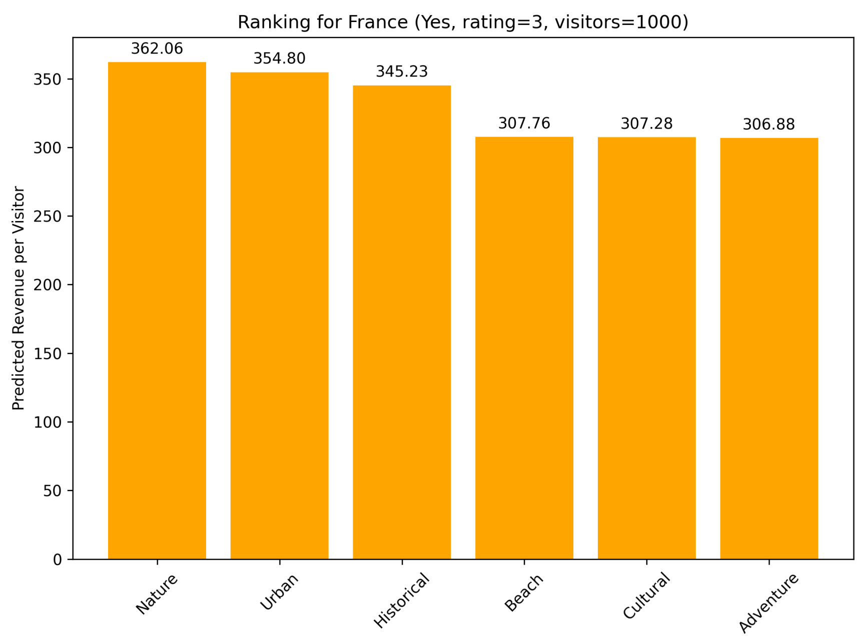


Figure 2: Bar chart illustrating the predicted `Revenue_per_visitor` for each category, under a hypothetical scenario of 1000 visitors, a rating of 3.0, and accommodation availability in France.

## 7 Conclusions and Future Work

### 7.1 Conclusions

- We successfully modeled `Revenue_per_visitor` using a Random Forest, which outperformed a baseline Linear Regression approach.
- The final inference step generates a ranking of categories for a chosen country, helping hotel owners prioritize the most profitable activities.

### 7.2 Possible Improvements

- **Hyperparameter tuning**: We could optimize the number of trees, maximum depth, or other Random Forest parameters.
- **Feature engineering**: Introduce more domain-specific features (e.g., local economy indicators, seasonal factors).
- **Explainability**: Use SHAP or feature-importance methods to better understand how each feature impacts the model's predictions.

## 8 References


1. Scikit-learn Documentation: `https://scikit-learn.org/stable/`
2. Kaggle Tourism Dataset: `https://www.kaggle.com/datasets/umeradnaan/tourism-dataset`
3. Github repository `https://github.com/helio18/ML_AP2`