Exploring and Modeling Tourism Trends in Ireland:
An End-to-End Data Analysis

Author: M. Vitucci
e-mail: sba24277@student.cct.ie
Student ID: 24277

Total Word Count: ~4300

Abstract

This study explores the factors shaping tourism events in Ireland, drawing from multiple data sources to examine trends in pricing, seasonality, and location. Through careful data preparation—addressing missing values, outliers, and feature engineering—a robust analysis was conducted. Predictive modeling revealed that booking platforms and venue types play a key role in determining event prices, with XGBoost emerging as the top-performing model, explaining more than half of the price variance. Additionally, inferential statistics, including binomial, Poisson, and normal distributions, provided deeper insights into patterns like the likelihood of free events and pricing trends. The findings highlight opportunities for refining features and improving models to enhance predictive accuracy and practical applications.

Keywords

Predictive modeling, Inferential statistics, Feature engineering, CRISP-DM framework, Binomial distribution, Poisson distribution, Normal distribution, Robust scaling, XGBoost, Random Forest, Box-Cox transformation, Spearman correlation, Shapiro-Wilk test

1 Introduction

Tourism events play a key role in fostering community and driving economic growth in Ireland, a country celebrated for its history and landscapes. This report investigates key questions about tourism events using both predictive modeling and inferential statistics to reveal insights:

- Do larger counties host more free events to engage wider audiences?
- Are free events more common in certain seasons, and what factors influence their occurrence?
- What factors drive event pricing, and how do these vary across different contexts?
- How do geographic, seasonal, and demographic trends shape event availability and characteristics?

Through data analysis and statistical techniques, this report explores how location, season, population, and other factors influence event costs, availability, and characteristics.

2 Materials & Methods

2.1 Software and Libraries

All processing, analysis, and modeling were done in Python, primarily within Jupyter Notebook for an organized, interactive workspace. The following libraries and frameworks supported data processing, analysis, and visualization:

- pandas, numpy, and scipy: Essential for efficient data manipulation, handling of missing values, filtering, numerical operations, advanced statistical analysis, and mathematical computations.
- matplotlib, seaborn, squarify, and scienceplots: Used for creating visualizations, with scienceplots applying scientific styling to improve readability and presentation quality (scienceplots, n.d.), squarify to create treemap with matplotlib.

- scikit-learn and XGBoost: Provided machine learning algorithms, data preprocessing, model evaluation, and advanced gradient boosting.
- pydot: Used to visualize decision trees, adding model interpretability.
- haversine: Utilized to calculate the distance between geographic coordinates.
- nbconvert and pandoc: Automated Jupyter Notebook conversion into report formats for consistent documentation.

The report aligns directly with notebook sections, ensuring a seamless 1:1 correspondence for clarity and organization. ChatGPT was utilized for brainstorming ideas, refining concepts, organizing thoughts and assisting with markdown formula implementation (OpenAI, 2024).

2.2 Data Collection

The analysis relied on four key datasets, each bringing useful insights to help make sense of trends and patterns in Ireland's tourism events:

- Events dataset: Sourced from Fáilte Ireland, this dataset provided comprehensive information on event details across the country.
- Population data: Provided by the Central Statistics Office (CSO), this data added demographic context to the analysis.
- Tourism statistics: This dataset, also from the CSO, captured monthly trends in foreign visitors, providing a basis for seasonal analysis.
- Venue information: Collected from the Google Places API, this data enriched the dataset with details on venue types, ratings, and popularity metrics.

Dataset and table details are outlined in Appendix A. Data Collection. The data dictionary in

Appendix B. Data Dictionary offers detailed descriptions of the final cleaned dataset, capturing each column's role, type, and measurement level after pre-processing and cleaning steps.

2.3 Programming Paradigms

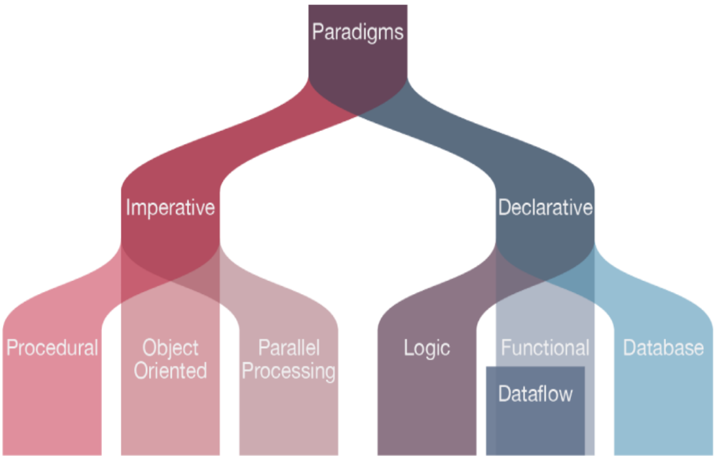


Figure 1: Paradigms hierarchy (Source: WatElectronics, n.d.)

The programming paradigms in Figure 1 highlight two main types: imperative and declarative, each with unique principles suited to specific tasks. This project primarily used a procedural approach to structure logical steps from data cleaning to modeling, ensuring a clear, step-by-step workflow. Functional programming was applied to create reusable functions for data transformations and visualizations, improving code readability and efficiency; future scalability could be enhanced with configuration files for parameter control. Although object-oriented programming (OOP) was minimally used in this project, it could enhance future workflows by organizing tasks like pre-processing, feature engineering, and model evaluation into reusable classes. Other paradigms like logic programming and parallel processing are less relevant for this type of analysis, as they are either unsuited for statistical tasks or typically managed by specialized libraries like Dask or Spark.

2.4 Project Management Framework

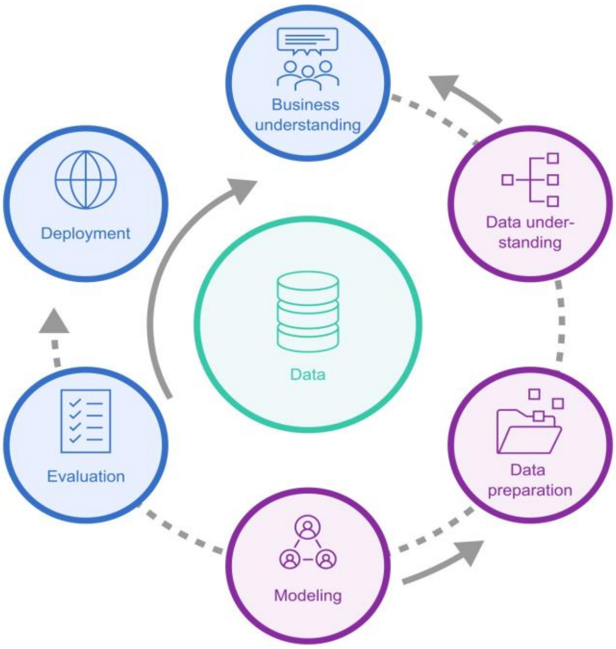


Figure 2: CRISP-DM Process Phases (Source: Fraunhofer Institute for Surface Engineering and Thin Films IST, n.d.)

For this data science project, the CRISP-DM (Cross-Industry Standard Process for Data Mining) framework (Figure 2) was selected due to its structured, iterative process, which aligns well with the dynamic nature of data science (Marbán, Mariscal, and Segovia, 2009). CRISP-DM consists of six phases allowing for flexibility and refinement throughout the project:

- Business Understanding: This phase focused on defining project goals, such as identifying factors that influence event popularity and pricing.
- Data Understanding: Initial data exploration and quality assessment allowed for a comprehensive view of event and tourism data.
- Data Preparation: Data cleaning, transformation, and feature engineering were performed iteratively. This included handling missing values and creating features, like seasonal indicators, essential for accurate modeling.
- Modeling: CRISP-DM's iterative modeling phase enabled experimenting with different algorithms, refining the models based on evaluation results, and optimizing predictions for event pricing.
- Evaluation: Models were evaluated against project goals, allowing adjustments to improve insights.
- Deployment: Although full deployment was not required, results were structured for practical application, such as in reports or dashboards for stakeholders.

CRISP-DM's flexibility, focus on business goals, and iterative approach make it ideal for data science projects. Unlike SEMMA (which centers on data mining) or KDD (more research-focused), CRISP-DM emphasizes aligning each phase with actionable insights, making it highly suitable for this analysis.

Supervised learning techniques, were chosen due to the availability of labeled data, making them suitable for predicting both continuous and categorical outcomes.

Unsupervised and semi-supervised techniques were deemed less relevant, as the analysis focused on predictive accuracy using a fully labeled dataset.

2.5 Tufts Principles

The visualizations in this project were designed following Tufte's principles to ensure clarity, accuracy, and effective data communication. Proportional representation and clear labeling were prioritized to prevent misinterpretation, and consistency in design highlighted data variation rather than unnecessary stylistic differences. Non-essential elements, or "chartjunk," were minimized, maximizing the data-ink ratio and focusing attention on the data itself. Each graphic was tailored to match the natural structure of the data, with appropriate aspect ratios chosen for readability (Tufte, 2007).

3 Exploratory Data Analysis (EDA)

3.1 Data Cleaning

To prepare the event dataset for analysis, essential preprocessing steps were undertaken:

- Rename Columns: Standardized column names to ensure consistency and ease of reference.
- Drop Irrelevant Columns: Removed contact_phone and booking_phone columns, as they are purely administrative and do not contribute to the statistical or predictive goals of the analysis.
- Remove Duplicate Rows: Ensured each event entry is unique, eliminating redundancy within the dataset.
- Convert Date Columns: Transformed date columns to datetime format to facilitate filtering, sorting, and time-based analysis, such as identifying seasonal trends or calculating event durations.
- Map Binary Values: Mapped binary columns, like is_free and is_recurring, to 0 and 1 to improve compatibility with machine learning models.
- Convert Price to Numeric: Converted the price column to a numeric type. 16 events had non-numeric entries in the price column, such as "Varies" or "Some ticketed events," indicating variable pricing structures that could not be quantified. To maintain data integrity, these entries were removed from the dataset as they could not be reliably converted to a numeric format.

3.2 Handling Missing values

To ensure data quality and completeness, several targeted strategies were applied to address missing values across key columns, as illustrated in Figure 3 and Figure 4:

- Time Columns: start_time and end_time had high missing rates and were considered non-essential, so they were removed.
- Price: For events marked as "free," missing prices were set to 0, while non-free events with missing prices were excluded to maintain a consistent pricing structure.
- Booking URLs: Missing booking_URL values for non-free events were supplemented from contact_URL, with manual updates where necessary.

- County: Events missing both county and address were removed to support accurate geographic and pricing analysis.
- Venue Ratings: Missing values in venue_rating were set to -1, and missing venue_userRatingCount entries were set to 0 to differentiate absence of interaction from actual ratings.

For further details and the rationale behind these approaches, refer to Appendix C. Missing Values.

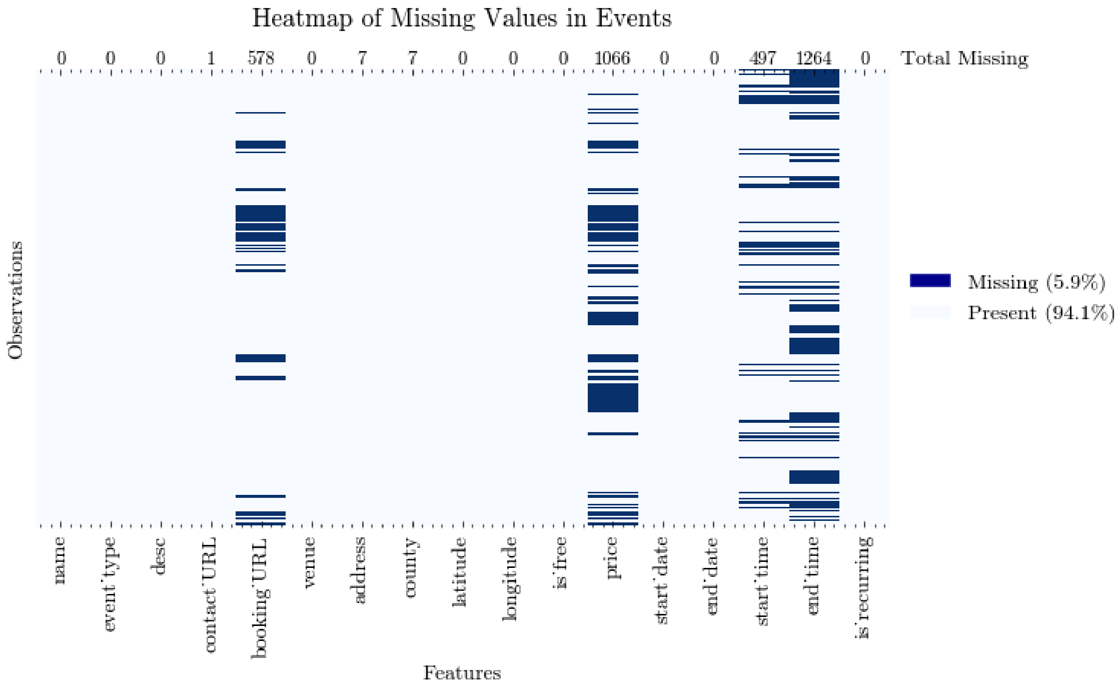


Figure 3: Missing Values in Event Dataset

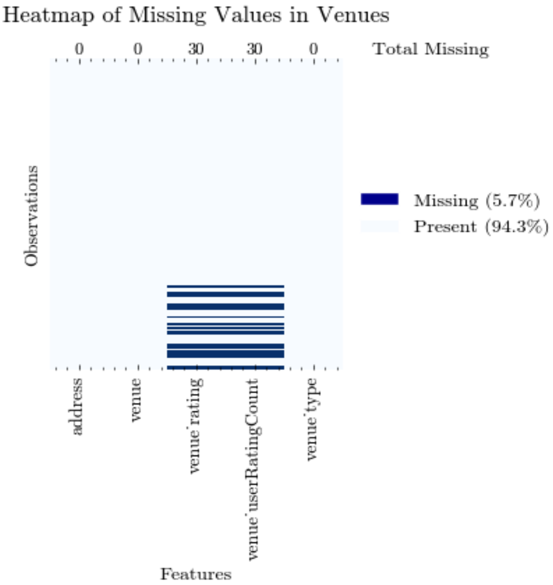


Figure 4:Missing Values in Venues Dataset

3.3 Feature Extraction

New features were extracted from the existing data to capture patterns and enhance the dataset, aiming to improve model performance:

- Duration: Calculated the duration of each event by counting repeated entries of the same event name.
- Month: Extracted the month of occurrence from the start_date column to support time-based analysis.
- Is Weekend: Added a binary indicator to distinguish events occurring on weekends. Season: Classified events by season (spring, summer, autumn, winter) to explore seasonal trends.
- Booking Platform: Extracted the primary booking platform from the booking_URL for platform-related analysis.
- Distance from Dublin: Calculated the distance from Dublin using latitude and longitude coordinates.
- Population: Merged population data by county to examine demographic influence.
- Foreign Visitors: Incorporated monthly foreign visitor data to analyze tourism's impact; simple time-series regression was used to predict future visitor counts.
- Venue Characteristics: Merged venue-specific attributes, including venue_rating, venue_userRatingCount, and venue_type, to enrich the dataset.

3.4 Descriptive Statistics

3.4.1 Measures of central tendency and dispersion

The dataset provides an overview of events across Ireland, revealing trends in pricing, recurrence, location, and seasonality. The average event price is €18, with a median of €15, indicating moderate pricing for most events. Approximately 30% of events are free, suggesting a mix of accessible and ticketed options. Most events (87%) are recurring, reflecting a focus on ongoing activities. Events are predominantly clustered near Dublin, with a median distance of 12 km, though some farther locations increase the average to 65 km. Events are often held in high-population and high-tourism areas, with a monthly average of 317,000 foreign visitors. Autumn, particularly October, sees the highest concentration of events, pointing to strong seasonal demand. For further details on each variable, please refer to Appendix D. Descriptive Statistics.

3.4.2 Kurtosis

As shown in Figure 5, variables such as Price and Foreign Visitors exhibit high kurtosis, indicating concentrated peaks with heavy tails. This suggests the presence of outliers or extreme values, especially in event prices and visitor counts, likely influenced by factors like special events or seasonal trends. Duration and Distance from Dublin display more balanced distributions, indicating a reasonable spread of values, while Venue Rating shows a strong central tendency, with most ratings clustering around the mean, reflecting general venue quality consistency. For a detailed analysis of kurtosis for each variable, please refer to Appendix D. Descriptive Statistics.

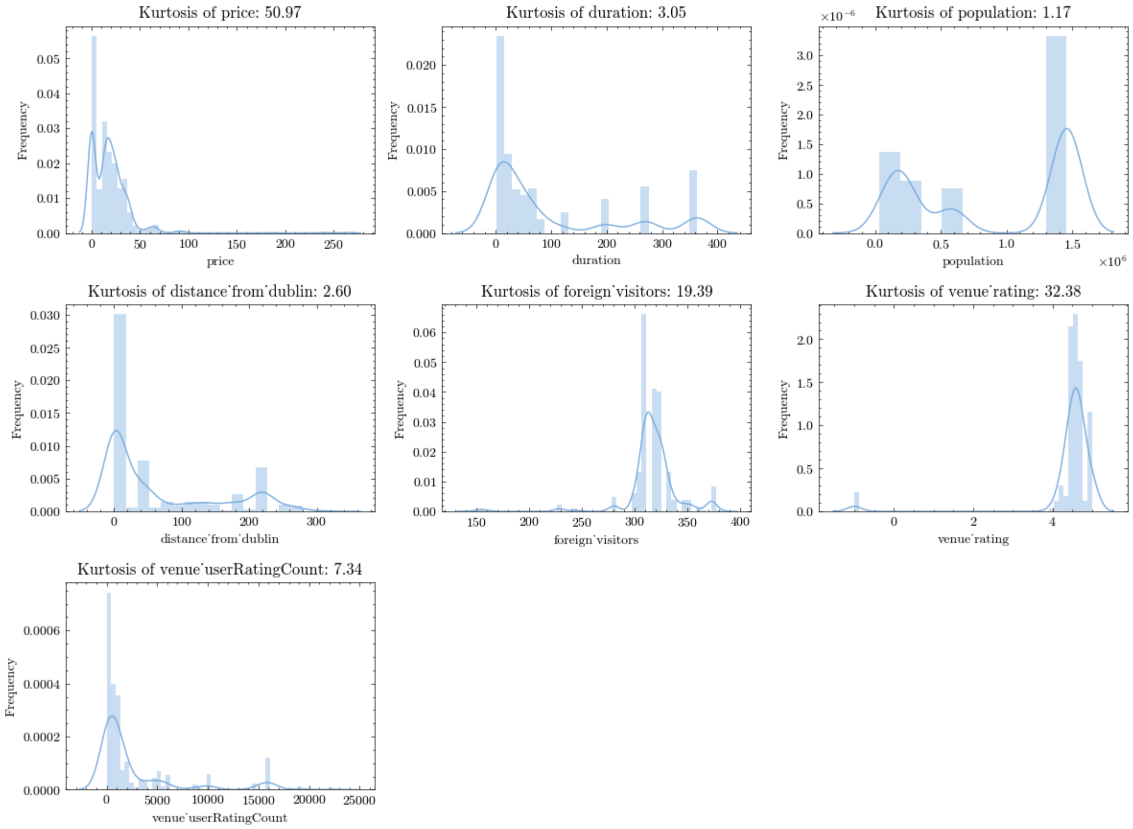


Figure 5: Distribution and Kurtosis of Key Numerical Features

3.5 Outliers Analysis

Outliers were evaluated based on their potential impact on analysis and modeling. While many outliers were retained to capture genuine variations in the data, specific outliers in the Price variable were removed due to their distinct characteristics.

Given the skewed distribution of the price data, the Interquartile Range (IQR) method was used for outlier identification and removal (Tukey, 1977). A handful of high-priced events (values such as €185, €240, €250, and €265) represent multi-day or bundled events with numerous activities or sessions. Unlike single-day events in the dataset, these entries have a pricing structure that covers multiple days or event categories, making them incomparable to standard single-event pricing. Therefore, these entries were removed to ensure a consistent pricing model across the dataset.

The retained outliers across Duration, Foreign Visitors, Venue Rating, and Venue User Rating Count reflect the natural diversity within these variables, with details provided in Appendix F. Outlier

3.6 Analysis of Free Events

As shown in the bar chart (Figure 6), free events are less common, comprising only 11.7% of the 690 unique events analyzed. The majority are paid, indicating a widespread presence of ticketed events across Ireland. The aggregated tables used for the comparison are provided in Appendix G. Events Analysis for further reference.

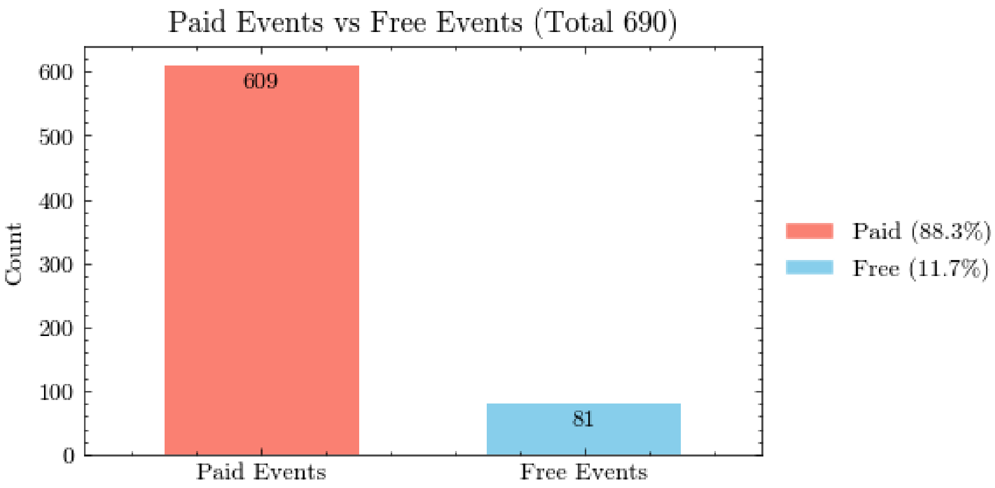


Figure 6: Distribution of Paid vs. Free Events

3.6.1 Free Events vs County and Population

To understand how free events are distributed relative to population and total event counts, the analysis combines data on population size, percentage of free events, and total events per county.

The bubble chart (Figure 7) illustrates that smaller populations tend to host a higher percentage of free events, as evidenced by bubbles representing lower population levels. The bubble size, which indicates the total number of events, provides context for understanding the proportion of free events relative to the overall event count in each population group.

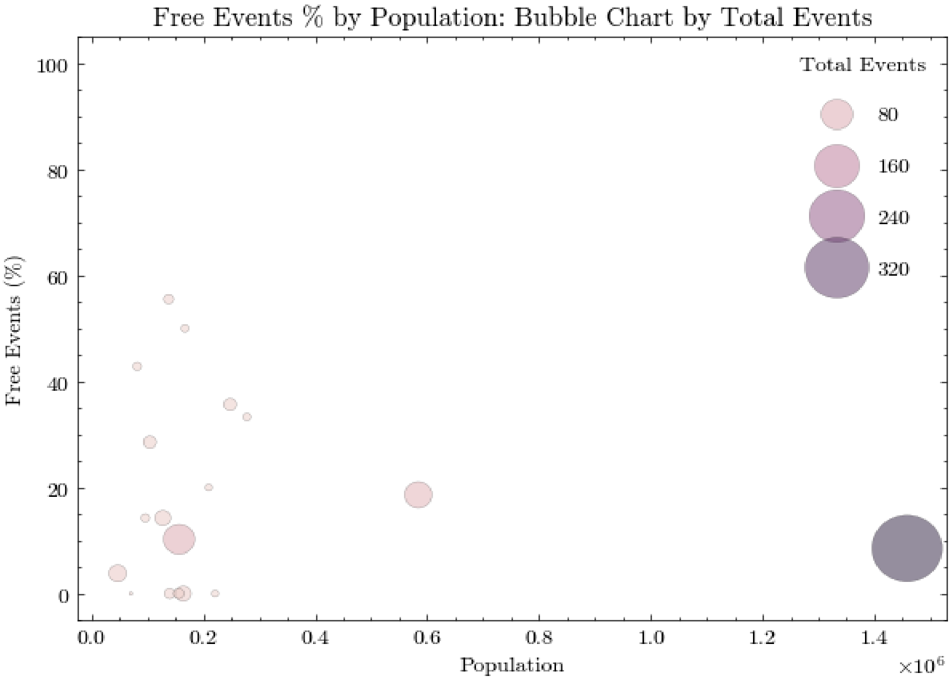


Figure 7: Free Events Percentage by Population with Total Events

The treemap (Figure 8) paints a clear picture of regional differences in event accessibility. Smaller counties like Monaghan, where every event is free (100%), and Mayo, with 56%

free events, stand out for their focus on offering no-cost options. On the other hand, larger, more populous counties like Dublin and Cork, with only 9% and 19% of their events being free, seem to lean more towards hosting paid events.

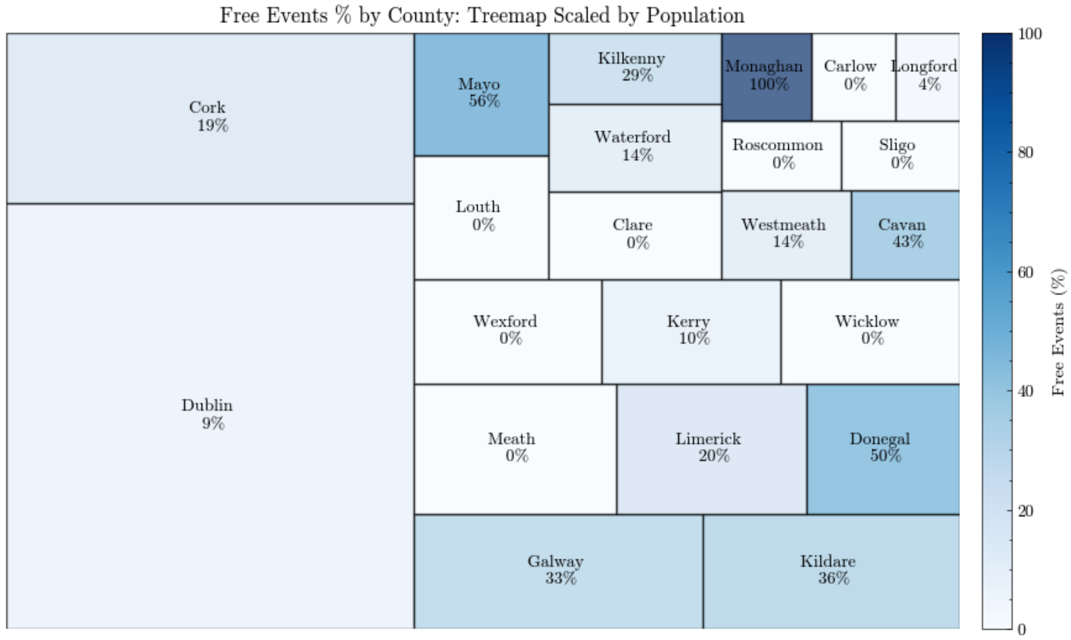


Figure 8: Free Events Percentage by County: Treemap Scaled by Population

Together, these visualizations suggest that less populated areas may prioritize free events to encourage participation, while larger counties, possibly driven by demand and revenue opportunities, tend to host more paid events.

3.6.2 Free Events vs Season

To understand how free events are distributed across seasons, the analysis uses both total counts and percentages. The stacked bar chart (Figure 9) displays total events by season, which might initially suggest that autumn has a high proportion of free events due to the larger blue segment.

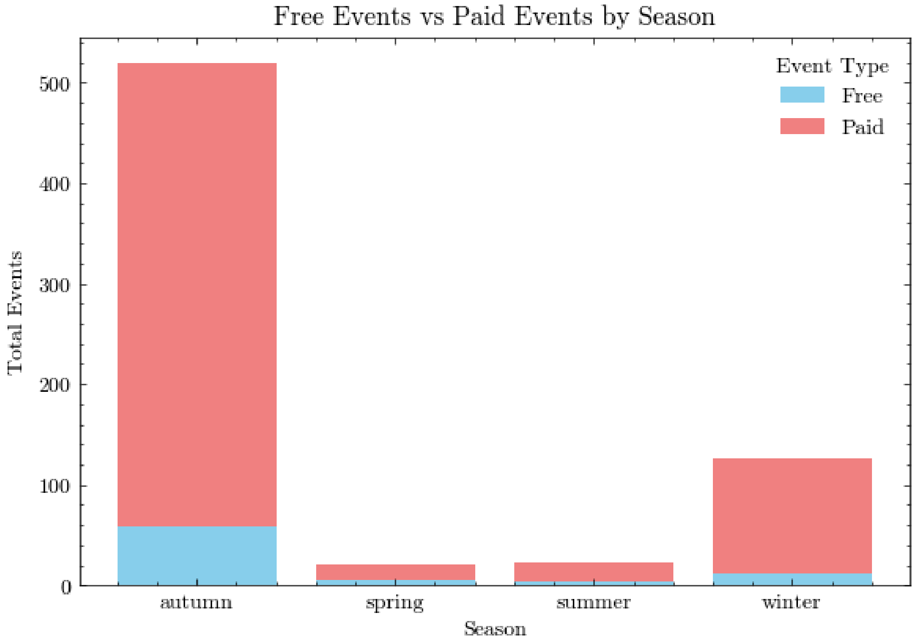


Figure 9: Free Events vs. Paid Events by Season

However, when looking at the actual percentage of free events by season in Figure 10, it's evident that spring has the highest proportion of free events at 28.57%, followed by summer at 17.39%. Although autumn has the most events overall, only 11.37% are free, and winter has the lowest proportion at 9.45%. This demonstrates that while autumn and winter host many events, they are predominantly paid, whereas spring has a relatively higher focus on free events.

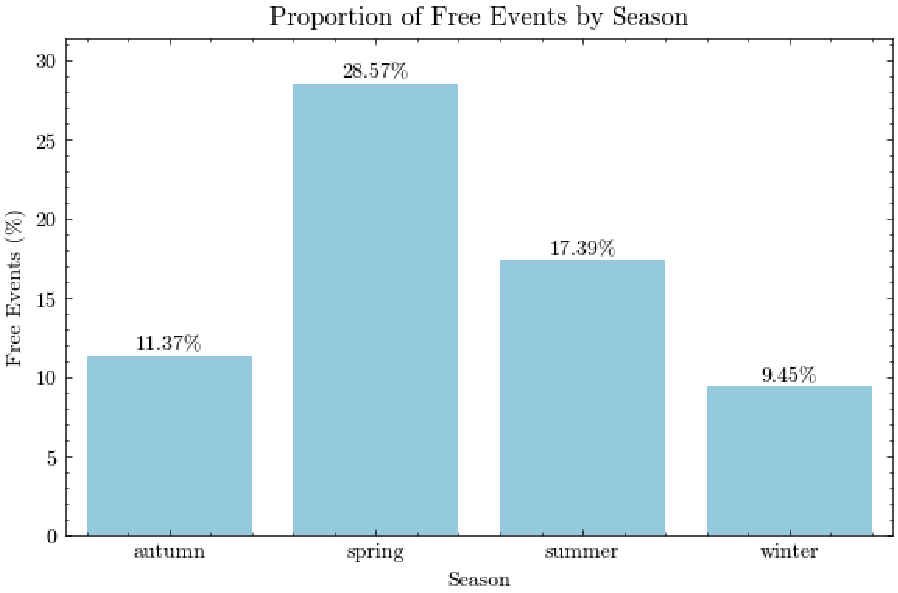


Figure 10: Proportion of Free Events by Season

3.7 Correlation Analysis of Paid Events

The relationship between price and various features was analyzed, distinguishing discrete and continuous numerical variables. Discrete variables were treated as categorical to avoid misleading interpretations (Odvese, Akpusugh & Tertsea, 2024), while continuous variables were assessed for correlations. Categorical variables were analyzed based on their distributions.

3.7.1 Discrete Numerical Variables

Analysis of unique value counts identified is_recurring and is_weekend (both with two unique values) as discrete features, while venue_rating was treated as discrete due to its limited distinct values, despite being ordinal. Visualizing these features alongside median prices (Figure 11) revealed:

- Venue Rating: Higher ratings are linked to higher median prices, indicating a positive relationship between venue quality and pricing.
- Is Weekend: Minimal differences in median prices suggest timing has little impact on pricing.
- Is Recurring: Recurring events are generally priced lower than non-recurring ones, reflecting potential variations based on frequency.

The median was used to represent typical prices, as it is less affected by skewed pricing data with high-cost outliers (Ahmed, 2024). The error bars, calculated by Seaborn using bootstrapped confidence intervals, indicate the range within which the median would likely fall with 95% confidence, providing insight into the reliability of these estimates.

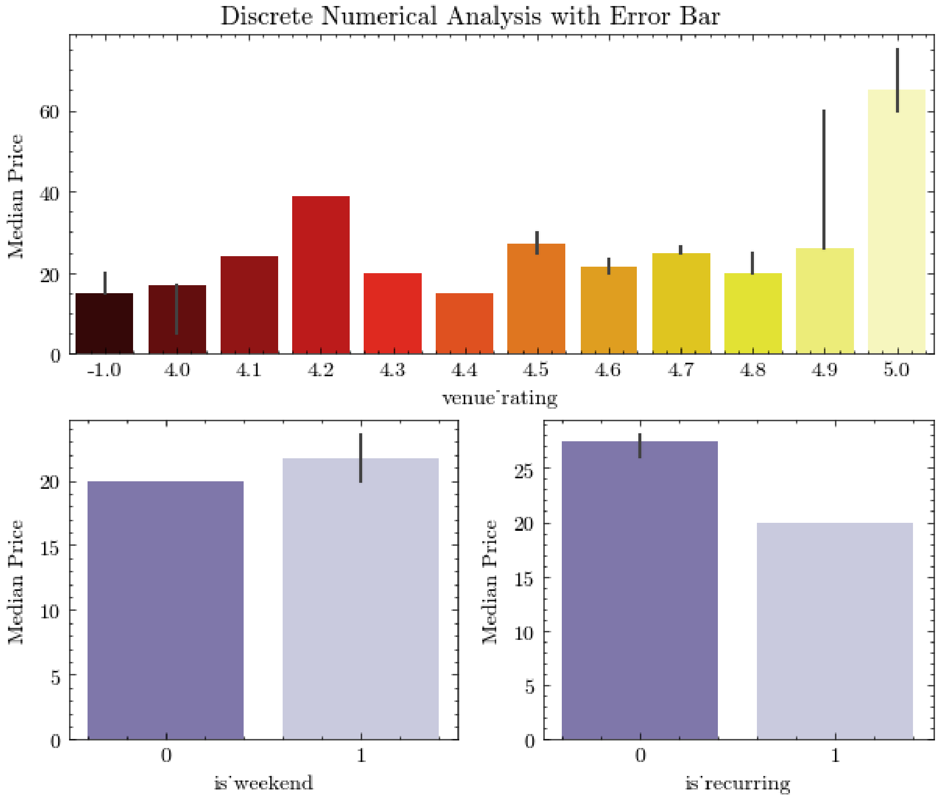


Figure 11: Discrete Numerical Analysis with Error Bars

3.7.2 Continuous Numerical Variables

Due to the presence of outliers and skewed distributions, Spearman correlation was chosen as it effectively captures non-linear relationships. Spearman Correlation Result from heatmap in Figure 12:

- Duration (-0.29): Weak negative correlation, suggesting that longer events are generally priced lower.
- Distance from Dublin (-0.10): Very weak negative correlation, indicating minimal effect of distance on pricing.
- Population (0.25): Weak positive correlation, indicating a tendency for higher prices in areas with larger populations.
- Foreign Visitors (0.27): Weak positive correlation, suggesting a slight association between tourist influx and higher prices.
- Venue User Rating Count (0.22): Weak positive correlation, with higher ratings possibly contributing to higher event prices.

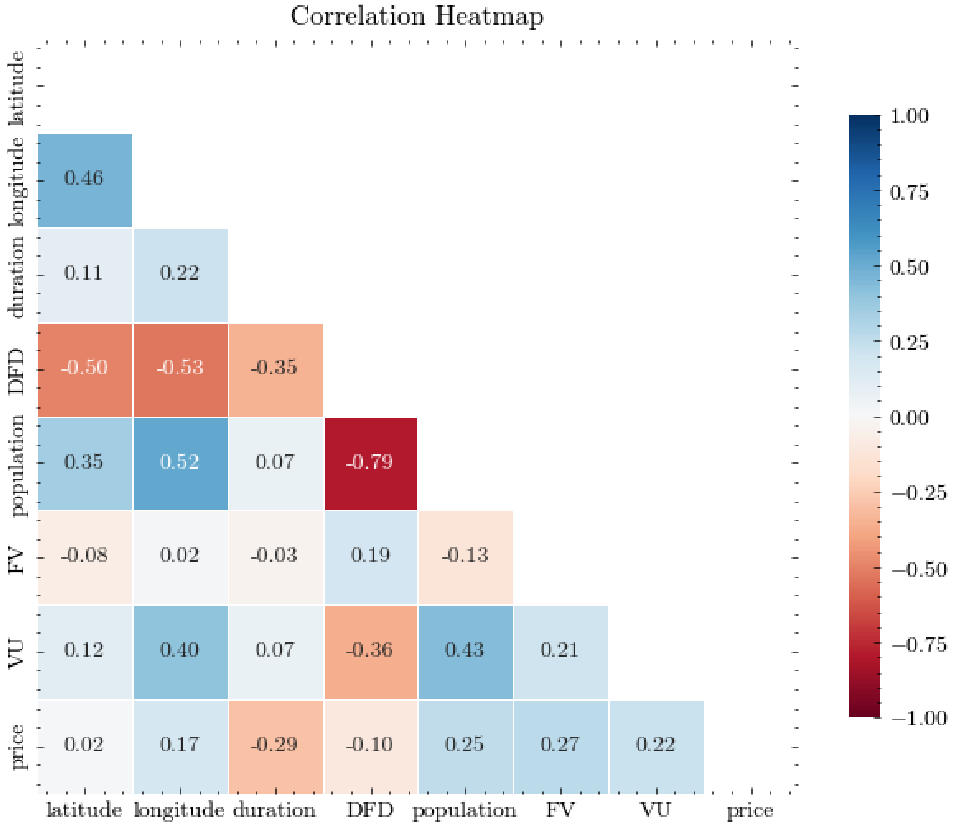


Figure 12: Correlation Heatmap of Numerical Features

The scatter plots (Figure 13), consistent with the heatmap findings, reveal no strong linear relationships between the selected features and price, suggesting that event pricing is influenced by a combination of other factors.

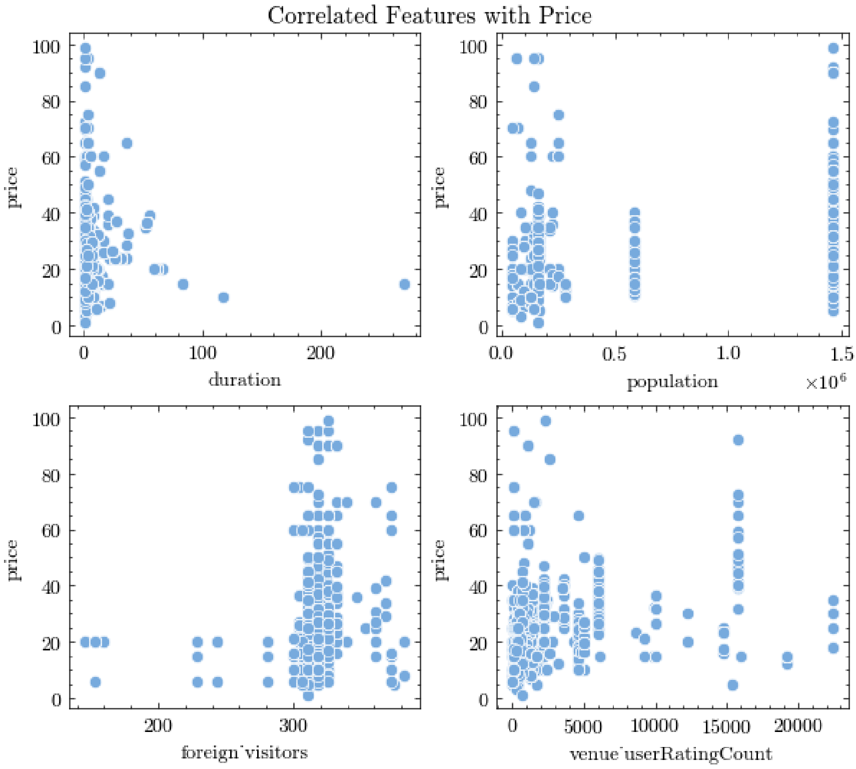


Figure 13: Scatterplots of Features Correlated with Price

3.7.3 *Categorical variables*

The objective is to compare median prices across categories to understand how categorical variables influence event pricing. Various visualizations were used to analyze categorical variables and pricing: box plots highlighted medians and variability, summary tables provided sample sizes (in Appendix G. Events Analysis), value distributions enabled simple comparisons for fewer categories, and swarm plots clarified distributions for many unique values.

- Event Type: Minimal association with price, as seen from overlapping distributions of median prices (Figure 14).

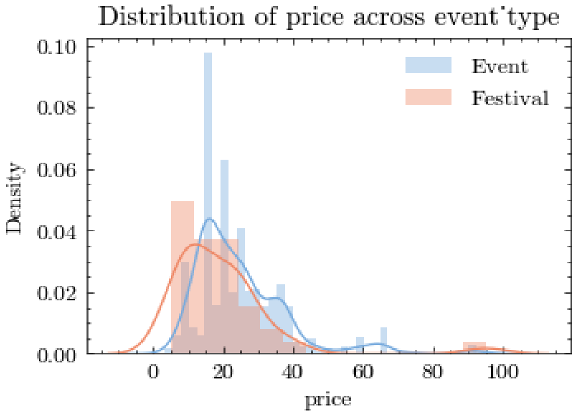


Figure 14: Distribution of Price Across Event Types

- County: Significant variation in median prices across counties, with Kildare county showing notably higher prices (Figure 15).

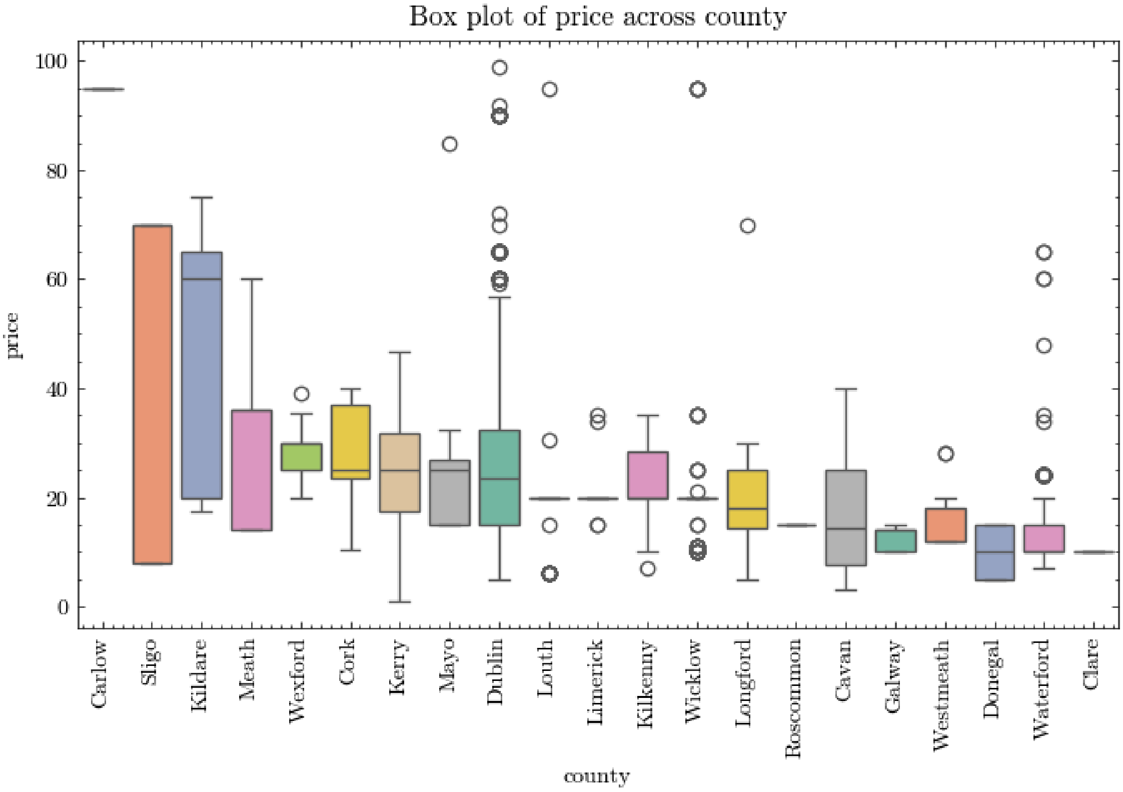


Figure 15: Box Plot of Price Across Counties

- Season: Winter events tend to have a higher median price and greater variability, while autumn, despite a high number of events, shows a lower median price (Figure 16).

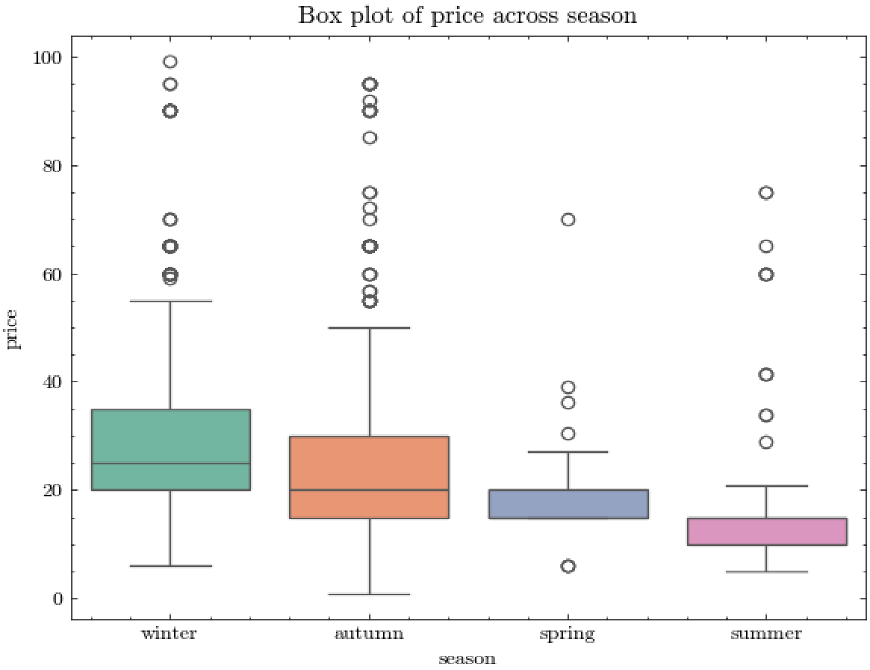


Figure 16: Box Plot of Price Across Seasons

- Booking Platform: Some platforms cluster around certain price points, suggesting platform-related pricing trends (Figure 17).

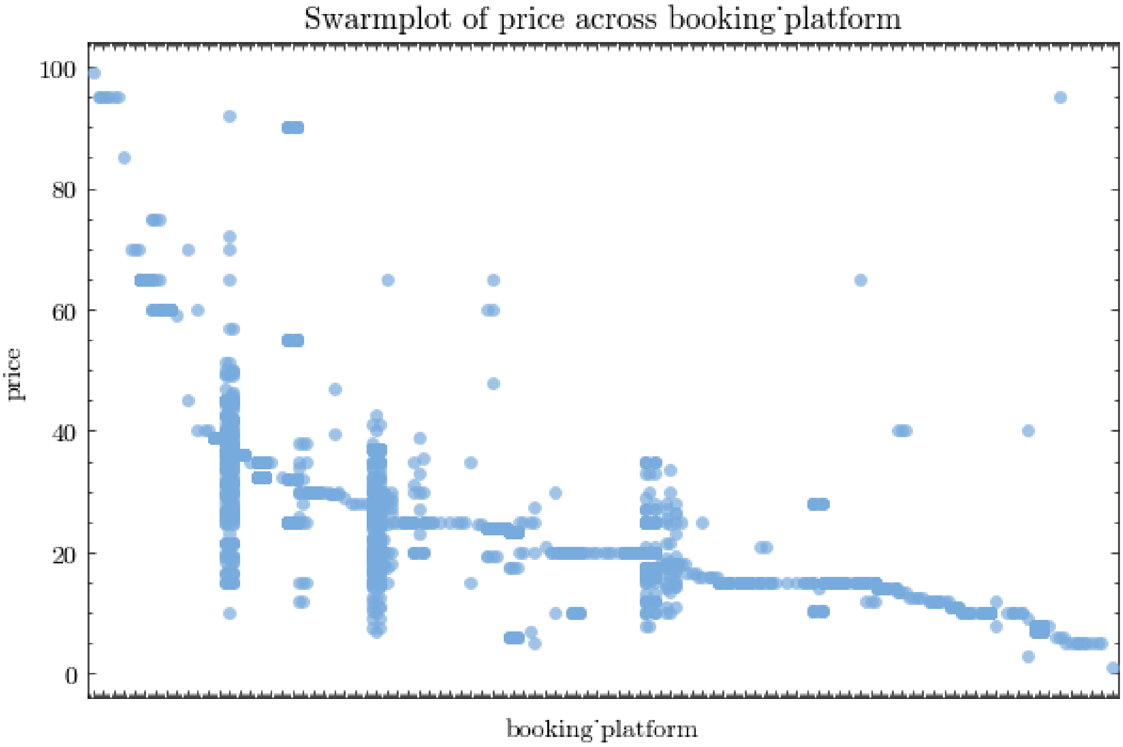


Figure 17: Swarmplot of Price Across Booking Platforms

- Venue Type: Venue type has a notable impact, with certain venues commanding higher median prices, reflecting differences in event costs based on venue nature (Figure 18).

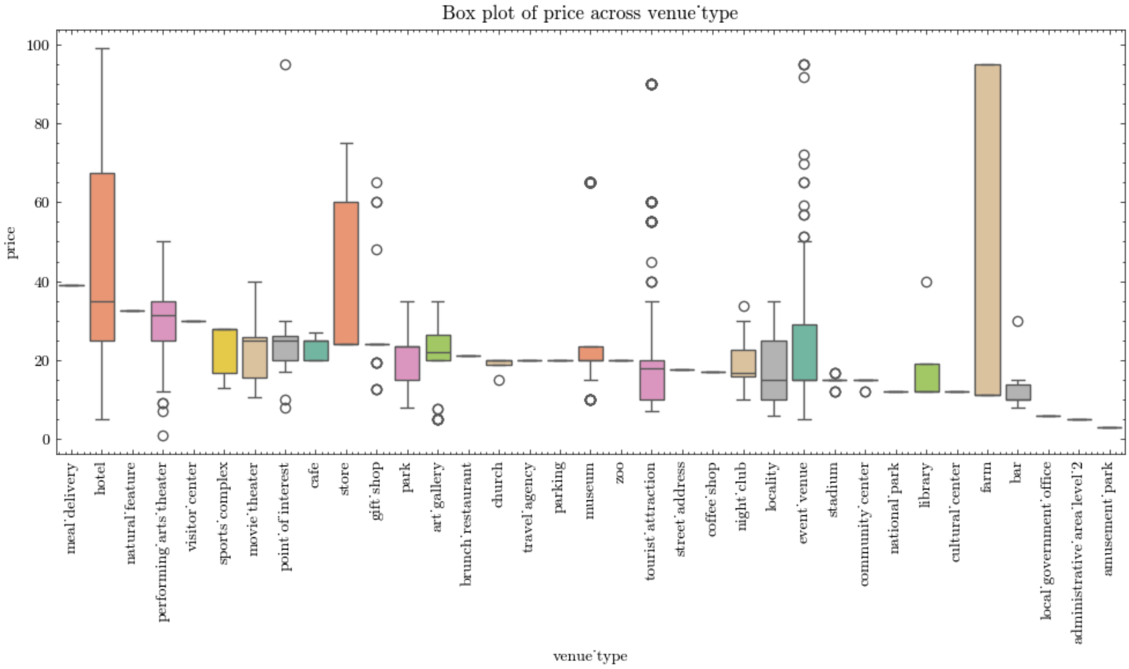


Figure 18: Box Plot of Price Across Venue Types

3.7.4 Datetime Variables

The analysis of the month variable aims to explore seasonal or time-dependent pricing patterns over the 18-month period from January 2024 to July 2025. While this limited timeframe restricts the identification of long-term trends, preliminary findings suggest that seasonal demand may have a modest influence on pricing (Figure 19).

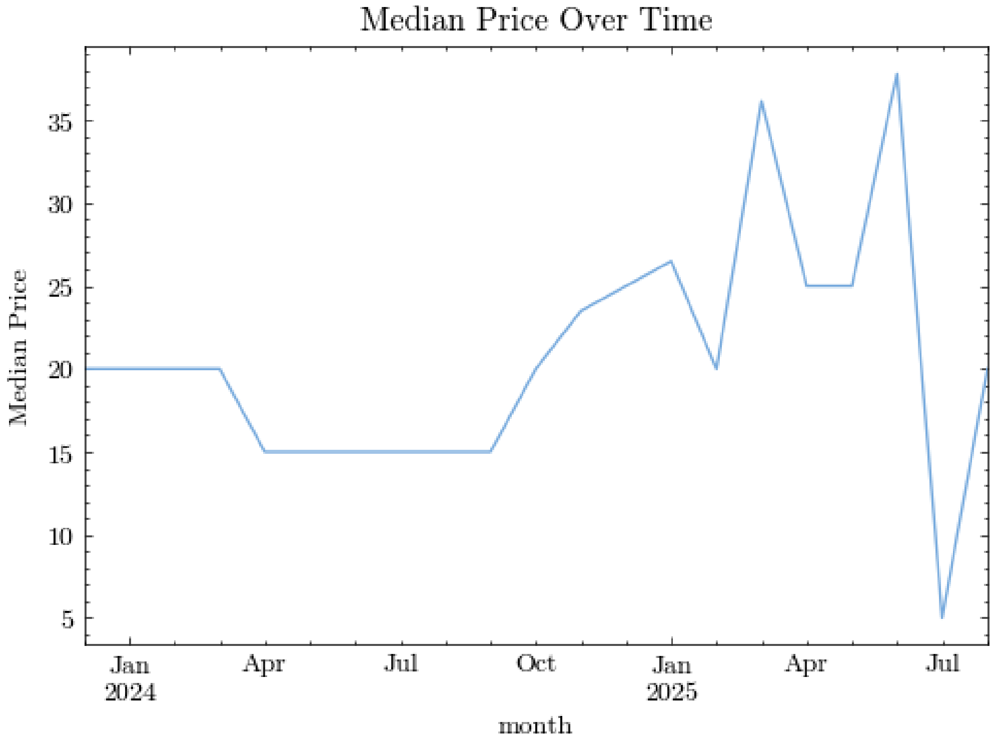


Figure 19: Median Price Over Time

4 Modeling

4.1 Feature Engineering

4.1.1 Feature Selection

Non-informative columns like Name, Venue, Address, and raw geographic coordinates were removed, along with Is_Free (irrelevant for priced events) and Start_Date/End_Date, as their key temporal details were already derived. Extracted features, such as event duration and distance from Dublin, were added as described in Section 3.3 Feature Extraction, ensuring the model leverages derived insights for more accurate predictions.

4.1.2 Feature Encoding

Feature encoding converts categorical variables into numeric formats for machine learning models. While decision trees handle categorical data inherently, linear regression requires numerical representations. Encoding strategies were applied based on feature characteristics and model requirements (Breskuvienė & Dzemyda, 2023):

- Label Encoding: Applied to ordinal features month and season, preserving temporal order and suitable for decision trees.

- One-Hot Encoding: Used for binary and low-cardinality features event_type and county to avoid implying ordinal relationships. This increased the number of features in the dataset from 15 to 68.
- Rare Encoding: High-cardinality features such as booking_platform and venue_type were consolidated into a “rare” category to reduce dimensionality and risk of overfitting.

Encoding choices were based on analyzing unique category counts to balance model efficiency and information retention. More details in Appendix H. Feature encoding.

4.1.3 Skeweness and Log transformation

By applying log and Box-Cox transformations (Box & Cox, 1964), the skewness of numeric features was significantly reduced (Table 1), stabilizing variance and enhancing data suitability for machine learning models that assume normality (James et al., 2021). However, as linear regression will only be used as a baseline in Section 4.3.1

Ordinary least squares (OLS), the primary focus will shift to non-linear models that do not rely on this assumption, meaning the transformed data will not be utilized for these models. Further analysis about skewness and application of log and Box-Cox transformations are provided in the Appendix I. Log Transformation.

Table 1: Skewness Reduction Using Log and Box-Cox Transformations

Feature	Skewness Before	Skewness After Log	Skewness After Box-Cox	Lambda (Box-Cox)
price	1.798	-0.438	0.031	0.181
duration	4.758	0.893	0.264	-0.358
distance_from_dublin	1.119	-0.053	-0.012	0.013
population	-0.439	-0.718	-0.488	0.654
foreign_visitors	-3.745	-5.835	0.843	3.76
venue_userRatingCount	2.249	-3.88	-0.222	0.217

4.1.4 Feature Scaling

Feature scaling ensures numeric features are comparable, enhancing some machine learning models performance. Robust scaling, was applied due to the skewness and outliers in the dataset. This method scales features using the median and interquartile range (IQR), making it particularly effective for data with significant outliers (Kim et al., 2023). The Table 2 summarizes statistics before and after scaling. While feature scaling is critical for linear regression to ensure consistent coefficients and stable convergence, decision tree-based models are not affected by feature scales as they split data based on thresholds (James et al., 2021). Therefore, was exclusively applied to the linear regression model. More details in Appendix J. Feature Scaling.

Table 2: Comparison of Descriptive Statistics Before and After Scaling

	Mean Before Scaling	Median Before Scaling	Std Dev Before Scaling	Mean After Scaling	Median After Scaling	Std Dev After Scaling
price	26.659	25	15.058	0.101	0	0.913
duration	16.784	3	43.011	1.378	0	4.301
distance_from_dublin	67.952	9.848	92.701	0.519	0	0.828
population	945894	1.45815e+06	624224	-0.394	0	0.48
foreign_visitors	315.771	318.048	21.562	-0.32	0	3.031
venue_userRatingCount	3072.86	1258	4357.42	0.429	0	1.03

4.2 Train-Test Split

The dataset was split into training (80%) and testing (20%) sets, following the standard 80-20 ratio to balance training and evaluation. This split provides sufficient data for the model to learn while reserving enough unseen data for reliable performance assessment. Scaling and Box-Cox transformations were applied separately to prevent data leakage, ensuring unbiased evaluation and result integrity (Géron, 2017).

- X_train: 648 samples, 67 features
- X_test: 163 samples, 67 features
- y_train: 648 samples
- y_test: 163 samples

4.3 Model Evaluation

This study aimed to predict event prices using multiple regression approaches to account for the complexity identified in exploratory data analysis (EDA), which revealed no clear linear relationships between features and the target variable. A summary of the findings will be provided in this section, with the full analysis included in Appendix K. Model Evaluation for reference.

4.3.1 Ordinary least squares (OLS)

OLS regression fits a linear equation to the data by minimizing the Mean Squared Error (MSE). While OLS is straightforward and interpretable, it relies on assumptions like linearity and homoscedasticity (Montgomery, Peck, & Vining, 2012), which may not hold in this dataset.

The OLS regression model shows moderate performance, with an R² score of ~0.30 on the test set (Table 3) and high variability across cross-validation folds (

Table 4), suggesting instability.

Table 3: OLS Training and Test Set Performance

Metric	Training Set	Test Set
Mean Absolute Error (MAE)	7.127	6.849
Mean Squared Error (MSE)	114.366	107.446
Root Mean Squared Error (RMSE)	10.694	10.366
R^2 Score	0.532	0.296

Table 4: OLS Cross-Validation Metrics

Metric	Value
Score Metric	R^2
CV Scores Mean	0.309 (+/- 0.192)
CV Scores Std	0.096
CV Folds Scores	['0.404', '0.331', '0.126', '0.360', '0.326']

Regularization methods such as Ridge, Lasso, and ElasticNet were explored to improve generalization and address multicollinearity (Montgomery, Peck, & Vining, 2012).

4.3.2 Elastic Net

ElasticNet combines L1 (Lasso) and L2 (Ridge) penalties, enabling both feature selection and coefficient shrinkage (Zou & Hastie, 2005). The best ElasticNet model, identified through grid search, has an alpha of 0.03 and an L1 ratio of 1.0, indicating reliance on Lasso regularization.

ElasticNet reduced the number of features with non-zero coefficients from 68 in the Ordinary Least Squares (OLS) model to 22, emphasizing only the most significant predictors (Figure 20). This reduction not only improves model interpretability but also contributes to enhanced stability by minimizing the influence of irrelevant features. The refined feature set was further leveraged in subsequent modeling to enhance model stability.

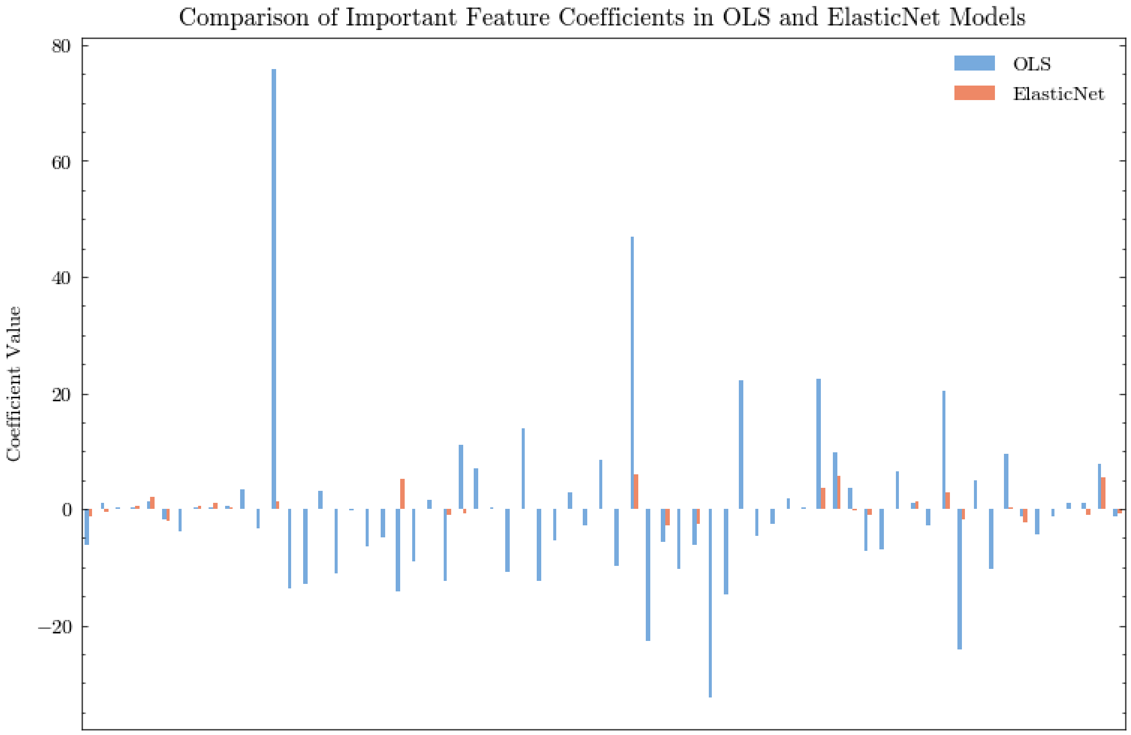


Figure 20: Comparison of Important Feature Coefficients in OLS and ElasticNet Models

4.3.3 Decision Tree Regressor

Decision Trees were chosen to address the limitations of linear models like OLS regression. The initial Decision Tree model (depth 20) overfitted (Figure 21), achieving R² scores of 0.93 on training data but only 0.31 on the test set.

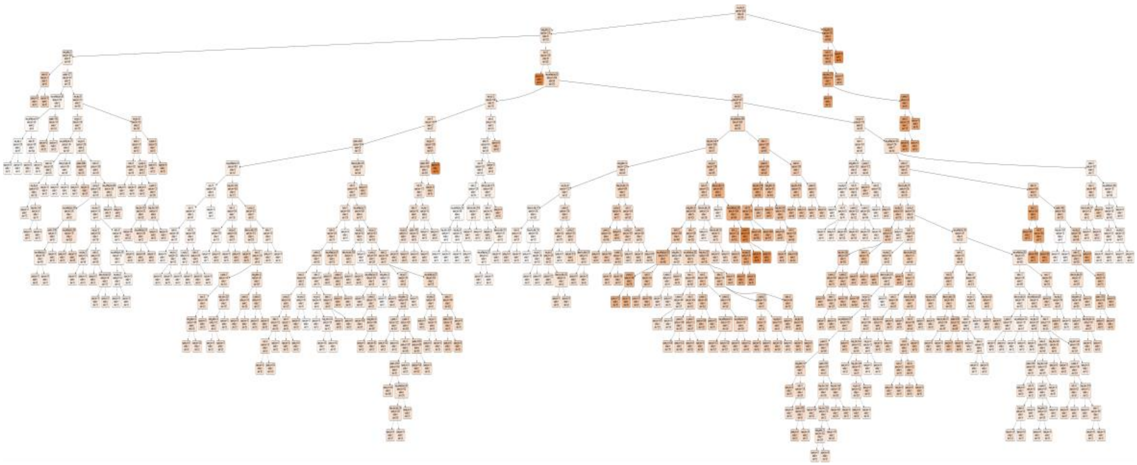


Figure 21: Initial Decision Tree model (Depth 20)

Depth tuning (Figure 22) revealed a global maximum at depth 23 but identified depth 17 as a simpler, less overfit option. Despite improvements, the final model achieved a modest test R² of 0.36 (Table 5).

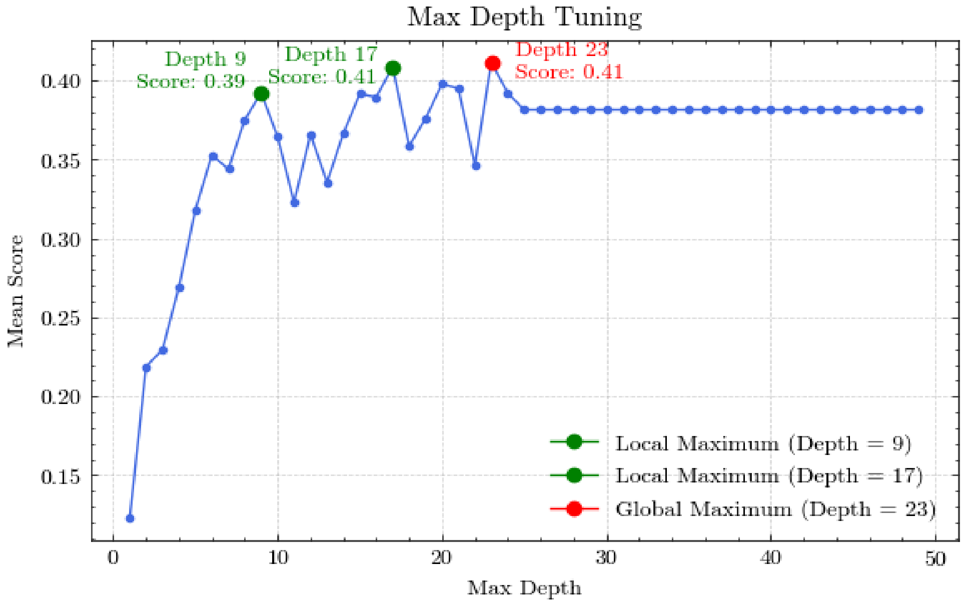


Figure 22: Max Depth Tuning for Decision Tree Model

Table 5: Training and Test Set Performance Metrics for the Final Decision Tree Model

Metric	Training Set	Test Set
Mean Absolute Error (MAE)	4.568	6.487
Mean Squared Error (MSE)	63.989	97.684
Root Mean Squared Error (RMSE)	7.999	9.884
R² Score	0.738	0.360

4.3.4 Random Forest Regressor

Ensemble methods, particularly Random Forest, were explored to improve predictive performance and robustness by combining multiple decision trees. Random Forest reduces overfitting and enhances generalization by averaging the predictions of trees built on random subsets of data and features (Breiman, 2001).

The initial Random Forest model outperformed Decision Trees, achieving a test R² score of 0.43 (compared to 0.36 for Decision Trees) and demonstrating improved generalization. Hyperparameter tuning was performed, focusing on max_depth, n_estimators, min_samples_split, and other key parameters. The best-tuned model achieved a test R² score of 0.459, with reduced variance across cross-validation folds, indicating stability (Table 7).

Table 6: Cross-Validation Metrics for the Final Random Forest Model

Metric	Value
Score Metric	R²
CV Scores Mean	0.496 (+/- 0.155)
CV Scores Std	0.078
CV Folds Scores	[‘0.561’, ‘0.459’, ‘0.384’, ‘0.475’, ‘0.603’]

Table 7: Training and Test Set Performance Metrics for the Final Random Forest Model

Metric	Training Set	Test Set
Mean Absolute Error (MAE)	4.193	6.220
Mean Squared Error (MSE)	45.071	82.583
Root Mean Squared Error (RMSE)	6.713	9.087
R² Score	0.816	0.459

However, the model still exhibited some overfitting, as the training R² score remained significantly higher (0.816). Further tuning and exploration of advanced ensemble methods, such as XGBoost, are suggested as future steps to enhance performance and robustness on unseen data.

4.3.5 XGBoost Regressor

XGBoost (Extreme Gradient Boosting) was explored to address Random Forest’s limitations by constructing trees sequentially to correct errors, enhancing accuracy. It incorporates regularization, optimized memory usage, and missing data handling, offering robust yet computationally intensive performance. (Chen & Guestrin, 2016).

After basic hyperparameter tuning, XGBoost achieved a final test R² score of 0.529, with improved error metrics (Table 8) and good stability with a standard deviation of 0.058 across fold (Table 9).

Table 8: Training and Test Set Performance Metrics the Final XGboost Model

Metric	Training Set	Test Set
Mean Absolute Error (MAE)	3.571	5.477
Mean Squared Error (MSE)	28.692	71.976
Root Mean Squared Error (RMSE)	5.356	8.484
R² Score	0.883	0.529

Table 9: Cross-Validation Metrics for the Final XGboost Model

Metric	Value
Score Metric	R²
CV Scores Mean	0.572 (+/- 0.116)

Metric	Value
CV Scores Std	0.058
CV Folds Scores	[‘0.580’, ‘0.543’, ‘0.477’, ‘0.617’, ‘0.642’]

Although XGBoost demonstrated better performance and generalization, its inclusion in this study was limited to showcasing its capabilities without extensive tuning or detailed evaluation. Future work could explore its full potential with thorough parameter optimization and comparative analysis.

5 Results

5.1 Models Comparison

To evaluate model performance, residual plots and actual vs. predicted plots (Figure 23 and Figure 24) were used. Residual plots reveal issues like non-linearity or outliers, while actual vs. predicted plots assess prediction accuracy by showing alignment with the diagonal line.

While OLS and Elastic Net were limited in their ability to model non-linear relationships, Random Forest and XGBoost demonstrated significant improvements. XGBoost, in particular, emerged as the most robust model, capturing data complexity effectively and providing the highest generalization accuracy. Table 10 summarizes the models’ performance on both the training and test datasets across R2, MAE, and MSE.

Table 10: Models Metrics Comparison

	Model	MAE (Train)	MSE (Train)	RMSE (Train)	R2 (Train)	MAE (Test)	MSE (Test)	RMSE (Test)	R2 (Test)
0	OLS	7.85	129.39	11.38	0.47	6.79	102.95	10.15	0.33
1	Elastic Net	9.5	194.92	13.96	0.2	8.15	132.97	11.53	0.13
2	Decision Tree	4.57	63.99	8	0.74	6.49	97.68	9.88	0.36
3	Random Forest	4.19	45.07	6.71	0.82	6.22	82.58	9.09	0.46
4	XGBoost	3.57	28.69	5.36	0.88	5.48	71.98	8.48	0.53

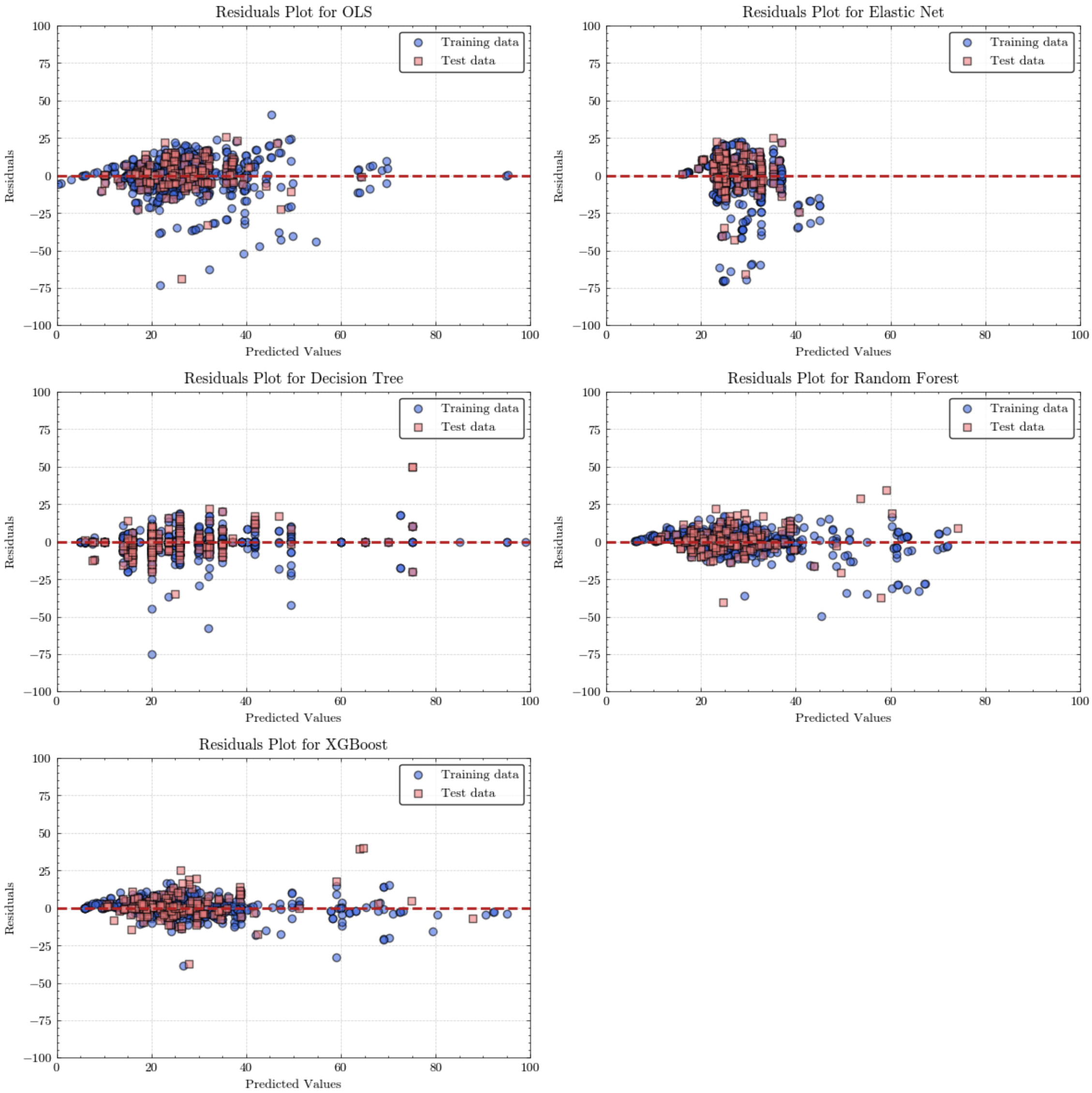


Figure 23: Residual Plots Modes Comparison

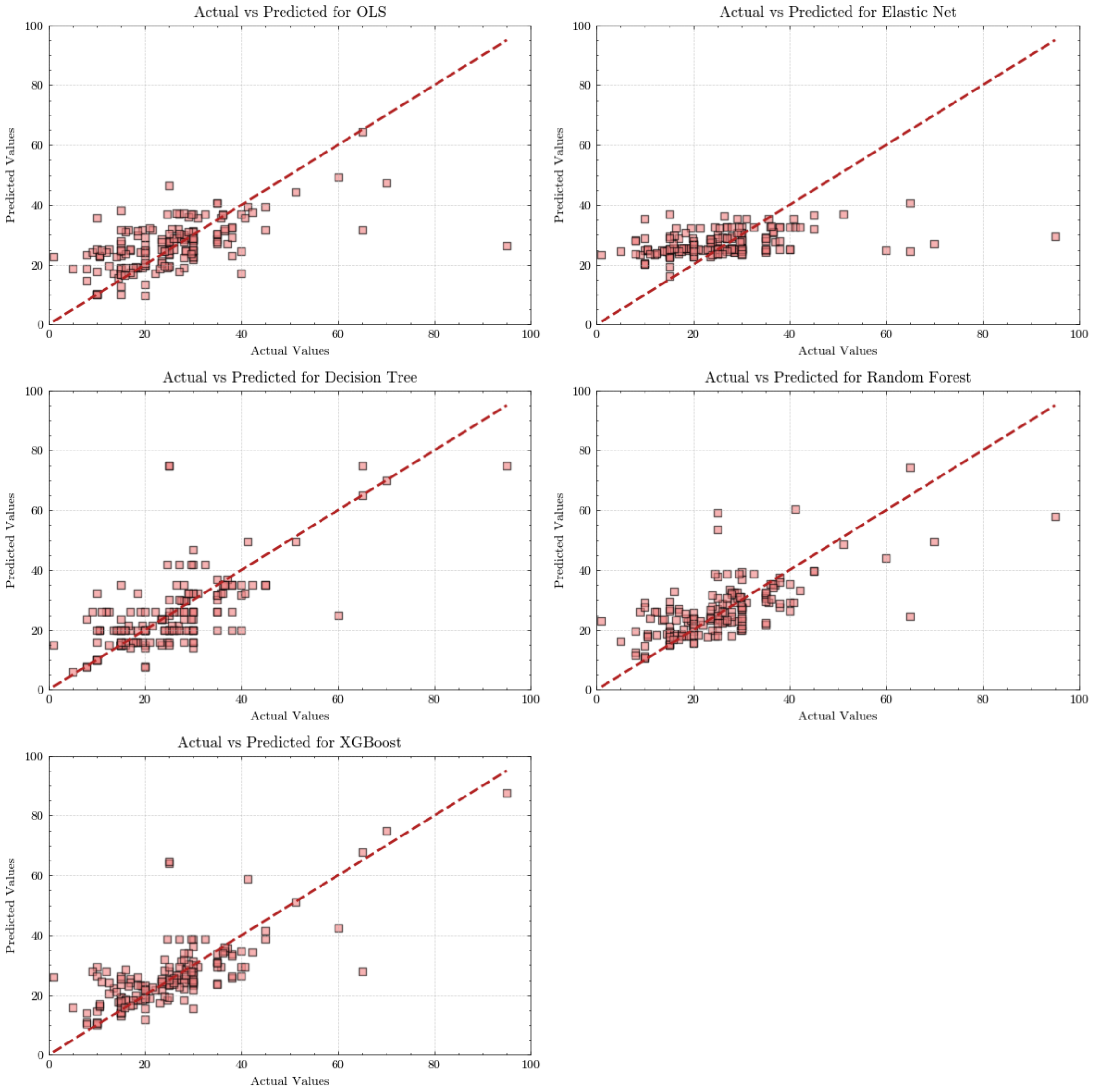


Figure 24: Actual vs Predicted Models Comparison

5.2 Feature Importance

Feature importance from XGBoost (Figure 25) and ElasticNet (Figure 26) highlights key predictors like “ticketmaster” and “bluewayartstudio” booking platforms and “store” venue type. ElasticNet emphasizes linear relationships with higher weights, while XGBoost captures non-linear interactions, spreading importance more evenly. Tools like SHAP could provide deeper insights but were not used due to time constraints.

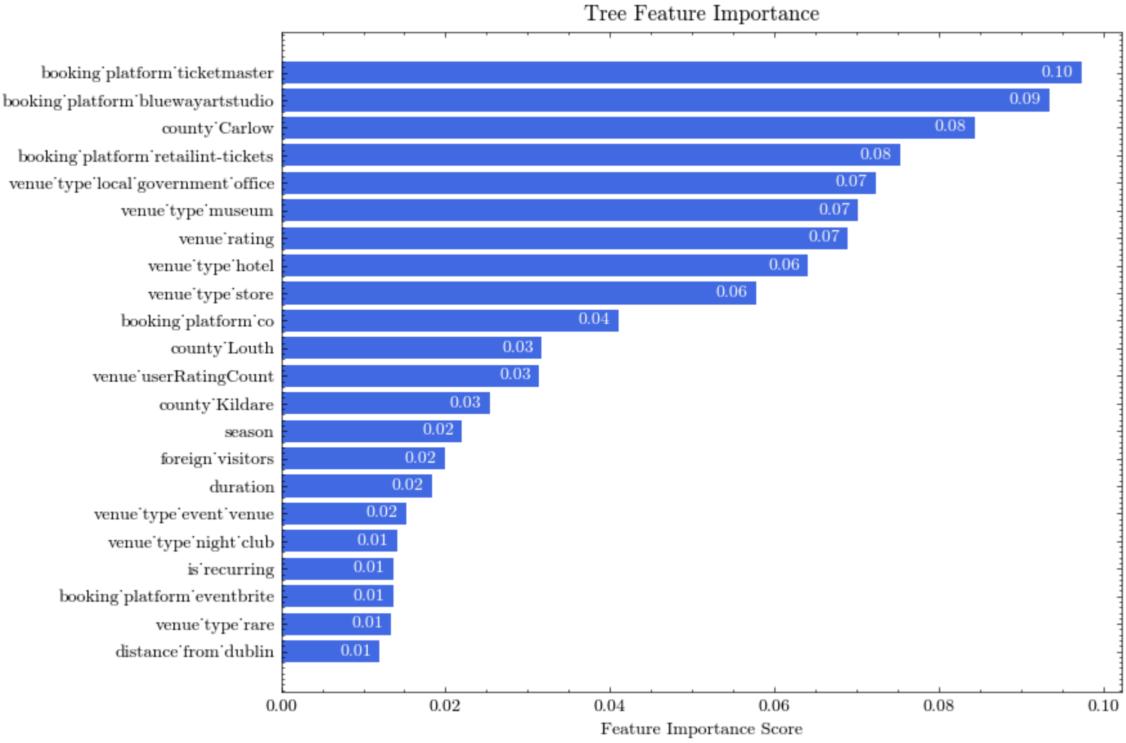


Figure 25: XGboost Feature Importance

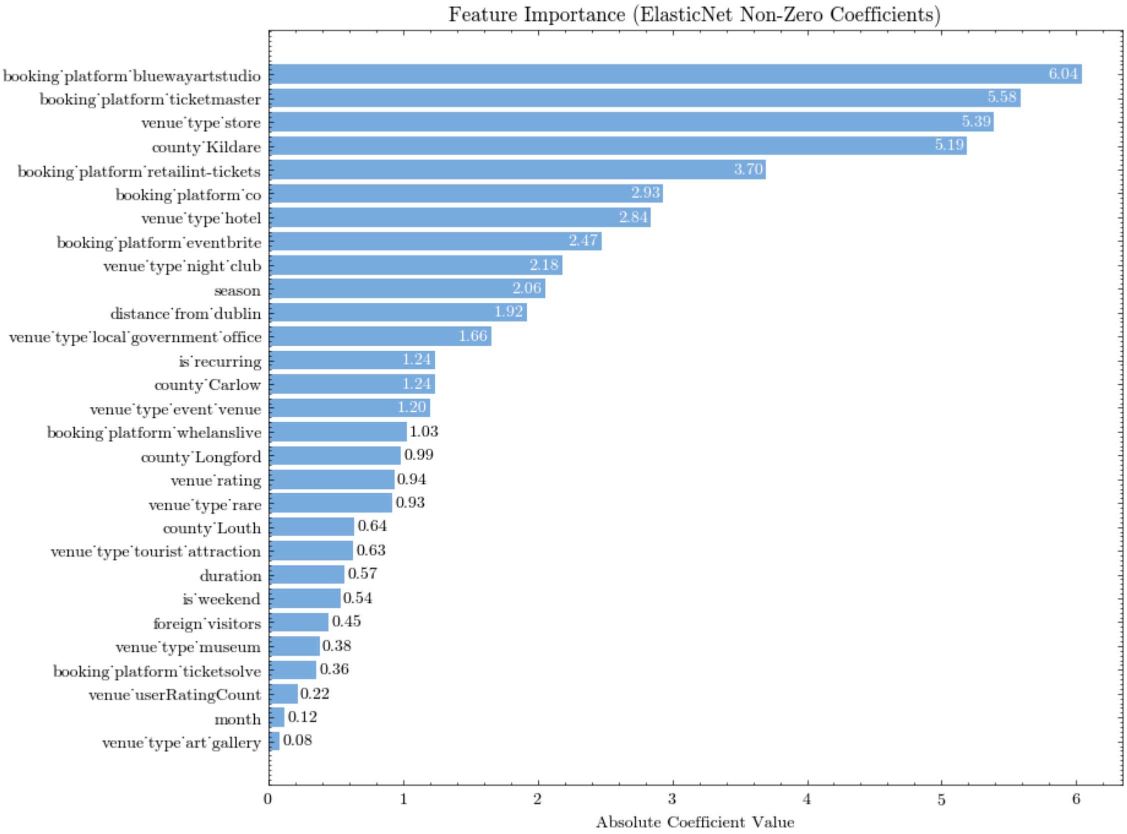


Figure 26: ElasticNet Coefficient Weight

5.3 Inferential Statistics

Inferential statistics using binomial, Poisson, and normal distributions are applied to analyze specific features in the dataset. This section summarizes key findings, with detailed analyses provided in the Appendix L. Inferential Statistic.

5.3.1 Binomial Distribution

The binomial distribution is applied to the is_free column to model the likelihood of free events across seasons (Figure 27), leveraging its binary nature (free vs. paid). This analysis quantifies and compares the probabilities of free events by season. Using the cumulative distribution function (CDF), probabilities for specific scenarios, such as observing a certain number of free events, are estimated.

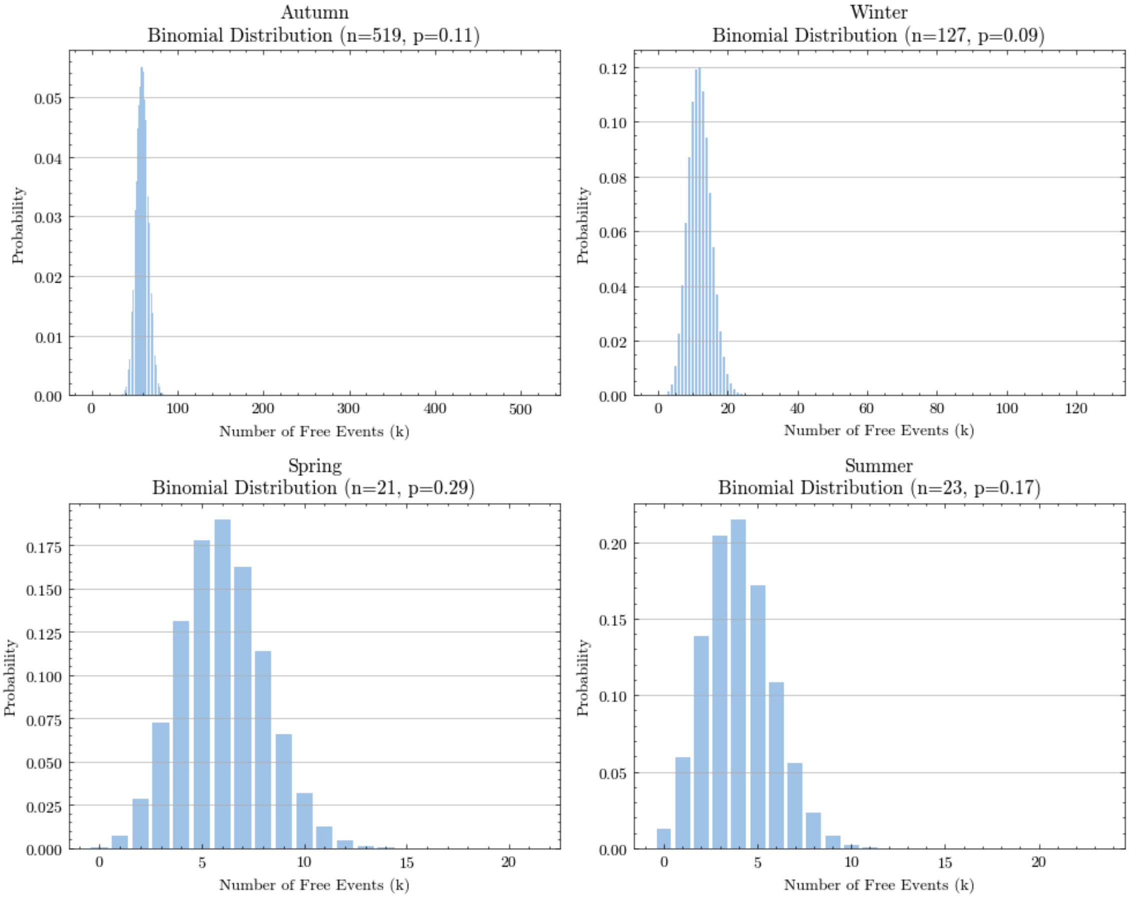


Figure 27: Binomial Distribution of Free Events Across Seasons

Approximations simplify binomial computations when trials (n) are large (Feller, 1968). As shown in Figure 28 and detailed in the Appendix L. Inferential Statistic, the normal approximation works well for moderate probabilities (p = 0.5), producing a symmetric distribution, while the Poisson approximation is effective for small probabilities (p = 0.05), capturing skewed distributions efficiently.

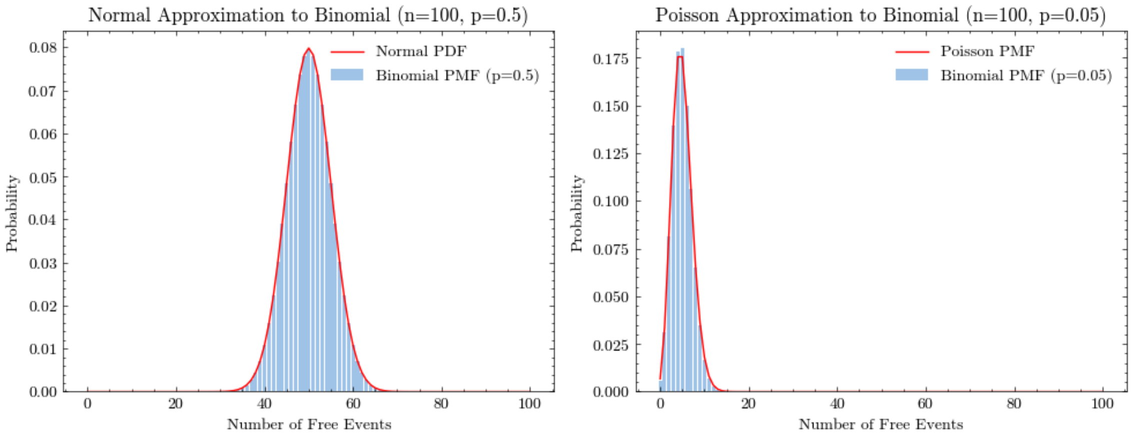


Figure 28: Approximations to the Binomial Distribution

5.3.2 Poisson Distribution

The Poisson distribution models daily counts of free and paid events, meeting criteria such as discrete values, non-overlapping events, and stable mean rates. It estimates probabilities, like observing more than three free events in a day, and aligns well with observed data despite slight deviations (Figure 29).

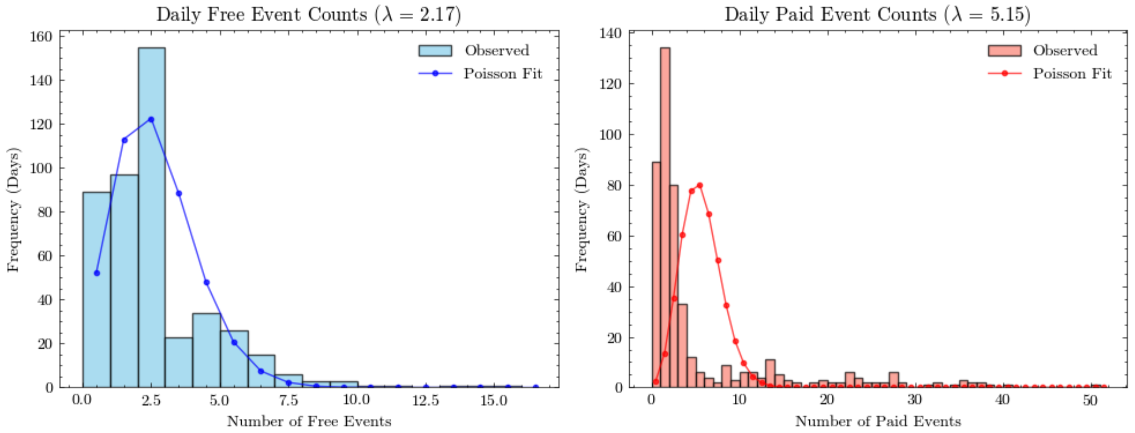


Figure 29: Poisson Fit for Daily Event Counts

As shown in Figure 30, for large λ (e.g., λ = 25), the Poisson distribution becomes symmetric and approximates a normal distribution (Feller, 1968). Using λ as the mean and √λ as the standard deviation, along with a continuity correction (+0.5), simplifies calculations while closely matching Poisson probabilities.

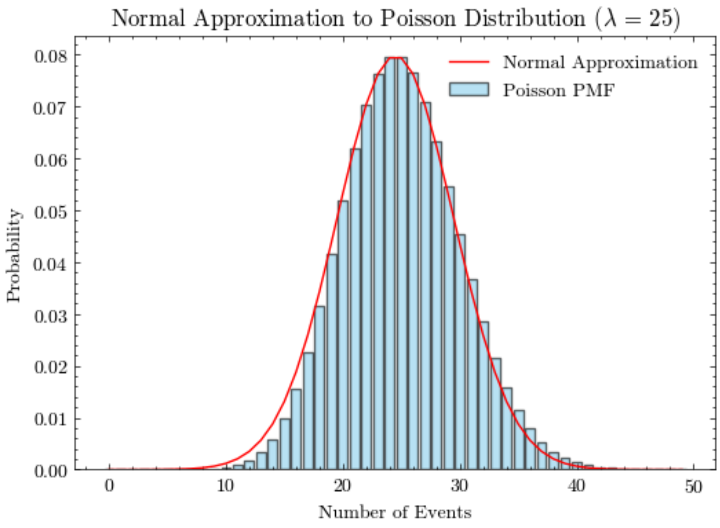


Figure 30: Normal Approximation to Poisson Distribution λ=25

5.3.3 Normal Distribution

The normal distribution is applied to model event prices, focusing exclusively on paid events to avoid skew from free events. By removing outliers and applying a Box-Cox transformation (Box & Cox, 1964), the initially right-skewed price distribution became more symmetric and closer to normal. The Shapiro-Wilk test (Shapiro & Wilk, 1965) confirmed improved alignment with normality, and z-scores revealed that approximately 62.3% of events have prices exceeding 20 units (Figure 31).

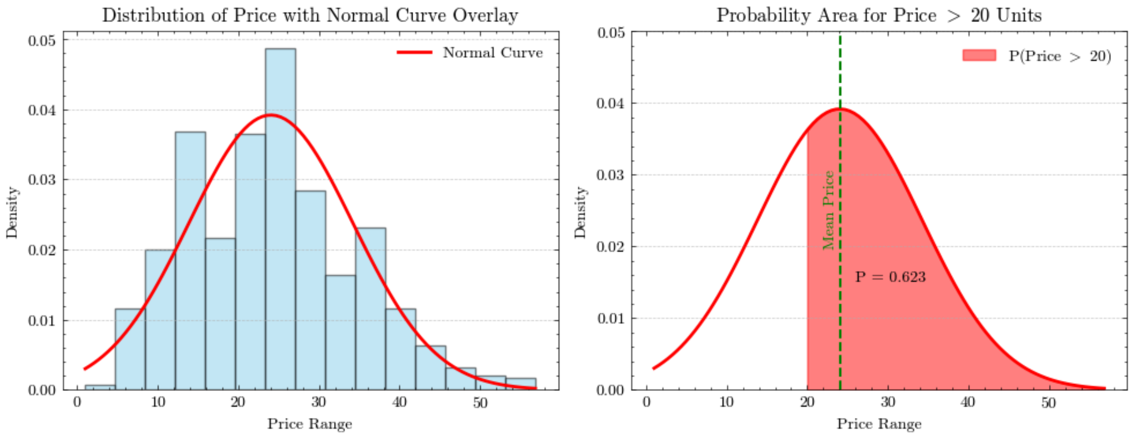


Figure 31: Price Distribution and Probability Analysis

In contrast, variables used for discrete distributions, such as event counts and binary outcomes, are not suitable for normal distribution modeling due to their discrete nature, asymmetry, and low mean values. While normal approximations (Feller, 1968) can be applied under specific conditions (e.g., large n or λ), these requirements are not consistently met in this dataset. As a result, binomial and Poisson distributions are more appropriate for accurately representing these variables.

6 Conclusions

In conclusion, this analysis highlighted the complex factors influencing tourism events in Ireland, including pricing, seasonality, and geography. Key insights were uncovered through data preparation, statistical analysis, and predictive modeling.

Feature analysis identified "booking platform" and "venue type" as the most significant contributors to model performance, reflecting distinct pricing patterns. Although the model's accuracy was limited, it explained nearly half of the variance in event prices, demonstrating its potential. Future work could enhance performance by integrating additional features, such as event-specific attributes or audience demographics.

7 References

Aggarwal, Charu. (2017). *Outlier Analysis*. 10.1007/978-3-319-47578-3.

Ahmed, Abdella. (2024). *Chapter one Introduction to Statistics.* 10.13140/RG.2.2.19043.36640.

Breskuvienė, Dalia & Dzemyda, Gintautas. (2023). *Categorical Feature Encoding Techniques for Improved Classifier Performance when Dealing with Imbalanced Data of Fraudulent Transactions*. INTERNATIONAL JOURNAL OF COMPUTERS COMMUNICATIONS & CONTROL. 18. 10.15837/ijccc.2023.3.5433.

Box, G. E. P., & Cox, D. R. (1964). *An analysis of transformations.* Journal of the Royal Statistical Society: Series B (Methodological), 26(2), 211–252.

Breiman, L. (2001). *Random forests.* Machine Learning, 45(1), 5-32. https://doi.org/10.1023/A:1010933404324

Central Statistics Office, 2022. *Population by County (F1001)*. [online] Available at: https://www.cso.ie/en/ [Accessed 8 October 2024]. Licensed under CC BY 4.0.

Central Statistics Office, n.d. *Foreign Visitors by Main Reason for Travel (ITM04).* [online] Available at: https://www.cso.ie/en/ [Accessed 10 October 2024]. Licensed under CC BY 4.0.

Chen, T., & Guestrin, C. (2016). *XGBoost: A scalable tree boosting system*. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 785–794). ACM. https://doi.org/10.1145/2939672.2939785

Fáilte Ireland, n.d. *Open Data*. [online] Available at: https://www.failteireland.ie/Research-Insights/Open-data.aspx [Accessed 5 October 2024]. Licensed under the Open Data Directive (PSI) License.

Feller, W. (1968) *An Introduction to Probability Theory and Its Applications*.

Fraunhofer Institute for Surface Engineering and Thin Films IST, n.d. *CRISP-DM - Cross Industry Standard Process for Data Mining in Surface Technology*. [online] Available at: https://www.ist.fraunhofer.de/en/expertise/simulation-digital-services/data-acquisition-model-based-process-optimization/crisp-dm-surface-technology.html [Accessed 12 November 2024].

Géron, A. (2017). *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. 1st ed. O'Reilly Media.

Google, n.d. *Google Places API.* [online] Available at: https://developers.google.com/maps/documentation/places/web-service/overview

[Accessed 10 October 2024]. Licensed under the Google Maps Platform Terms of Service.

James, G. et al. (2021) *An Introduction to Statistical Learning: with Applications in R*. Springer Nature.

Kim, Jae-Dong & Hwang, Ji-Hwan & Doh, Hyoung-Ho. (2023). *A Predictive Model with Data Scaling Methodologies for Forecasting Spare Parts Demand in Military Logistics*. Defence Science Journal. 73. 666-674. 10.14429/dsj.73.19129.

Marbán, Oscar & Mariscal, Gonzalo & Segovia, Javier. (2009). *A Data Mining & Knowledge Discovery Process Model.* 10.5772/6438.

Matthews, Nicholas. (2017). *Measurement, Levels of.* 10.1002/9781118901731.

Montgomery, D.C., Peck, E.A. and Vining, G.G. (2012) *Introduction to Linear Regression Analysis.* John Wiley & Sons.

Mukaka, Mavuto. (2012). *Statistics Corner: A guide to appropriate use of Correlation coefficient in medical research.* Malawi medical journal : the journal of Medical Association of Malawi. 24. 69-71.

Odvese, Akpusugh & Tertsea, Dekpoghol. (2024). *Comparing Discrete and Continuous Data: Concepts, Differences, and Applications*. 10.14293/PR2199.001249.v1.

OpenAI (2024) *ChatGPT (Version GPT-4)* [Large language model]. Available at: https://openai.com (Accessed: 16 November 2024).

scienceplots, n.d. *SciencePlots Documentation.* [online] Available at: https://github.com/garrettj403/scienceplots [Accessed 18 October 2024].

Shapiro, S. S., & Wilk, M. B. (1965). *An analysis of variance test for normality (complete samples).* Biometrika, 52(3-4), 591–611.

Sterne, Jonathan & White, Ian & Carlin, John & Spratt, Michael & Royston, Patrick & Kenward, Michael & Wood, Angela & Carpenter, James. (2009). *Multiple imputation for missing data in epidemiological and clinical research: Potential and pitfalls*. British Medical Journal. BMJ (Clinical research ed.). 338. b2393. 10.1136/bmj.b2393.

Tufte, E.R. (2007). *The visual display of quantitative information.* 2nd ed., 5th printing. Cheshire, CT: Graphics Press.

Tukey, J.W. (1977) *Exploratory Data Analysis*. Addison-Wesley Publishing Company.

WatElectronics, n.d. *Types of programming languages with differences.* [online] Available at: https://www.watelectronics.com/types-of-programming-languages-with-differences/ [Accessed 12 November 2024].

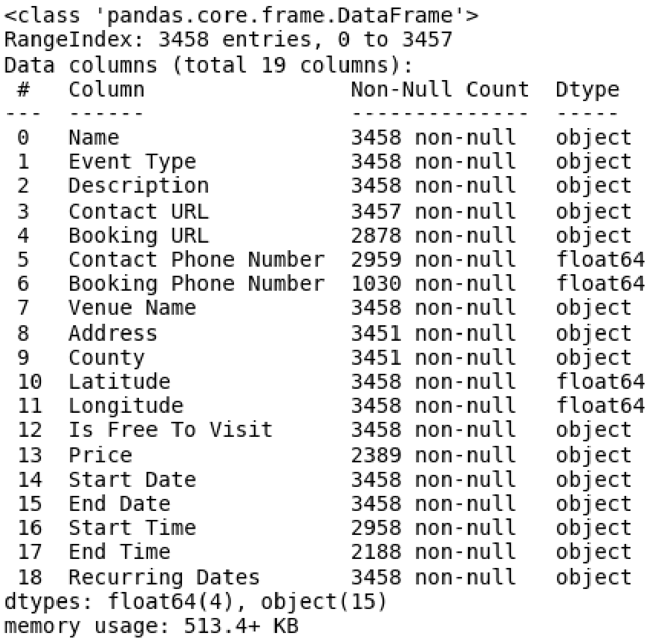
Zou, H., & Hastie, T. (2005). *Regularization and variable selection via the elastic net*. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 67(2), 301-320. https://doi.org/10.1111/j.1467-9868.2005.00503.x

Appendix

Appendix A. Data Collection

Events Dataset

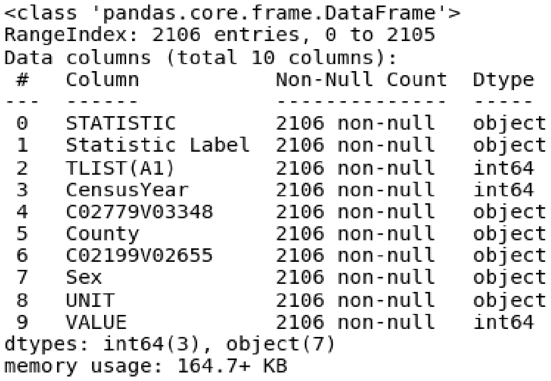
The Events dataset, sourced from Open Data provided by Fáilte Ireland, serves as the primary dataset for this analysis and contains 3458 records. The data was accessed and automatically fetched through an API provided by Fáilte Ireland, ensuring up-to-date event information. It is available through Fáilte Ireland's Research and Insights portal (Open Data - Fáilte Ireland). The dataset is licensed under the Open Data Directive (PSI) License, which allows for reuse with attribution.



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3458 entries, 0 to 3457
Data columns (total 19 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   Name                  3458 non-null   object 
 1   Event Type            3458 non-null   object 
 2   Description           3458 non-null   object 
 3   Contact URL           3457 non-null   object 
 4   Booking URL           2878 non-null   object 
 5   Contact Phone Number  2959 non-null   float64
 6   Booking Phone Number  1030 non-null   float64
 7   Venue Name            3458 non-null   object 
 8   Address               3451 non-null   object 
 9   County                3451 non-null   object 
 10  Latitude              3458 non-null   float64
 11  Longitude             3458 non-null   float64
 12  Is Free To Visit      3458 non-null   object 
 13  Price                 2389 non-null   object 
 14  Start Date            3458 non-null   object 
 15  End Date              3458 non-null   object 
 16  Start Time            2958 non-null   object 
 17  End Time              2188 non-null   object 
 18  Recurring Dates       3458 non-null   object 
dtypes: float64(4), object(15)
memory usage: 513.4+ KB
```

Population Dataset

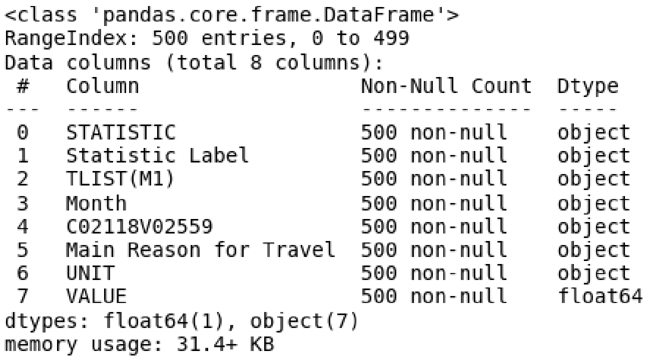
The Population by County (F1001) dataset, sourced from the Central Statistics Office (CSO), provides the latest population data from the 2022 Census (Central Statistics Office, 2022), with 2106 records. This data was accessed through an API provided by the CSO and is publicly available on the CSO's website. The dataset is licensed under the Creative Commons Attribution 4.0 International (CC BY 4.0) License, permitting reuse with appropriate attribution.



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2106 entries, 0 to 2105
Data columns (total 10 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   STATISTIC        2106 non-null   object
 1   Statistic Label  2106 non-null   object
 2   TLIST(A1)        2106 non-null   int64 
 3   CensusYear       2106 non-null   int64 
 4   C02779V03348     2106 non-null   object
 5   County           2106 non-null   object
 6   C02199V02655     2106 non-null   object
 7   Sex              2106 non-null   object
 8   UNIT             2106 non-null   object
 9   VALUE            2106 non-null   int64 
dtypes: int64(3), object(7)
memory usage: 164.7+ KB
```

Tourism Dataset

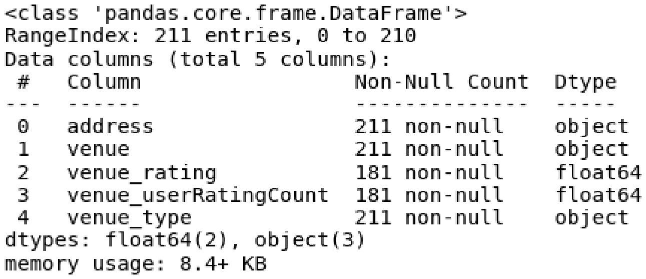
The Foreign Visitors by Main Reason for Travel (ITM04) dataset, also sourced from the CSO (Central Statistics Office, n.d.), with 500 records provides data on the number of foreign visitors categorized by their main reason for travel. It is available as well on the CSO's website and licensed under CC BY 4.0. This dataset was transformed from a "long" to a "wide" format using the pivot function to create a monthly summary of foreign visitors.



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 8 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   STATISTIC             500 non-null    object
 1   Statistic Label       500 non-null    object
 2   TLIST(M1)             500 non-null    object
 3   Month                 500 non-null    object
 4   C02118V02559          500 non-null    object
 5   Main Reason for Travel  500 non-null    object
 6   UNIT                  500 non-null    object
 7   VALUE                 500 non-null    float64
dtypes: float64(1), object(7)
memory usage: 31.4+ KB
```

Venue Dataset

The Venue Dataset was created using the Google Places API (Google, n.d.) to gather information on venue type, average rating, and user rating count. Each venue was searched by name and location, and the API returned details such as types, rating, and user_ratings_total. The data was parsed, and search parameters were adjusted to handle missing values, resulting in a structured dataset of 211 records, with key venue attributes for analysis.



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 211 entries, 0 to 210
Data columns (total 5 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   address                 211 non-null    object
 1   venue                   211 non-null    object
 2   venue_rating            181 non-null    float64
 3   venue_userRatingCount   181 non-null    float64
 4   venue_type              211 non-null    object
dtypes: float64(2), object(3)
memory usage: 8.4+ KB
```

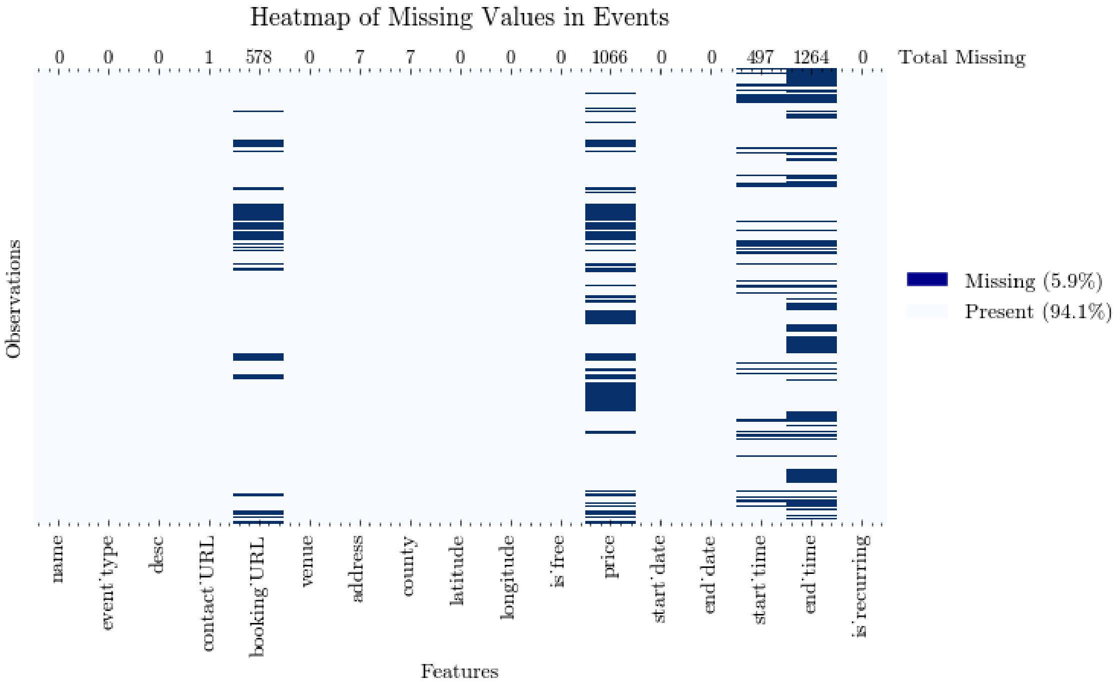
Appendix B. Data Dictionary

A data dictionary provides a metadata of the dataset, describing each column's name, data type, level of measurement, and role in the analysis. Understanding levels of measurement—nominal, ordinal, interval, or ratio—helps in choosing suitable statistical methods and analyses (Matthews, 2017).

Column Name	Data Type	Level of Measurement	Description
name	object	Nominal	Name of the event.
event_type	object	Nominal	Type of event (e.g., Event, Festival).
desc	object	Nominal	Description of the event.
booking_URL	object	Nominal	URL for booking tickets for the event.
venue	object	Nominal	Name of the venue where the event takes place.
address	object	Nominal	Full address of the venue.
county	object	Nominal	County where the event is hosted.
latitude	float64	Interval	Latitude coordinate of the venue.
longitude	float64	Interval	Longitude coordinate of the venue.
is_free	int64	Nominal	Indicates if the event is free (0 = no, 1 = yes).
price	float64	Ratio	Price of the event if not free.
start_date	object	Ordinal	Start date of the event.
end_date	object	Ordinal	End date of the event.
is_recurring	int64	Nominal	Indicates if the event is recurring (1 = yes, 0 = no).
duration	int64	Ratio	Duration of the event in days.
month	object	Ordinal	Month of the event.
is_weekend	int64	Nominal	Indicates if the event takes place during the weekend (1 = yes, 0 = no).
season	object	Nominal	Season in which the event occurs (e.g., autumn, winter).
booking_platform	object	Nominal	Platform used for booking the event (e.g., Ticketsolve).
distance_from_dublin	float64	Ratio	Distance from Dublin in kilometers.
population	int64	Ratio	Population of the county where the event takes place.
foreign_visitors	float64	Ratio	Estimated Ireland foreign tourist per month, in thousands.
venue_rating	float64	Interval	Average user rating of the venue.
venue_userRatingCount	float64	Ratio	Total number of user ratings for the venue.
venue_type	object	Nominal	Type of venue (e.g., event_venue).

Appendix C. Missing Values

To ensure dataset completeness and consistency, various strategies were employed to address missing values across different columns. In the Events dataset and Venues dataset several fields displayed missing values, which required targeted handling. In contrast, the Tourism and Population datasets were complete, with no missing values identified. This appendix explains the specific methods used to handle these gaps, based on the relevance of each field to the analysis.



Drop Time Columns

The start_time and end_time columns had a high number of missing values (497 and respectevely 1264) and were deemed non-essential for the analysis, so they were removed from the dataset.

Fill Missing Prices

For the 1066 missing values in the price column, different strategies were applied based on event characteristics. Events marked as "free" but with filled price values were treated as anomalies, resulting in 11 entries where the is_free column was updated to False. For genuinely free events with missing prices, 996 entries had their missing values set to 0, aligning with their designation as free. Non-free events with missing prices, which were critical for predictive modeling, accounted for 70 entries and were removed from the dataset to maintain data integrity.

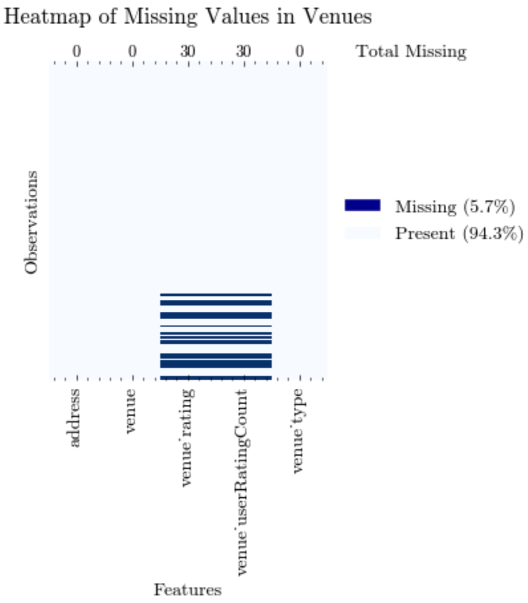
Fill Booking URLs

578 missing values were initially present in the booking_URL column. For non-free events, where the booking platform could be relevant for analysis, 13 entries lacked a booking_URL. Among these, two cases were manually updated by visiting the

contact_URL and retrieving the booking link from external sources. The remaining 576 missing values were supplemented using values from the contact_URL, which had no missing entries and served as a reliable proxy for booking information.

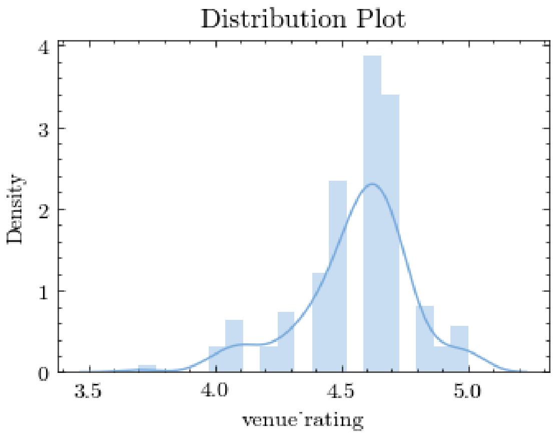
Drop address and county missing

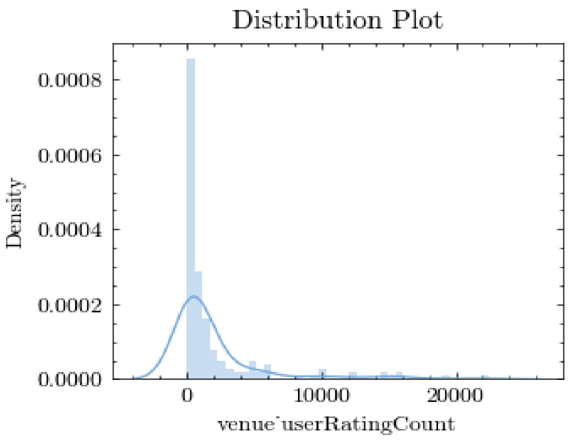
To examine whether county influences price, we will remove observations where county is missing, as only 7 rows are affected. Notably, address and county were missing in equal measure, reinforcing the decision to exclude these incomplete entries for a more accurate analysis.



Fill venue_rating

venue_rating and venue_userRatingCount with 30 missing values each, exhibit missing patterns linked to specific venue characteristics, such as street routes that inherently lack ratings. This suggests the missing values are Missing Not at Random (MNAR), making traditional imputation methods like mean or median inappropriate, as they assume randomness and could distort the data's structure (Sterne et al., 2009).





venue_rating is normally distributed around a central value, while venue_userRatingCount is clustered near zero. This supports using meaningful placeholders that maintain the data's original structure:

- venue_rating: Missing values are set to -1 to signify no available ratings, preserving the distribution without introducing arbitrary values.
- venue_userRatingCount: Missing values are filled with 0, indicating no user interaction logically rather than implying engagement.

Appendix D. Descriptive Statistics

This Appendix provides detailed descriptive statistics for key variables in the dataset, including measures of central tendency (such as mean, median, and mode), measures of dispersion (such as range, interquartile range, variance, and standard deviation) and Kurtosis. These statistics offer deeper insights into the distribution and variability of each variable, supplementing the analysis presented in the main report.

Numeric Columns

	latitude	longitude	is_free	price	is_recurring	duration
count	3349	3349	3349	3349	3349	3349
mean	53.0292	-6.90029	0.295312	18.1048	0.872499	94.8155
std	0.596368	0.96615	0.456251	21.0789	0.333583	121.84
min	51.5582	-10.2747	0	0	0	1
25%	52.6551	-7.21255	0	0	1	6
50%	53.3409	-6.27161	0	15	1	36
75%	53.3484	-6.26086	1	25.5	1	117
max	54.9558	-6.07421	1	265	1	363

	is_weekend	distance_from_dublin	population	foreign_visitors	venue_rating	venue_userRatingCount
count	3349	3349	3349	3349	3349	3349
mean	0.372649	65.4727	898616	317.11	4.45554	2680.58
std	0.483582	86.292	602854	24.9759	0.921045	4508.3
min	0	0.000872838	35199	145.5	-1	0
25%	0	0.788625	247774	310.935	4.4	103
50%	0	12.1265	1.45815e+06	318.048	4.6	809
75%	1	133.881	1.45815e+06	325.161	4.7	2148
max	1	313.772	1.45815e+06	382.065	5	22403

- Latitude and Longitude: The mean latitude (53.03) and longitude (-6.90) align closely with coordinates within Ireland. The range between the minimum and maximum values indicates that the data covers locations across the country.
- Is_Free: A mean of 0.30 suggests that 30% of events are free. Both the median and 25th percentile are 0, showing that the majority of events are not free.
- Price: The mean price is €18, with a median of €15. This suggests that most events are moderately priced, but a wide range of prices (up to €265) leads to a higher average.
- Is_Recurring: A mean of 0.87 indicates that 87% of events are recurring, as most values are 1.

- Duration: The average duration is 95 days, with a median of 36 days. This difference between the mean and median indicates a right-skewed distribution, with some events lasting much longer than the typical duration.
- Is_Weekend: The mean of 0.37 indicates 37% of events occur on weekends, as values are generally either 0 (weekday) or 1 (weekend).
- Distance_From_Dublin: The mean distance from Dublin is 65 km, while the median is 12 km, indicating that most events are closer to Dublin, but some far-off events increase the average.
- Population: The mean population is about 900,000, with a median of 1,458,154. This difference suggests that while some events are held in smaller towns, many occur in densely populated areas.
- Foreign_Visitors: The mean monthly visitors to Ireland is 317,000, with a median of 318,047. This close alignment of mean and median suggests a relatively symmetric distribution around these values.
- Venue Rating and User Rating Count: The mean venue rating is 4.45, with a median of 4.6, indicating generally high ratings. The mean user rating count is 2,681, but the median is 809, showing a few highly-rated venues with a large number of reviews that skew the average higher.

Text Columns

	name	event_type	desc	venue	address
count	3349	3349	3349	3349	3349
unique	600	3	599	198	190
top	Iconic Costumes of the Irish Silver Screen at The Museum of Style icons	Event	This exclusive exhibition, curated by renowned cos	Newbridge Silverware Museum of Style Icons	Newbridge Silverware Style Icon Museum, Athgarvan Rd, Kilbelin, Newbridge, Kildare
freq	363	3138	363	363	363

	county	month	season	booking_platform	venue_type
count	3349	3349	3349	3349	3349
unique	23	21	4	131	42
top	Dublin	2024 October	autumn	visitnewbridgesilverware	tourist_attraction
freq	1761	977	1838	363	642

- Name: There are 600 unique event names, reflecting a wide variety of events.
- Event_Type: With three unique types, “Event” is the most prevalent (3,138 occurrences), indicating that most listings are general events rather than specialized categories.

- Venue: The dataset includes 198 unique venues, showing diversity in event locations.
- Address: There are 190 unique addresses, slightly fewer than unique venues (198), suggesting that multiple venues may share the same address.
- County: Events are distributed across 23 counties, with Dublin being the most frequent (1,761 entries).
- Month: There are 21 unique month-year combinations, with October 2024 being the most common (977 entries), indicating a concentration of events during this period.
- Season: All four seasons are represented, with autumn being the most common (1,838 occurrences), suggesting a high concentration of events during this season.
- Booking_Platform: There are 131 platforms, with “visitnewbridgesilverware” as the top platform (363 occurrences), potentially associated with a specific venue.
- Venue_Type: Among 42 venue types, “tourist_attraction” is the most frequent, appearing 642 times, which suggests a large number of events are held at popular tourist destinations.

Kurtosis

Kurtosis is a statistical measure that describes the shape of a distribution’s tails in comparison to a normal distribution. Specifically, kurtosis indicates the presence of extreme values (outliers) and the “peakedness” or “flatness” of the distribution:

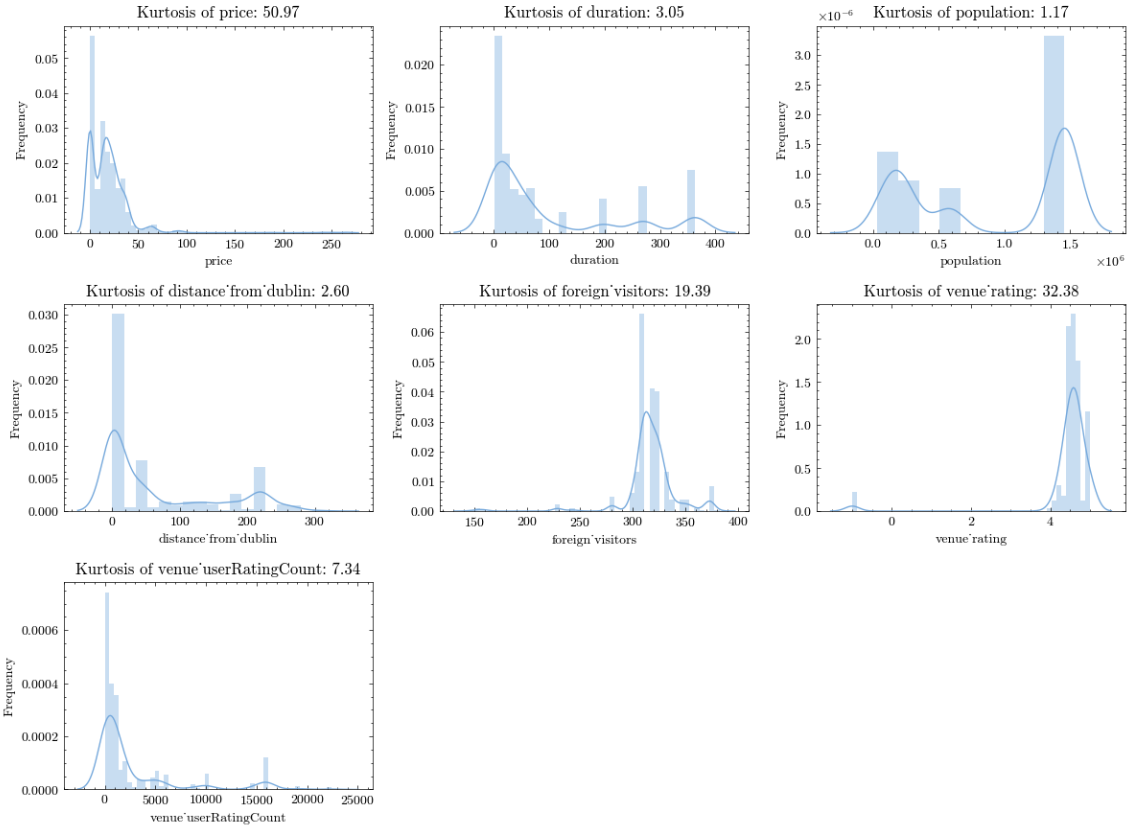
- High Kurtosis (Leptokurtic): A distribution with high kurtosis has heavier tails, meaning more extreme outliers. This results in a sharper peak around the mean and fatter tails.
- Low Kurtosis (Platykurtic): A distribution with low kurtosis has lighter tails, indicating fewer outliers and a flatter shape around the mean.
- Normal Kurtosis (Mesokurtic): A distribution with kurtosis close to 3 resembles a normal distribution.

$$K = \frac{n\sum_{i=1}^{n}(x_i - \bar{x})^4}{(\sum_{i=1}^{n}(x_i - \bar{x})^2)^2}$$

where: - K is the kurtosis. - n is the number of data points. - x_i represents each individual data point. - $\bar{x}$ is the mean of the dataset.

Kurtosis requires calculating the mean, deviations from the mean, and higher moments (specifically, the squared and fourth powers) of these deviations. As such, it is inherently applicable only to numeric data, where arithmetic operations are meaningful.

Below is a summary of the kurtosis and key characteristics of each feature:



- Price (Kurtosis = 50.97): This high kurtosis indicates a leptokurtic distribution with a sharp peak and heavy tails. Most prices are clustered around lower values, with a few high-priced events skewing the distribution and creating outliers.
- Duration (Kurtosis = 3.05): With a kurtosis close to 3, this distribution is approximately mesokurtic, resembling a normal distribution. Most event durations are short, but the balanced tail suggests a reasonable spread, with some events of extended duration.
- Population (Kurtosis = 1.17): The low kurtosis indicates a flatter distribution with fewer extreme population values, suggesting that event locations are spread across areas with varied but generally moderate population sizes, without extreme high or low outliers.
- Distance from Dublin (Kurtosis = 2.60): Slightly less than 3, this kurtosis suggests a platykurtic distribution with lighter tails and a less pronounced peak. Most events are near Dublin, with fewer at greater distances, indicating a wide spread without extreme outliers.
- Foreign Visitors (Kurtosis = 19.39): The high kurtosis reflects a sharp peak around typical visitor numbers with heavy tails, indicating extreme values. The distribution is concentrated around common visitor counts, with some months showing significantly higher or lower numbers, likely due to seasonal tourism trends.
- Venue Rating (Kurtosis = 32.38): The high kurtosis shows that most venues have ratings close to the mean, with only a few venues rated significantly differently. This concentrated peak suggests that venue quality is generally consistent, with few venues experiencing notably different ratings.

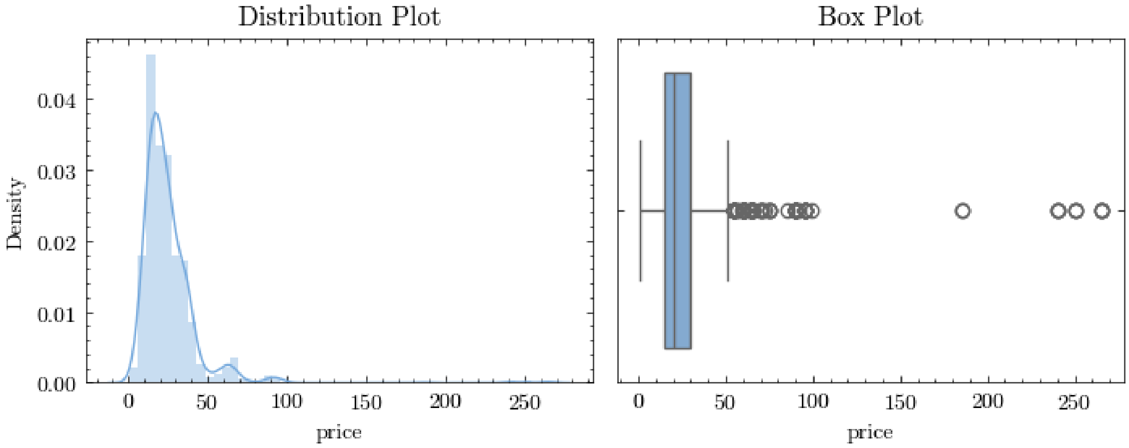
- Venue User Rating Count (Kurtosis = 7.34): This high kurtosis suggests a leptokurtic distribution, with most venues having low to moderate rating counts, while a few popular venues receive a large number of reviews, stretching the distribution tail.

Appendix F. Outlier

Outliers are data points that significantly deviate from the general distribution of the dataset. They are critical to identify and interpret, as they can influence statistical analysis and model performance. This Appendix evaluates the presence and implications of outliers in key features, based on box plots and distribution plots. The Interquartile Range (IQR) method was applied to highlight outliers in skewed distributions, while a normality assessment was conducted for features resembling normal distributions. Below are key findings from this analysis.

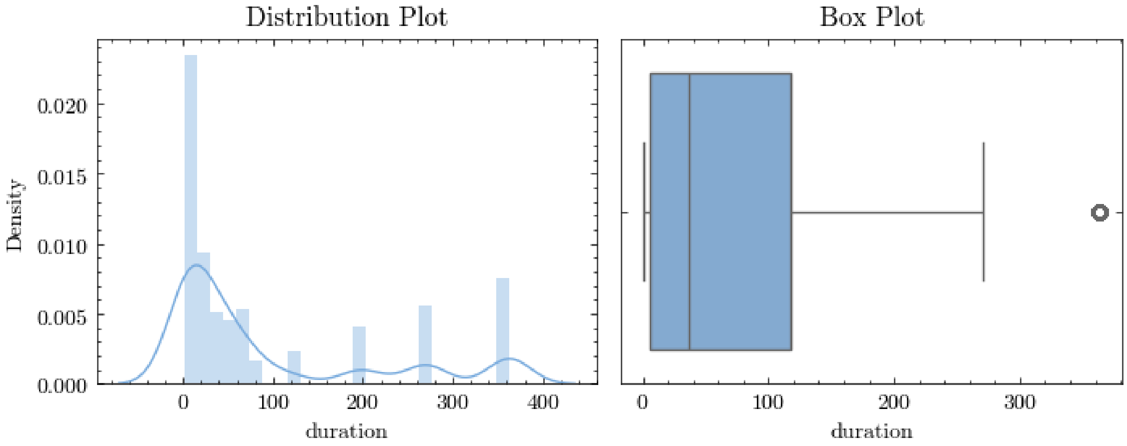
Price

Focusing on paid events, the distribution of prices was analysed to understand typical price levels. Most events were moderately priced, but a few high-priced events (e.g., tickets over €50) stood out as outliers, indicating high-demand events. These outliers were retained as they represent natural market variations. However, four extreme values (185, 240, 250, and 265) were reviewed and deemed atypical for this dataset. Based on further inspection, these were multi-day events, and therefore, excluded from further analysis.



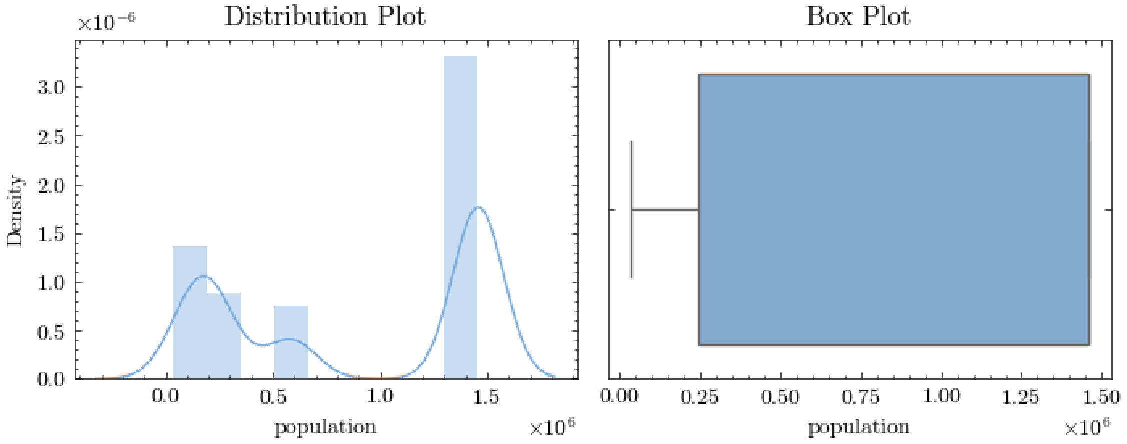
Duration

Event duration presented a right-skewed distribution, with the majority of events lasting less than 100 days. Although some events extended to more than 350 days, these long durations reflect legitimate events rather than data anomalies and were therefore retained in the dataset.



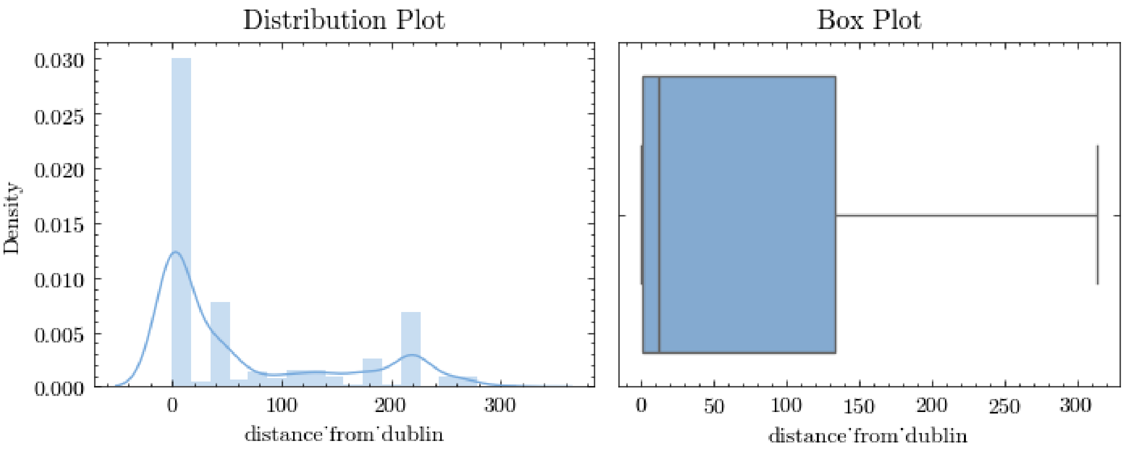
Population

The population distribution across event locations shows two primary peaks: one for densely populated areas and another for less populated counties. This variation reflects the natural differences in population density across regions rather than statistical outliers. The box plot confirms that there are no extreme outliers in population data for this dataset, and as such, all values were included in the analysis.



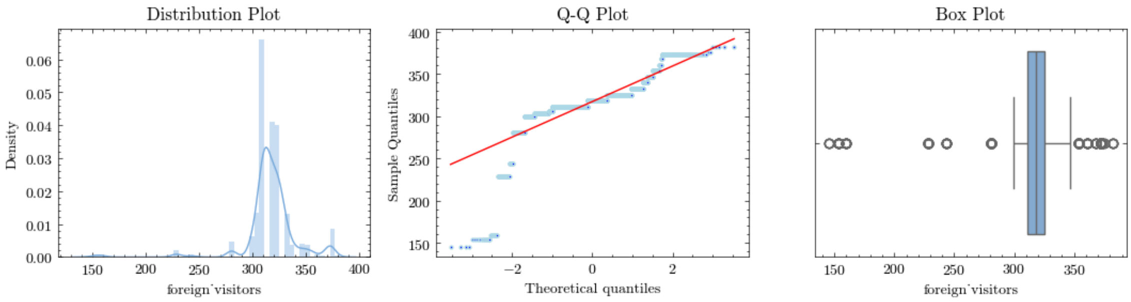
Distance from Dublin

The distance from Dublin distribution is skewed towards zero, reflecting the clustering of events near Dublin. This skewness and the long tail for more distant locations align with geographic patterns, not statistical outliers.



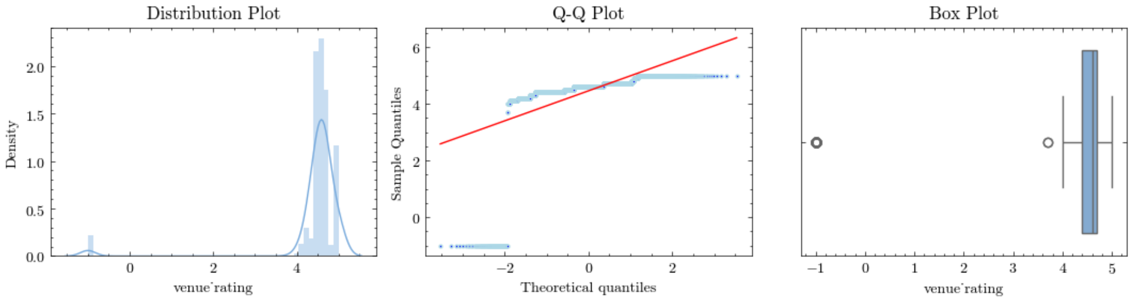
Foreign Visitors

Monthly foreign visitor numbers demonstrated a near-normal distribution, with a few seasonal spikes and lows that deviate from the trend. These deviations likely correspond to seasonal tourism patterns rather than true outliers, making them valuable data points for analysis.



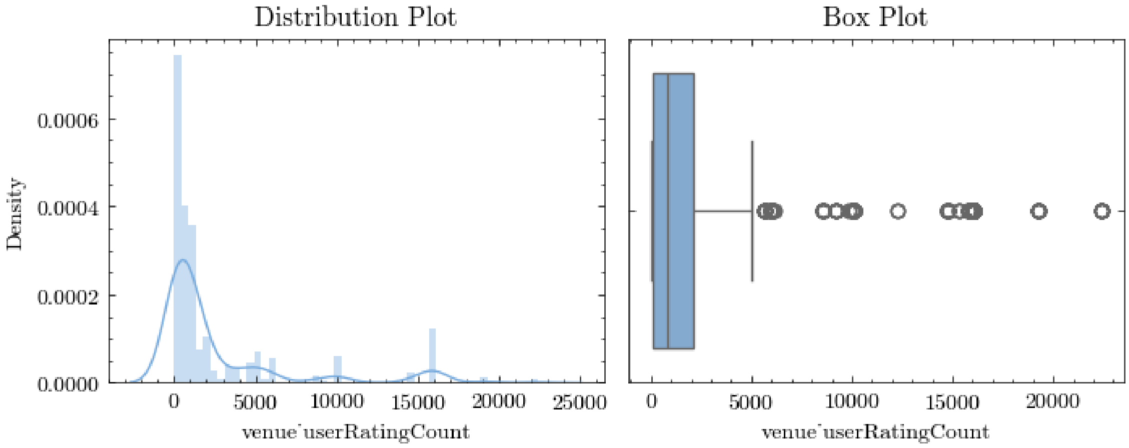
Venue Rating

The majority of venues have ratings around 4.5, with -1 values representing intentionally imputed missing ratings rather than actual outliers.



Venue User Rating Count

This feature is right-skewed, with a few venues receiving significantly high user ratings compared to others. While most venues have moderate counts, a handful have reviews in the thousands, representing popular venues. These high counts provide insights into venue popularity and were thus included in the analysis.


Distribution Plot
Box Plot
Density
0.0006
0.0004
0.0002
0.0000
0
5000
10000
15000
20000
25000
venue˙userRatingCount
0
5000
10000
15000
20000
venue˙userRatingCount

Appendix G. Events Analysis

Free Events vs County and Population

	county	Total_Events	Free_Events	population	Free Events (%)
0	Monaghan	1	1	65288	100
1	Mayo	9	5	137970	55.56
2	Donegal	6	3	167084	50
3	Cavan	7	3	81704	42.86
4	Kildare	14	5	247774	35.71
5	Galway	6	2	277737	33.33
6	Kilkenny	14	4	104160	28.57
7	Limerick	5	1	209536	20
8	Cork	59	11	584156	18.64
9	Westmeath	7	1	96221	14.29
10	Waterford	21	3	127363	14.29
11	Kerry	78	8	156458	10.26
12	Dublin	386	33	1.45815e+06	8.55
13	Longford	26	1	46751	3.85
14	Sligo	2	0	70198	0
15	Wexford	21	0	163919	0
16	Carlow	1	0	61968	0
17	Roscommon	1	0	70259	0
18	Meath	5	0	220826	0
19	Louth	10	0	139703	0
20	Clare	1	0	127938	0
21	Wicklow	10	0	155851	0

Free Events vs Season

	season	Total_Events	Free_Events	Free Events (%)	Paid_Events
0	autumn	519	59	11.37	460
1	spring	21	6	28.57	15
2	summer	23	4	17.39	19
3	winter	127	12	9.45	115

Summary table of Median Price by Event Types

event_type	count	median_value
Event	2235	20
Festival	113	15

Summary table of Median Price by County

county	count	median_value
Carlow	3	95
Sligo	5	70
Kildare	25	60
Meath	32	36
Wexford	43	30
Cork	104	25
Kerry	103	25
Mayo	8	25
Dublin	1512	23.5
Louth	83	20
Limerick	16	20
Kilkenny	11	20
Wicklow	92	20
Longford	33	18
Roscommon	17	15
Cavan	4	14.5
Galway	19	14
Westmeath	14	12
Donegal	4	10
Waterford	219	10
Clare	1	10

Summary table of Median Price by Season

season	count	median_value
winter	685	25
autumn	1370	20
spring	105	15
summer	188	15

Summary table of Median Price by Venue Type

venue_type	count	median_value
meal_delivery	56	39
hotel	22	35
natural_feature	1	32.5
performing_arts_theater	429	31.5
visitor_center	17	30

venue_type	count	median_value
sports_complex	6	28
movie_theater	38	25
point_of_interest	71	25
cafe	28	25
store	39	24
gift_shop	53	24
park	54	23.5
art_gallery	34	22
brunch_restaurant	2	21
church	4	20
travel_agency	2	20
parking	12	20
museum	154	20
zoo	9	19.95
tourist_attraction	539	18
street_address	3	17.5
coffee_shop	5	17
night_club	54	16.725
locality	33	15
event_venue	601	15
stadium	22	15
community_center	20	15
national_park	2	12
library	4	12
cultural_center	3	12
farm	11	11
bar	6	10
local_government_office	11	6
administrative_area_level_2	2	5
amusement_park	1	3

Appendix H. Feature encoding

Two distinct encoding strategies were applied for categorical variables, tailored to each model. For the decision tree model, which inherently handles categorical features, label encoding was used for compatibility with scikit-learn. After addressing rare categories with rare encoding, LabelEncoder assigned unique numeric labels to each category, enabling effective processing.

For linear regression, the encoding method was selected based on the cardinality of each feature:

- Event_Type (Cardinality: 2): As a binary feature with no ordinal relationship, One-Hot Encoding was applied.
- County (Cardinality: 21): With moderate cardinality, One-Hot Encoding was deemed feasible, especially for manageable sample sizes.
- Month (Cardinality: 21): Representing specific months across multiple years, Label Encoding was implemented to capture distinct time periods.
- Season (Cardinality: 4): Given the temporal order of seasons, Label Encoding assigned integers reflecting their position within the year.
- Booking_Platform (Cardinality: 98): High cardinality was addressed through Rare Encoding to group less frequent platforms into a "Rare" category, followed by One-Hot Encoding for final processing.
- Venue_Type (Cardinality: 35): Rare Encoding or Frequency Encoding was applied initially, with One-Hot Encoding used for the remaining categories.

Appendix I. Log Transformation

Log Transform

The logarithmic function, or log, is the inverse of the exponential function, defined as $\log_a(a^x) = x$, where a is a positive base and x is any positive number. The log function compresses large numbers and expands small ones. This compression reduces skewness, stabilizes variance, and normalizes data, making it particularly useful for features with wide ranges or extreme values.

Since logs are undefined for zero or negative values, a small constant is often added to ensure all values remain positive. This makes log transformation effective for improving model performance and handling skewed data distributions.

Skewness is calculated using the following formula:

$$\text{Skewness} = \frac{n}{(n-1)(n-2)} \sum_{i=1}^{n} \left(\frac{x_i - \bar{x}}{s} \right)^3$$

Where:

- n is the number of data points.
- x_i is each individual data point.
- $\bar{x}$ is the mean of the data.
- s is the standard deviation of the data.

The resulting skewness value indicates the degree to which the distribution deviates from symmetry:

- If skewness > 0, the distribution is positively skewed, with data points more spread out to the right of the mean.
- If skewness < 0, the distribution is negatively skewed, with data points more spread out to the left of the mean.

In Pandas, the .skew() method utilizes this formula to calculate the skewness for numerical columns in a dataset.

	Skewness Before	Skewness After Log
price	1.798	-0.438
duration	4.758	0.893
distance_from_dublin	1.119	-0.053
population	-0.439	-0.718
foreign_visitors	-3.745	-5.835
venue_userRatingCount	2.249	-3.88

An increase in skewness for population and foreign_visitors was observed after applying the log transformation. This effect can occur when the data contains numerous small positive values close to zero, as the log function disproportionately stretches these values, potentially increasing the skew. Furthermore, if the original distribution is already symmetrical or has a limited range with clustered values, the log transformation may distort the data rather than normalize it. The presence of outliers can also diminish the effectiveness of the transformation, as the log transformation may not adequately reduce the influence of extreme values relative to smaller ones.

Box-Cox Transform

The Box-Cox transformation can address these issues more effectively due to its flexibility, which includes an adjustable parameter, λ. This parameter allows for different power transformations to be applied to the data, enabling the transformation to find the optimal power that best normalizes the data, reduces skewness, and stabilizes variance. Unlike a log transformation, which uses a fixed base, the Box-Cox transformation can adapt to the data's characteristics by tuning λ. This makes it particularly useful for handling data with small positive values, a limited range, or near-symmetrical distributions, resulting in a more appropriate adjustment than a simple log transformation.

The Box-Cox transformation is defined by the following formula:

$$y' = \begin{cases} \frac{y^\lambda - 1}{\lambda}, & \text{if } \lambda \neq 0 \\ \ln(y), & \text{if } \lambda = 0 \end{cases}$$

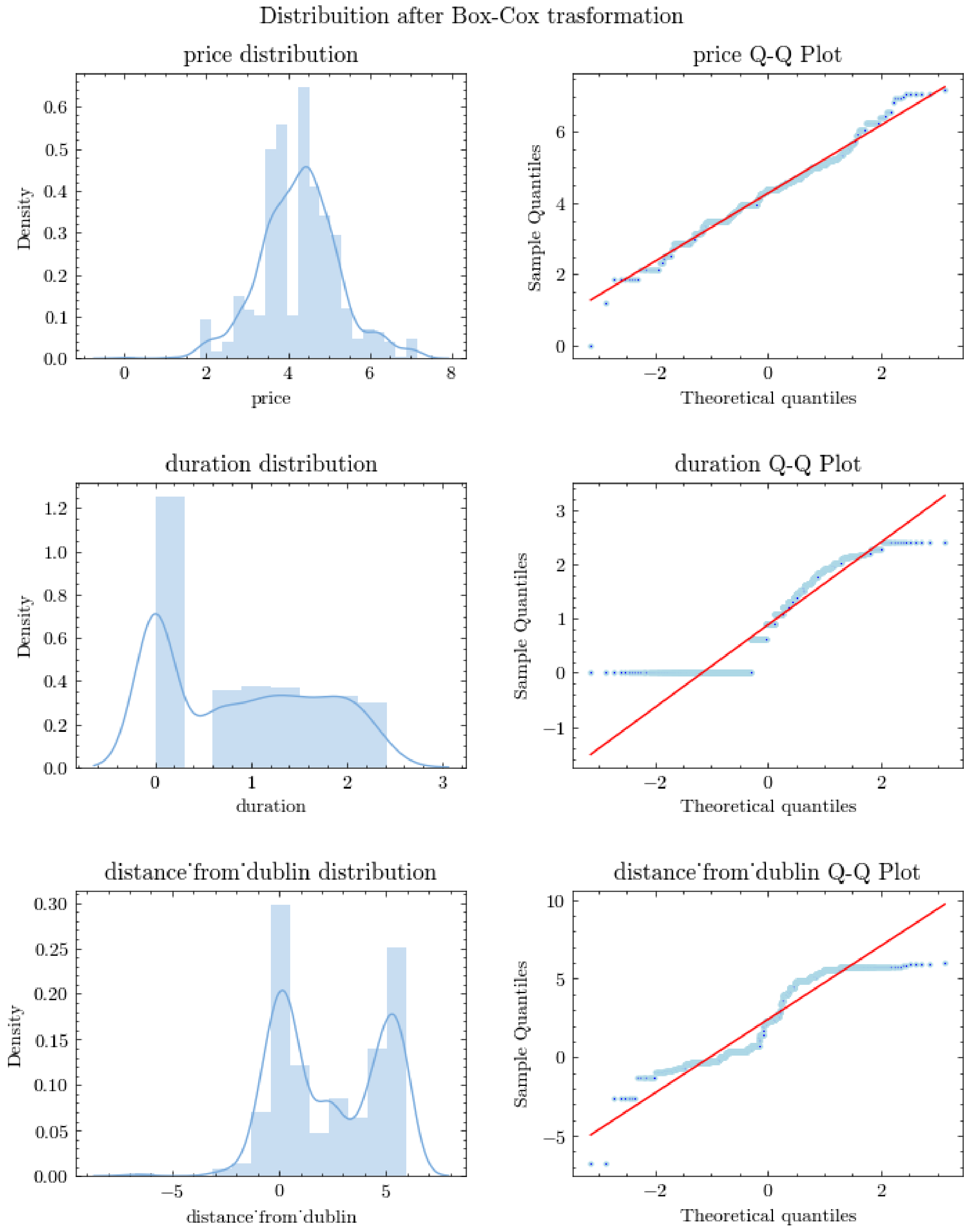
Where:

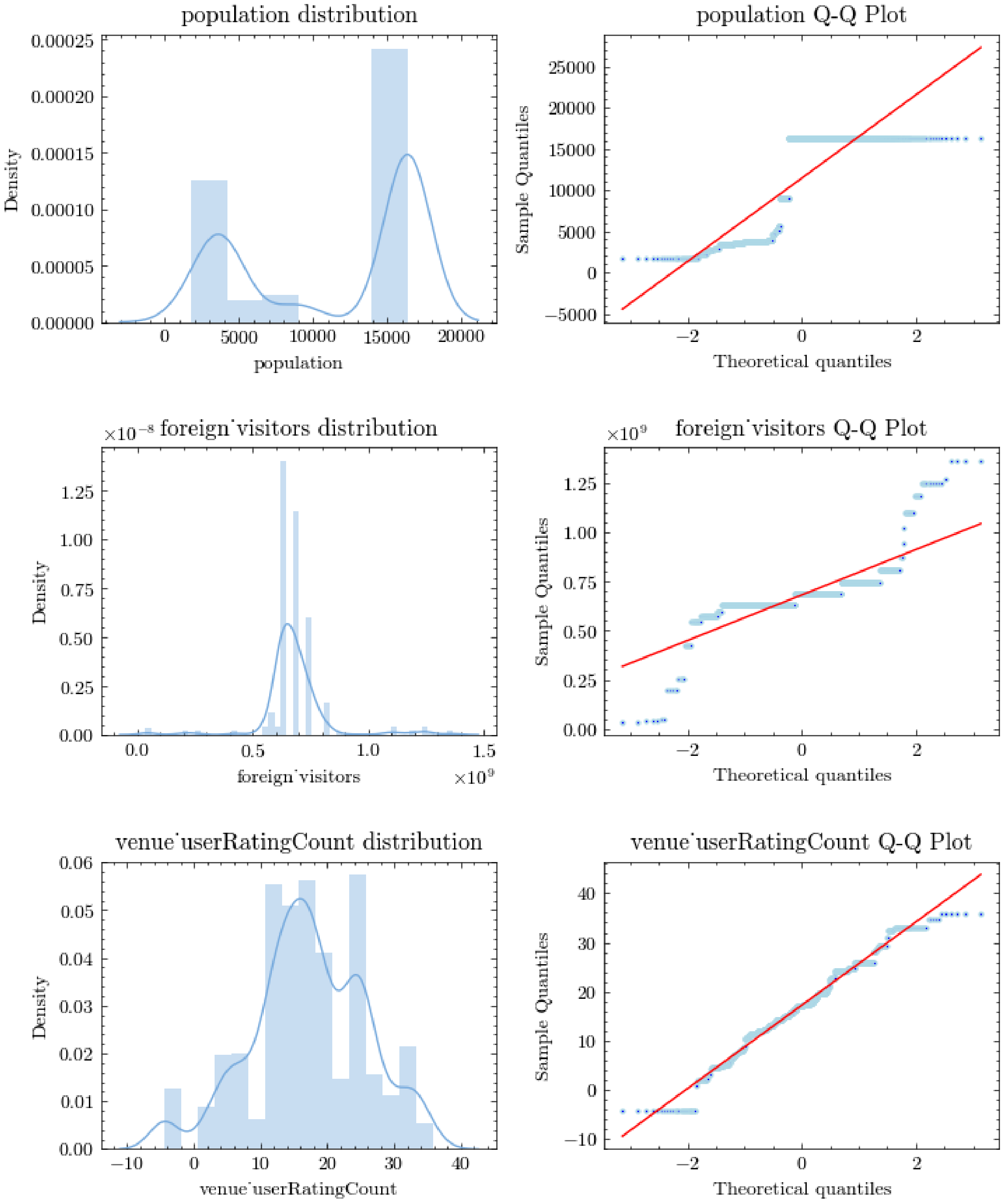
- y is the original data value.
- y' is the transformed data value.
- λ is the transformation parameter that is optimized to best normalize the data.
- $\ln(y)$ represents the natural logarithm of y (used when λ = 0)

	Skewness Before	Skewness After Log	Skewness After Box-Cox	Lambda (Box-Cox)
price	1.798	-0.438	0.031	0.181
duration	4.758	0.893	0.264	-0.358
distance_from_dublin	1.119	-0.053	-0.012	0.013
population	-0.439	-0.718	-0.488	0.654
foreign_visitors	-3.745	-5.835	0.843	3.76
venue_userRatingCount	2.249	-3.88	-0.222	0.217

- Price (λ = 0.181): The λ value indicates a transformation slightly different from a pure log function. The distribution is effectively normalized with minimal residual skewness.
- Duration (λ = -0.358): The negative λ suggests a transformation reducing the impact of larger values. The transformation brings noticeable improvement, although the presence of a peak at zero suggests many short durations remain.
- Distance from Dublin (λ = 0.013): With a λ near zero, the transformation closely resembles a log function. The distribution becomes more normalized, but some variation between groups persists.

- Population (λ = 0.654): The positive λ represents a mild power transformation, which improves normality but retains some peaks, reflecting possible bimodal characteristics in the data.
- Foreign Visitors (λ = 3.76): A large λ indicates a strong power transformation. While this reduces extreme values effectively, normalization may still be incomplete due to a sharp peak and residual clusters.
- Venue User Rating Count (λ = 0.217): The small positive λ achieves a slight power transformation, reducing skewness and moving the distribution closer to normal.





The Box-Cox transformation effectively stabilizes variance and reduces skewness for most features, enhancing their suitability for regression modeling. However, features like foreign_visitors and duration still exhibit characteristics (e.g., peaks or clusters) that may require additional pre-processing techniques, such as binning or segmentation, for further improvement. These adjustments can be explored in future iterations.

Appendix J. Feature Scaling

Feature scaling is a data pre-processing technique that adjusts the scales of numeric features, making them comparable, a critical step in many machine learning models.

- Normalization rescales features to a specified range, commonly [0, 1] or [-1,1]. This approach is useful when data doesn't follow a Gaussian distribution.

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

- Standardization centres data around a mean of 0 with a standard deviation of 1, making it ideal for algorithms that rely on distance metrics (e.g., k-means clustering, SVM).

$$X_{std} = \frac{X - \mu}{\sigma}$$

- Robust scaling utilizes the median and interquartile range (IQR), making it resistant to outliers.

$$X_{robust} = \frac{X - \text{median}}{\text{IQR}}$$

In the log transformation section, it was observed that the numeric variables exhibited varying degrees of skewness and did not follow a normal distribution. Additionally, as noted in the outlier's section, a substantial number of outliers were present in these variables. Therefore, robust scaling was applied to reduce the model's sensitivity to these variations.

	Mean Before Scaling	Median Before Scaling	Std Dev Before Scaling	Mean After Scaling	Median After Scaling	Std Dev After Scaling
duration	16.784	3	43.011	1.378	0	4.301
distance_from_dublin	67.952	9.848	92.701	0.519	0	0.828
population	945894	1.45815e+06	624224	-0.394	0	0.48
foreign_visitors	315.771	318.048	21.562	-0.32	0	3.031
venue_userRatingCount	3072.86	1258	4357.42	0.429	0	1.03

After applying robust scaling, each feature was centred around zero, leading to a more standardized dataset:

- Duration: Centred with a mean of 1.378 and a standard deviation of 4.301, reflecting its broad range after scaling.
- Distance from Dublin: Achieved a mean of 0.519 and a standard deviation of 0.828, indicating a compressed spread post-scaling.
- Population: Adjusted to a mean of -0.394 with a standard deviation of 0.48, showing significant compression from its original scale.
- Foreign Visitors: cantered at a mean of -0.32 with a standard deviation of 3.031, reducing sensitivity to outliers while retaining variability.

- Venue User Rating Count: Scaled to a mean of 0.429 with a standard deviation of 1.03, minimizing the effect of larger counts.

Appendix K. Model Evaluation

Ordinary least squares (OLS)

Linear Regression, specifically Ordinary Least Squares (OLS) Linear Regression, is a statistical method used to model the relationship between a continuous target variable and one or more features by fitting a linear equation to the data. OLS estimates the coefficients by minimizing the sum of squared differences between the predicted and actual values, a process that seeks to minimize the Mean Squared Error (MSE). This approach makes OLS linear regression straightforward to implement and interpret: each feature's coefficient represents its direct contribution to the target variable, allowing easy insights into feature importance.

However, OLS relies on several assumptions, including linearity, independence of errors, homoscedasticity (constant variance of errors), and normality of error distribution. While it performs well under these assumptions, OLS linear regression may struggle with complex, non-linear patterns, sensitivity to outliers, and issues with multicollinearity (high correlations between features), which can distort coefficient estimates. In such cases, techniques like regularized regression (e.g., Ridge, Lasso, or ElasticNet) may be more effective in creating a stable and interpretable model.

To evaluate how well the model generalizes, we compare its performance on the training set and the test set. A significant discrepancy between these scores (e.g., a high training score but a low test score) may indicate overfitting. In regression analysis, model performance is commonly evaluated using the following metrics:

- Mean Squared Error (MSE): Calculates the average of the squared differences between actual and predicted values, penalizing larger errors more heavily to highlight significant prediction deviations.
- Mean Absolute Error (MAE): Measures the average of absolute errors, offering an interpretable metric of average prediction error that is less influenced by outliers.
- R2 Score: Indicates the proportion of variance in the target variable explained by the model, with a score close to 1 implying a strong model fit. Useful for comparing different models' performances and assessing how well the model generalizes beyond the training data.

```
Training Set Performance:
Mean Absolute Error (MAE): 7.127
Mean Squared Error (MSE): 114.366
Root Mean Squared Error (RMSE): 10.694
R2 Score: 0.532

Test Set Performance:
Mean Absolute Error (MAE): 6.849
Mean Squared Error (MSE): 107.446
Root Mean Squared Error (RMSE): 10.366
R2 Score: 0.296
```

For final model evaluation, the test score is generally more important as it represents the model's ability to generalize to new data. However, during model selection and tuning, the cross-validation score is crucial, as it helps assess model stability and guides adjustments prior to final testing

Cross-validation reliably assesses a model’s generalization by splitting data into k folds, training on k-1 folds, and testing on the remaining fold, repeated k times. It provides an average score to account for data variability and reduces overfitting risk. Additionally, cross-validation estimates score variance, helping gauge whether differences between models are statistically meaningful and ensuring robust model selection.

5-fold cross-validation is often chosen as a balanced approach, offering a reliable assessment without excessive computation. Fewer folds may underestimate variability, while more folds increase computation time without significantly improving accuracy.

```
Score metric: r2
CV scores mean: 0.309 (+/- 0.192)
CV scores std: 0.096
CV folds score: ['0.404', '0.331', '0.126', '0.360', '0.326']
```

The alignment between the cross-validation R2 score and the test R2 score indicates that the model is relatively stable, showing similar performance on unseen data as during cross-validation. However, the overall low R2 scores (around 0.30) imply that this model explains only about 30% of the variance in the target variable. This suggests that a linear model may not capture the data’s complexity effectively.

Before exploring non-linear techniques, I will apply regularization methods such as Ridge or Lasso regression to address the instability observed across cross-validation folds. Regularization may help reduce the standard deviation in cross-validation scores by diminishing the influence of less relevant features, thereby potentially improving generalization and stabilizing the model’s performance across different data subsets.

```
[  -6.2491311     1.00540828    0.15898222    0.19751119    1.31117194
  -1.75655103   -3.85935951    0.18261664    0.15392073    0.52925892
   3.30978958   -3.30978958   75.85044395  -13.65635444  -12.78313579
   3.08043741  -11.17298596   -0.30041627   -6.48320331   -4.79545951
 -14.17922106   -8.98303642    1.51611979  -12.23651574   11.04317026
   7.01697506    0.36745399  -10.76134113   13.95361019  -12.24726249
  -5.40859145    2.93796449   -2.75865158    8.62074214   -9.68219017
  47.09028301  -22.65217387   -5.69856245  -10.28727691   -6.26487942
 -32.47632401  -14.77669039   22.31054902   -4.5442242    -2.43238973
   1.75550439    0.2923212    22.47630018    9.86963723    3.62450778
  -7.2251338    -6.96381355    6.36955852    1.16642162   -2.90084891
  20.47068583  -24.31114853    4.98027222  -10.27411518    9.54031521
  -1.14438072   -4.2401119    -1.33076228    0.96978611    1.1516872
   7.86318582  -1.34673144]
```

Elastic Net

Elastic Net is a regularization technique that combines both Lasso (L1) and Ridge (L2) penalties, allowing it to perform both feature selection (like Lasso) and coefficient shrinkage (like Ridge). This hybrid approach makes Elastic Net particularly useful when there are multiple correlated features, as it can stabilize the model and enhance generalization. Elastic Net uses two parameters:

- alpha: controls the overall strength of the regularization. A higher alpha value increases the penalty, shrinking the coefficients more and potentially removing irrelevant features.
- l1_ratio: determines the balance between Lasso and Ridge regularization. An l1_ratio of 1 means pure Lasso, 0 means pure Ridge, and values between 0 and 1 give a mixture of both.

Grid Search is a hyperparameter tuning technique used to identify the optimal parameters for a model by exhaustively testing combinations within a defined range. It systematically evaluates each combination of hyperparameters through cross-validation, selecting the configuration that yields the best performance.

```
Hyperparameter = { ‘alpha’: np.arange(0, 1.0, 0.01), ‘l1_ratio’: np.arange(0.1, 1.1, 0.1)}
Fitting 5 folds for each of 1000 candidates, totalling 5000 fits
Score metric: None
CV scores mean: 0.344 (+/- 0.174)
CV scores std: 0.087
CV folds score: ['0.389', '0.405', '0.171', '0.383', '0.371']
Best Parameters:
{'alpha': 0.03, 'l1_ratio': 1.0}

Training Set Performance:
Mean Absolute Error (MAE): 7.406
Mean Squared Error (MSE): 120.881
Root Mean Squared Error (RMSE): 10.995
R2 Score: 0.506

Test Set Performance:
Mean Absolute Error (MAE): 6.701
Mean Squared Error (MSE): 102.873
Root Mean Squared Error (RMSE): 10.143
R2 Score: 0.326
```

The results show a mean cross-validation R2 score of 0.34 with a standard deviation of 0.09, indicating variability across folds. The best parameters include a low alpha (0.03) and a high l1_ratio (1), suggesting reliance on Lasso regularization.

To reduce score variability, stronger regularization can be applied by increasing alpha and adjusting l1_ratio to include more Ridge (L2) regularization. Testing a wider range of higher alpha values and slightly lower l1_ratio values, or using ElasticNetCV for automatic tuning, may lead to more consistent cross-validation results.

```
Previous best hyperparameters = {‘alpha’: 0.03, ‘l1_ratio’: 1.0}

Hyperparameter = {‘alpha’: 0.1, ‘l1_ratio’: 0.9}
```

```
Score metric: r2
CV scores mean: 0.292 (+/- 0.097)
CV scores std: 0.048
CV folds score: ['0.311', '0.338', '0.198', '0.304', '0.309']
```

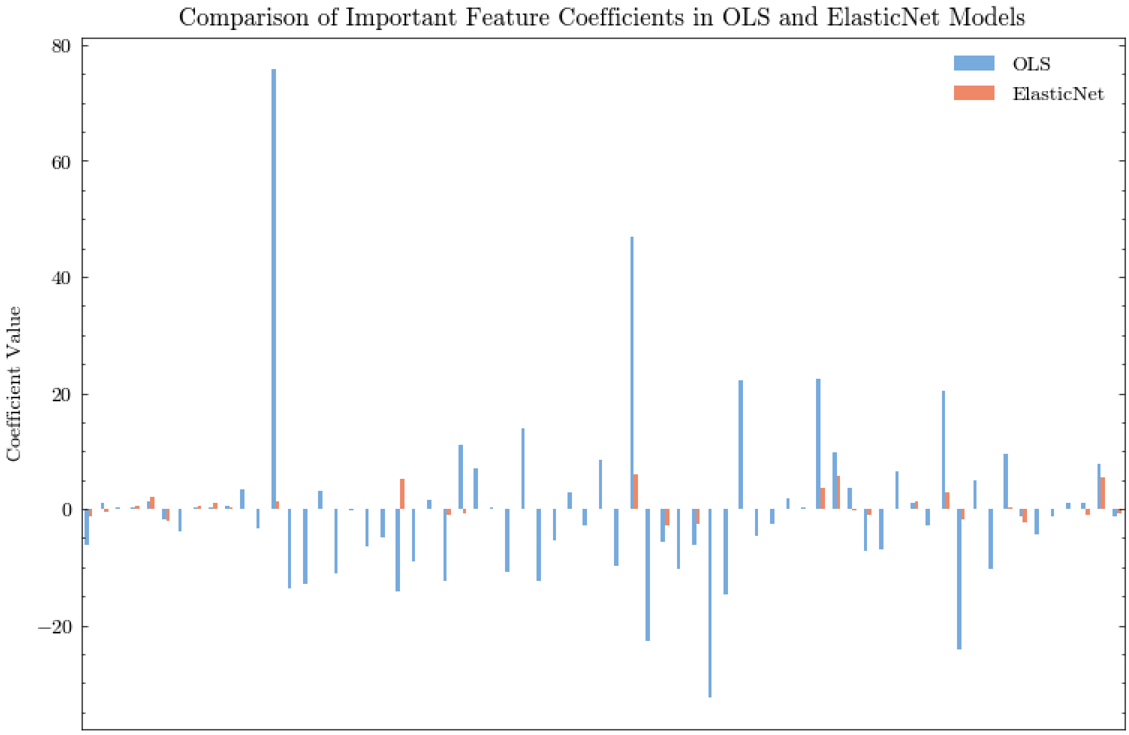
```
Hyperparameter = {‘alpha’: 0.2, ‘l1_ratio’: 0.8}
Score metric: r2
CV scores mean: 0.225 (+/- 0.047)
CV scores std: 0.023
CV folds score: ['0.211', '0.252', '0.187', '0.236', '0.241']
```

```
Coefficients: {‘alpha’: 0.2, ‘l1_ratio’: 0.8}
[-1.23864865  -0.57127647   0.12414766   0.53775375   2.05528031  -1.91870811
  0.           0.45010479   0.93734125   0.21860633   0.          -0.
  1.23607     -0.          -0.           0.          -0.           0.
 -0.          -0.           5.18672419  -0.           0.          -0.98665018
 -0.63945665   0.           0.          -0.           0.          -0.
 -0.          -0.          -0.           0.          -0.           6.04042216
 -0.          -2.93004039  -0.          -2.47369061   0.          -0.
  0.          -0.          -0.          -0.          -0.           3.69525607
  5.58285161  -0.36248278  -1.03145255  -0.08408666  -0.           1.20232244
  0.           2.8447017   -1.65736686   0.          -0.           0.38208164
 -2.18399146  -0.           0.           0.          -0.92607063   5.38857226
 -0.63027903] 21.351752484019947
```

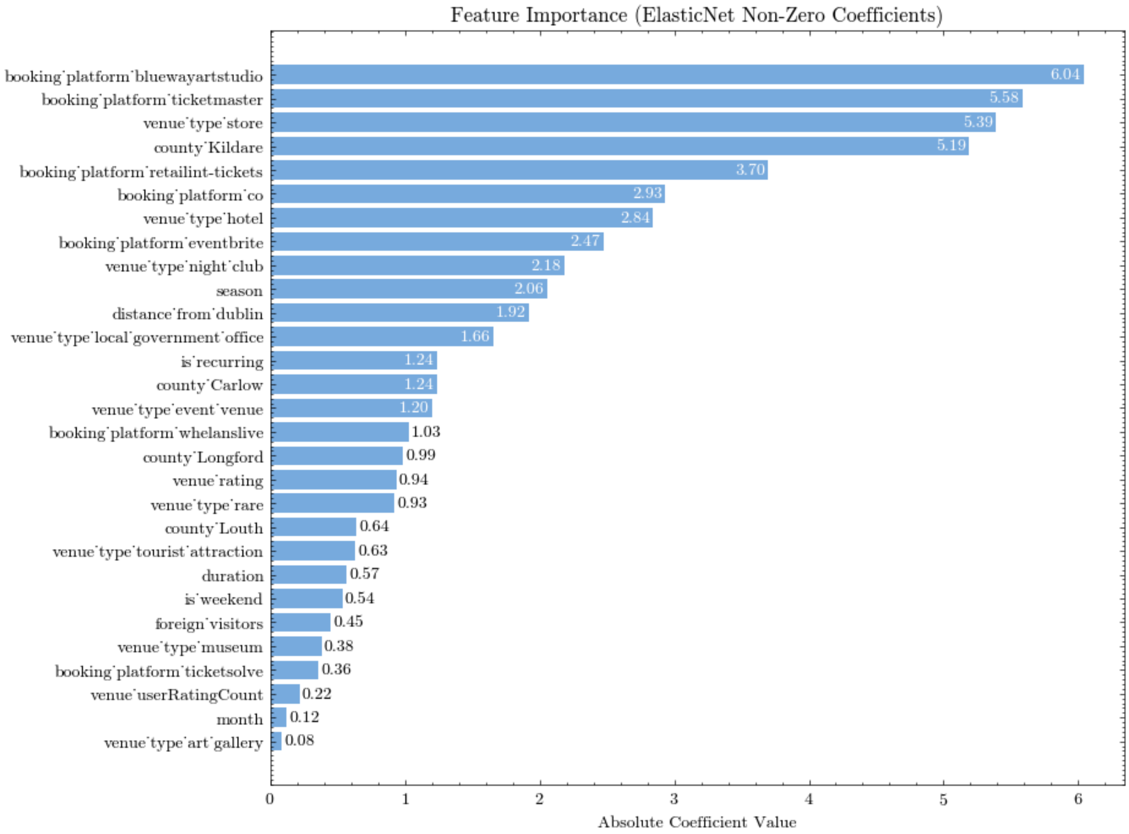
The tuned ElasticNet model demonstrates a balanced combination of model stability across folds and predictive performance. Although it achieves a modest mean R2 of 0.23, it also has a lower standard deviation of 0.02 across cross-validation folds, indicating consistent performance and minimal sensitivity to specific subsets of the training data. The Lasso component, weighted with an l1_ratio of 0.8, emphasizes feature selection by reducing the coefficients of irrelevant features to zero. Meanwhile, the Ridge component addresses multicollinearity by distributing influence across correlated features, thereby preventing any single feature from dominating the model.

Feature	OLS	ElasticNet
is_recurring	-6.24913	-1.23865
duration	1.00541	-0.571276
month	0.158982	0.124148
is_weekend	0.197511	0.537754
season	1.31117	2.05528
distance_from_dublin	-1.75655	-1.91871
foreign_visitors	0.182617	0.450105
venue_rating	0.153921	0.937341
venue_userRatingCount	0.529259	0.218606
county_Carlow	75.8504	1.23607
county_Kildare	-14.1792	5.18672
county_Longford	-12.2365	-0.98665

Feature	OLS	ElasticNet
county_Louth	11.0432	-0.639457
booking_platform_bluewayartstudio	47.0903	6.04042
booking_platform_co	-5.69856	-2.93004
booking_platform_eventbrite	-6.26488	-2.47369
booking_platform_retailint-tickets	22.4763	3.69526
booking_platform_ticketmaster	9.86964	5.58285
booking_platform_ticketsolve	3.62451	-0.362483
booking_platform_whelanslive	-7.22513	-1.03145
venue_type_art_gallery	-6.96381	-0.0840867
venue_type_event_venue	1.16642	1.20232
venue_type_hotel	20.4707	2.8447
venue_type_local_government_office	-24.3111	-1.65737
venue_type_museum	9.54032	0.382082
venue_type_night_club	-1.14438	-2.18399
venue_type_rare	1.15169	-0.926071
venue_type_store	7.86319	5.38857
venue_type_tourist_attraction	-1.34673	-0.630279



The plot compares the coefficients from an Ordinary Least Squares (OLS) model and an ElasticNet model, showing the impact of regularization. The OLS model (in blue) has large coefficients for many features, indicating sensitivity to feature scale and multicollinearity. In contrast, the ElasticNet model (in orange) reduces or eliminates many coefficients by setting them closer to zero, retaining only the most important features.



The ElasticNet model has retained features reflecting different aspects of the events that are important for predicting the target price variable. Key groups include booking platforms, venue types, county, distance from Dublin, recurrence, seasonality and venue rating.

To incorporate feature importance from ElasticNet into other models, the feature matrix was transformed using weights derived from ElasticNet coefficients. The absolute values of these coefficients served as importance weights, with larger coefficients indicating greater influence. These weights were normalized and applied to each feature, allowing key predictors to have a stronger impact in the transformed data. This approach enables other models, such as linear regression or decision trees, to benefit from ElasticNet's feature selection and regularization insights, resulting in more robust and interpretable models focused on the most impactful features.

Decision Tree Regressor

Decision Trees are interpretable machine learning models used for both classification and regression tasks. They split data into subsets based on feature values, forming a tree structure where each path represents a decision rule. In classification, common splitting criteria include Gini impurity and Information Gain, which measure the purity of the resulting subsets. In regression, the splitting criterion typically involves minimizing the variance or Mean Squared Error (MSE) within the subsets, aiming to reduce the variability of the target variable. While Decision Trees can capture non-linear relationships and handle both numerical and categorical data, they are prone to overfitting on complex datasets. Techniques such as pruning or ensemble methods are often employed to improve generalization.

A depth of 20 indicates a complex tree with numerous levels, suggesting that the model may be capturing intricate patterns in the training data. However, deeper trees are often more prone to overfitting.

```
Score metric: r2
CV scores mean: 0.418 (+/- 0.378)
CV scores std: 0.189
CV folds score: ['0.532', '0.104', '0.572', '0.297', '0.588']

Training Set Performance:
Mean Absolute Error (MAE): 1.640
Mean Squared Error (MSE): 16.578
Root Mean Squared Error (RMSE): 4.072
R2 Score: 0.932

Test Set Performance:
Mean Absolute Error (MAE): 5.826
Mean Squared Error (MSE): 105.497
Root Mean Squared Error (RMSE): 10.271
R2 Score: 0.309
```

The metrics confirm overfitting due to the model's depth. With a depth of 20, the Decision Tree performs well on the training set but experiences a significant drop on the test set, indicating poor generalization. High variability across folds, similar to what was observed in the linear regression model, persists, suggesting instability. To address this, we will test the weighted feature matrix obtained from ElasticNet to potentially improve stability and emphasize the most impactful features.

```
Score metric: r2
CV scores mean: 0.381 (+/- 0.197)
CV scores std: 0.099
CV folds score: ['0.507', '0.236', '0.328', '0.363', '0.474']

Training Set Performance:
Mean Absolute Error (MAE): 1.662
Mean Squared Error (MSE): 16.868
Root Mean Squared Error (RMSE): 4.107
R2 Score: 0.931

Test Set Performance:
Mean Absolute Error (MAE): 6.212
Mean Squared Error (MSE): 119.700
Root Mean Squared Error (RMSE): 10.941
R2 Score: 0.216
```

Applying ElasticNet weighting improved the Decision Tree model's stability by reducing variability across cross-validation folds (standard deviation decreased from 0.18 to 0.10).

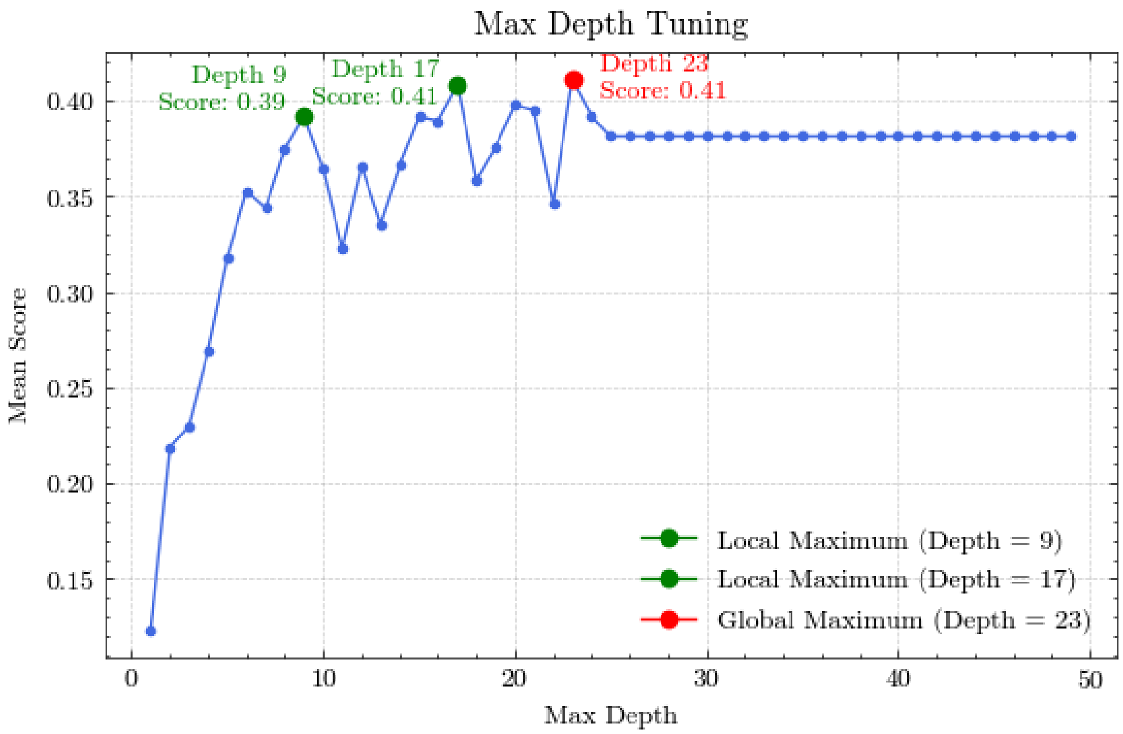
However, this came with a trade-off, as test performance slightly declined, indicating reduced generalization.

The max_depth parameter is being tuned to achieve an optimal balance between model complexity and performance. If the tree depth is too shallow, underfitting may occur. Conversely, an excessively deep tree may result in overfitting, where the model learns noise and specific patterns that don’t generalize well to new data.

```
Hyperparameter = { ‘max_depth’: range(1, 50) }
Fitting 5 folds for each of 49 candidates, totalling 245 fits
Score metric: None
CV scores mean: 0.411 (+/- 0.166)
CV scores std: 0.083
CV folds score: ['0.503', '0.269', '0.376', '0.436', '0.474']
Best Parameters:
{'max_depth': 23}

Training Set Performance:
Mean Absolute Error (MAE): 1.662
Mean Squared Error (MSE): 16.868
Root Mean Squared Error (RMSE): 4.107
R2 Score: 0.931

Test Set Performance:
Mean Absolute Error (MAE): 6.212
Mean Squared Error (MSE): 119.700
Root Mean Squared Error (RMSE): 10.941
R2 Score: 0.216
```



While the grid search identifies a max_depth of 23 as the global maximum for the mean cross-validation R2 score (0.41), the plot reveals that performance gains beyond a depth of 9 are minimal. Up to max_depth = 9, the mean score rises steadily, reaching a local maximum, after which additional depth yields only marginal improvements, and the curve plateaus. This suggests that increasing complexity past this local maximum adds little predictive power. Choosing max_depth = 9 captures similar performance to the global maximum while maintaining a simpler, more interpretable model. Another option would be max_depth = 17, where the performance is identical to max_depth = 23.

Before evaluating the model, max_depth and the following other parameters will be tuned:

- max_depth: [9, 17, 23] – Sets the maximum tree depth based on Max Depth Tuning.
- min_samples_split: [2, 5, 10] – Controls the minimum number of samples required to split a node. Higher values prevent splits based on small sample sizes, which helps reduce overfitting.
- min_samples_leaf: [1, 5, 10] – Determines the minimum samples required for a leaf node. Larger values create fewer, larger leaves, reducing overfitting by avoiding small leaves that may fit noise.
- max_features: [‘auto’, ‘sqrt’, ‘log2’, None] – Limits the number of features considered at each split, introducing a regularization effect. Tuning this parameter helps find a feature subset size that improves generalization.
- max_leaf_nodes: [10, 20, 50] – Restricts the maximum number of terminal nodes, thus controlling tree complexity. Lower values simplify the model, while higher values allow more detail.
- criterion: [‘squared_error’, ‘absolute_error’] – Specifies the function for evaluating splits. squared_error is sensitive to outliers, while absolute_error minimizes average errors and is less affected by outliers.

```
Fitting 5 folds for each of 864 candidates, totalling 4320 fits
Score metric: None
CV scores mean: 0.452 (+/- 0.260)
CV scores std: 0.130
CV folds score: ['0.547', '0.534', '0.211', '0.420', '0.549']
Best Parameters:
{'criterion': 'absolute_error',
 'max_depth': 17,
 'max_features': None,
 'max_leaf_nodes': 50,
 'min_samples_leaf': 1,
 'min_samples_split': 2}

Training Set Performance:
Mean Absolute Error (MAE): 4.568
Mean Squared Error (MSE): 63.989
Root Mean Squared Error (RMSE): 7.999
R2 Score: 0.738
```

```
Test Set Performance:
Mean Absolute Error (MAE): 6.487
Mean Squared Error (MSE): 97.684
Root Mean Squared Error (RMSE): 9.884
R2 Score: 0.360
```

The Decision Tree model shows poor generalization, with a low test R2 score (0.36) despite further tuning. This gap suggests the model struggles to capture complex patterns and is sensitive to data variance. Decision Trees inherently have high variance and may perform poorly with complex or noisy data. Therefore, further tuning is unlikely to help, and moving to ensemble methods, such as Random Forests or Gradient Boosting, can improve stability, reduce variance, and enhance generalization by combining multiple trees.

Random Forest Regressor

Ensemble methods combine multiple models to enhance predictive performance and robustness. By aggregating the outputs of several weak learners (often Decision Trees), ensemble techniques reduce the risk of overfitting and improve generalization to unseen data. Two popular ensemble approaches include Random Forest and Gradient Boosting:

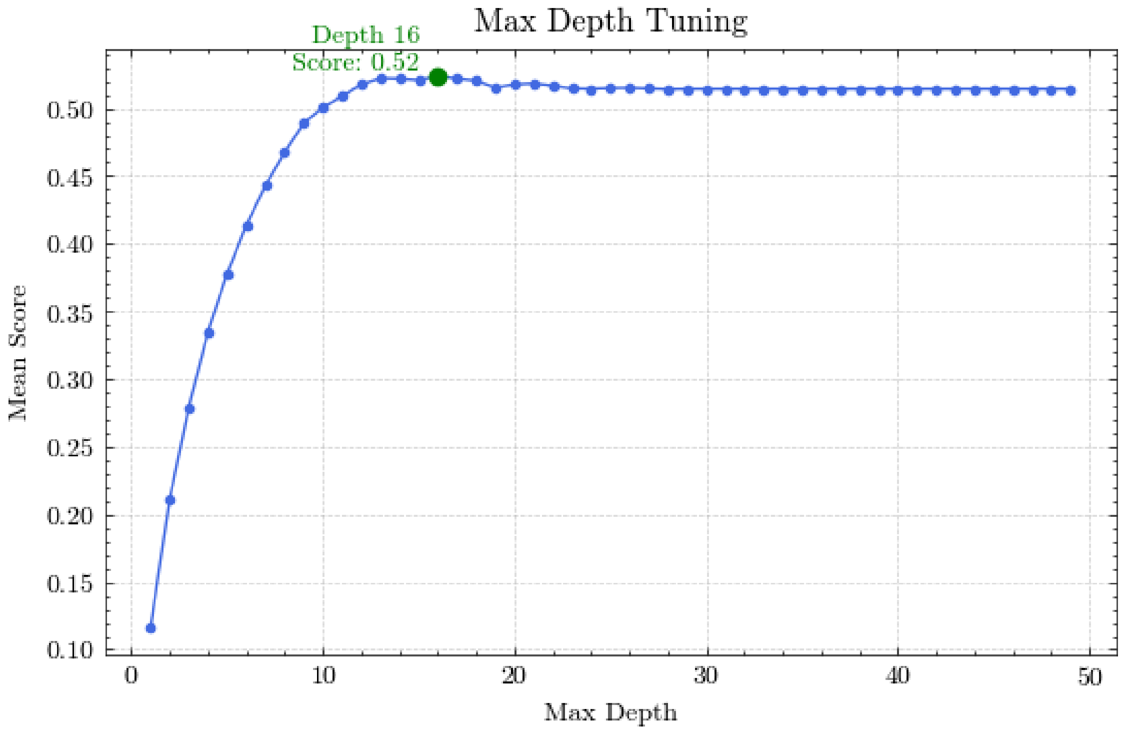
- Random Forest: Constructs multiple Decision Trees and averages their predictions, reducing variance and making the model more robust to noise and overfitting.
- Gradient Boosting: Sequentially builds trees, with each tree correcting the errors of the previous one, resulting in a strong model capable of capturing complex patterns effectively.

Here we will explore Random Forest only. This is an ensemble learning algorithm that employs bagging (bootstrap aggregating) to construct multiple decision trees on random subsets of data, then averages their predictions to improve accuracy and stability. In bagging, each tree is built independently on a unique data subset sampled with replacement, introducing randomness that helps reduce overfitting and enhance generalization. Additionally, Random Forest selects a random subset of features at each split, promoting model diversity and robustness to noise. This approach effectively captures complex, non-linear relationships and provides feature importance scores to identify key predictors.

The Random Forest model with default parameters shows improved generalization compared to the Decision Tree, with a higher test R2 score (0.43 vs. 0.36) and more balanced error metrics. While it still overfits somewhat, indicated by a gap between training and test performance, the cross-validation standard deviation (0.09) suggests stability. Random Forest captures more general patterns in the data, but further tuning or trying other ensemble methods could enhance generalization further.

```
Hyperparameter = { ‘max_depth’: range(1, 50), }
```

```
Fitting 5 folds for each of 49 candidates, totalling 245 fits
Score metric: None
CV scores mean: 0.524 (+/- 0.172)
CV scores std: 0.086
CV folds score: ['0.590', '0.441', '0.409', '0.546', '0.634']
Best Parameters:
{'max_depth': 16}
```



```
Training Set Performance:
Mean Absolute Error (MAE): 3.287
Mean Squared Error (MSE): 27.460
Root Mean Squared Error (RMSE): 5.240
R2 Score: 0.888

Test Set Performance:
Mean Absolute Error (MAE): 6.002
Mean Squared Error (MSE): 86.010
Root Mean Squared Error (RMSE): 9.274
R2 Score: 0.437
```

A practical approach to choose hyperparameters is to consult the library documentation, which highlights key hyperparameters such as n_estimators (the number of trees) and max_features (the number of features considered for each split). Although research papers offer theoretical insights into optimal settings, a more practical strategy is to test a broad range of values and apply methods like Randomized Search to empirically determine the best parameters for the specific dataset.

Randomized Search is a hyperparameter tuning technique that samples a specified number of random combinations from a defined parameter space, rather than exhaustively testing all possibilities as in Grid Search. This approach is faster and more efficient, particularly for complex models with large parameter spaces, as it avoids the computational burden of evaluating every combination. Key settings include n_iter, which controls the number of combinations tried.

```
Hyperparameter = = { ‘n_estimators’: [100, 200, 300], ‘max_depth’: [10, 12, 14,
16], ‘min_samples_split’: [4, 8, 10, 12], ‘min_samples_leaf’: [2, 4, 6],
‘max_features’: [‘sqrt’, ‘log2’, 0.3, 0.6], ‘bootstrap’: [True], ‘criterion’:
[‘squared_error’] }
Fitting 5 folds for each of 500 candidates, totalling 2500 fits
Score metric: None
CV scores mean: 0.496 (+/- 0.155)
CV scores std: 0.078
CV folds score: ['0.561', '0.459', '0.384', '0.475', '0.603']
Best Parameters:
{'bootstrap': True,
 'criterion': 'squared_error',
 'max_depth': 16,
 'max_features': 0.6,
 'min_samples_leaf': 2,
 'min_samples_split': 4,
 'n_estimators': 200}

Training Set Performance:
Mean Absolute Error (MAE): 4.193
Mean Squared Error (MSE): 45.071
Root Mean Squared Error (RMSE): 6.713
R2 Score: 0.816

Test Set Performance:
Mean Absolute Error (MAE): 6.220
Mean Squared Error (MSE): 82.583
Root Mean Squared Error (RMSE): 9.087
R2 Score: 0.459
```

The Random Forest model achieved an improved R2 score of 0.2 on test data but still shows signs of overfitting, indicating a need for further adjustments to enhance its robustness on unseen data. Achieving optimal results with machine learning often requires extensive parameter tuning, which can be both complex and computationally intensive; thus, this has been designated as future work. Additionally, exploring alternative models, such as XGBoost, may further improve performance. In this study, XGBoost was included primarily as an example to illustrate its potential, with no in-depth analysis or optimization conducted.

XGBoost Regressor

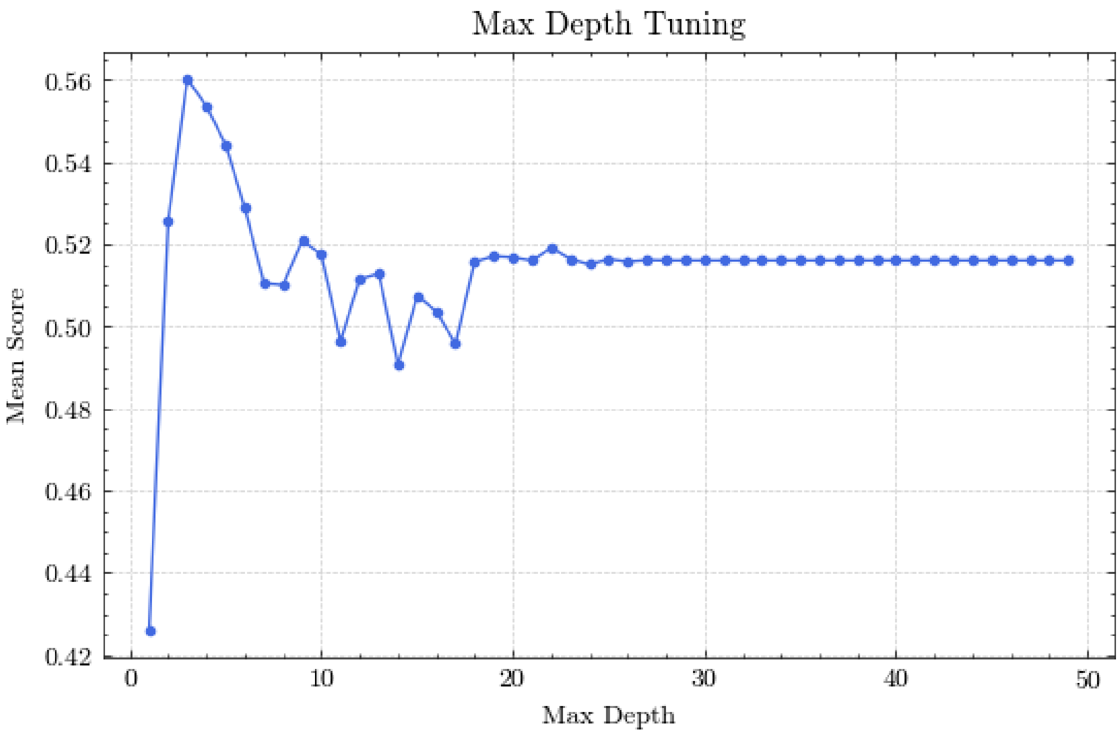
XGBoost, short for “Extreme Gradient Boosting”, is a powerful machine learning algorithm based on the gradient boosting framework. Unlike Random Forest, which builds multiple decision trees independently and aggregates their results, XGBoost builds trees sequentially, with each tree aiming to correct the errors of the previous one. This iterative approach allows XGBoost to learn complex patterns by minimizing the residual errors at each step, often leading to higher accuracy. XGBoost also includes features like regularization, handling of missing data, and optimized memory usage, making it both faster and more robust on many datasets compared to Random Forest. However, XGBoost can be more computationally intensive and may require more careful tuning.

```
Score metric: r2
CV scores mean: 0.529 (+/- 0.114)
CV scores std: 0.057
CV folds score: ['0.555', '0.508', '0.444', '0.520', '0.617']

Training Set Performance:
Mean Absolute Error (MAE): 1.925
Mean Squared Error (MSE): 17.218
Root Mean Squared Error (RMSE): 4.149
R2 Score: 0.930

Test Set Performance:
Mean Absolute Error (MAE): 5.375
Mean Squared Error (MSE): 82.621
Root Mean Squared Error (RMSE): 9.090
R2 Score: 0.459
```

```
Hyperparameter = { ‘max_depth’: range(1, 50), }
Fitting 5 folds for each of 49 candidates, totalling 245 fits
Score metric: None
CV scores mean: 0.560 (+/- 0.145)
CV scores std: 0.072
CV folds score: ['0.574', '0.565', '0.426', '0.595', '0.642']
Best Parameters:
{'max_depth': 3}
```



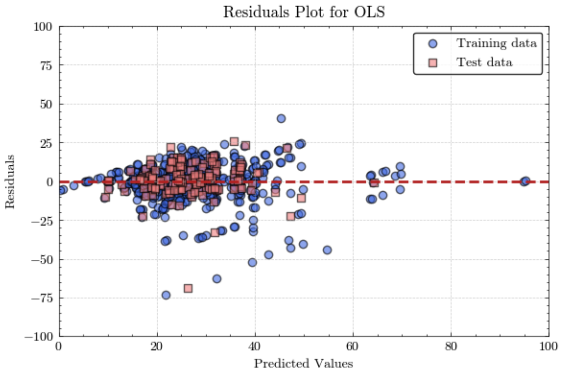
Hyperparameter = { ‘learning_rate’: [0.01, 0.05, 0.1], ‘n_estimators’: [100, 200, 300], ‘max_depth’: [3, 5, 7], ‘subsample’: [0.7, 0.8, 0.9, 1.0], ‘colsample_bytree’: [0.7, 0.8, 0.9, 1.0], ‘reg_alpha’: [0, 0.01, 0.1, 1], ‘reg_lambda’: [1, 0.1, 0.5, 1.5] }

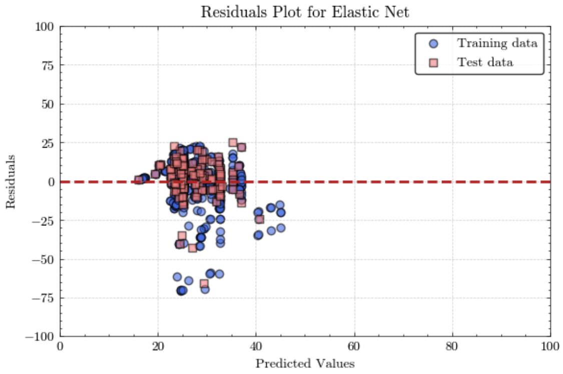
Training Set Performance:
Mean Absolute Error (MAE): 3.571
Mean Squared Error (MSE): 28.692
Root Mean Squared Error (RMSE): 5.356
R2 Score: 0.883

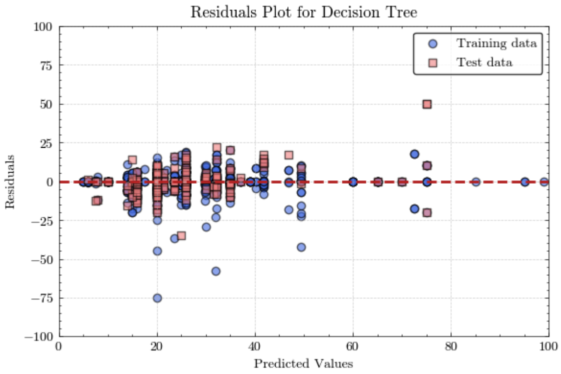
Test Set Performance:
Mean Absolute Error (MAE): 5.477
Mean Squared Error (MSE): 71.976
Root Mean Squared Error (RMSE): 8.484
R2 Score: 0.529

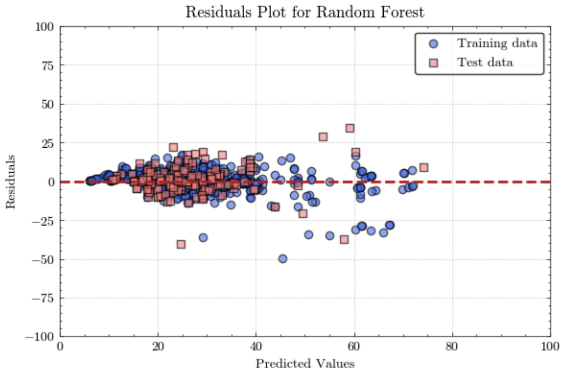
Models Comparison

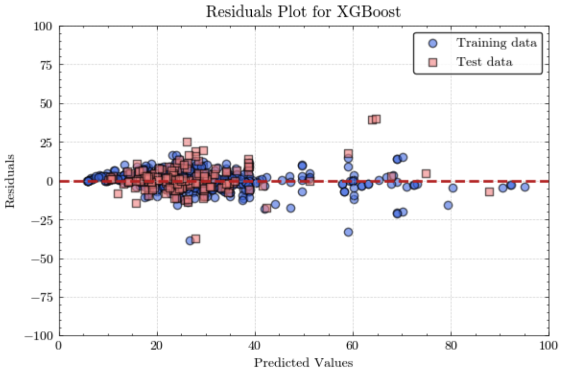
Residual plots are a graphical tool used to evaluate the performance and reliability of regression models by displaying the differences between predicted and actual values, known as residuals. These plots assist in diagnosing issues such as non-linearity, unequal error variances, and outliers that may impact model accuracy.

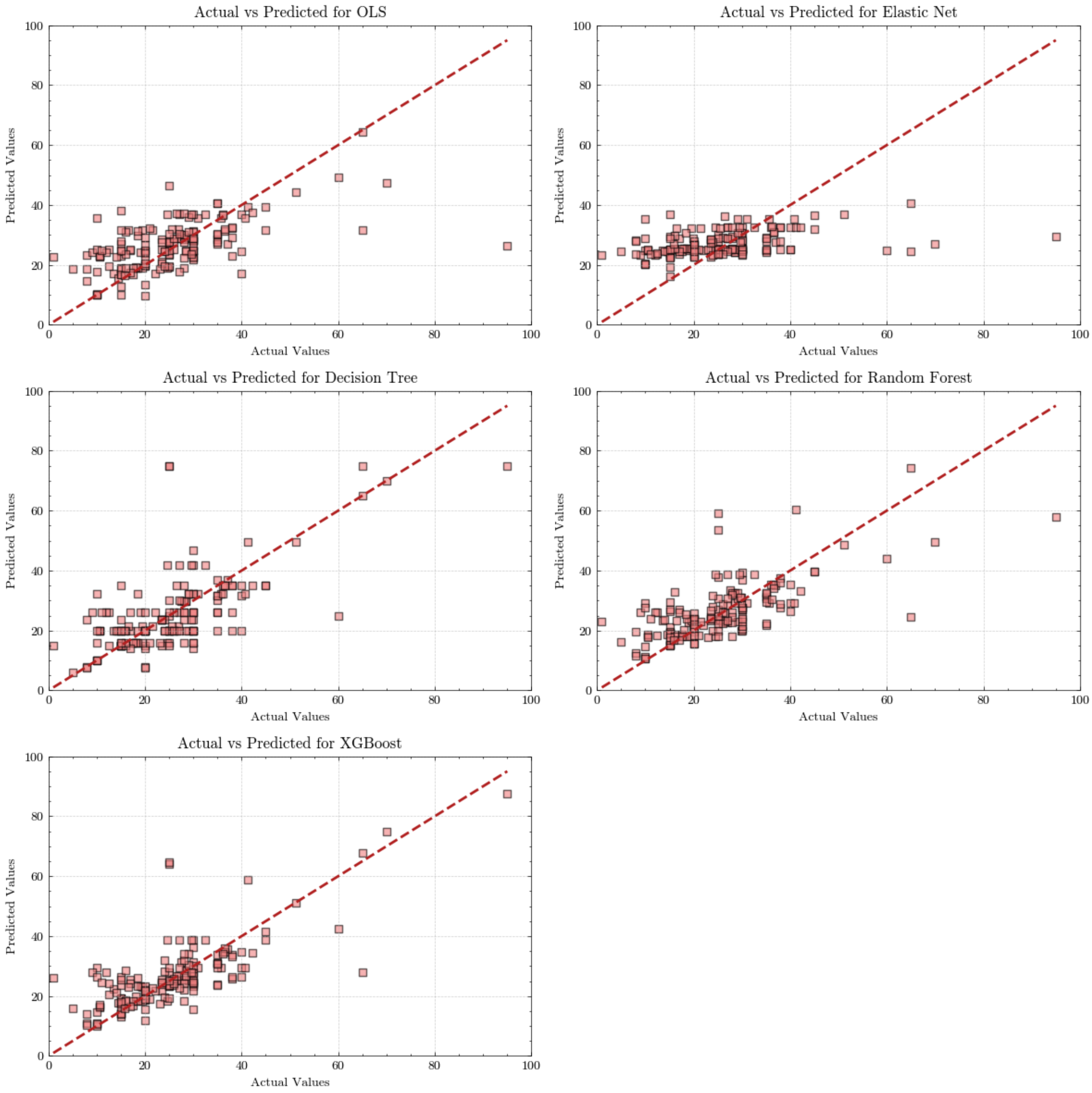

Residuals Plot for OLS


Residuals Plot for Elastic Net


Residuals Plot for Decision Tree


Residuals Plot for Random Forest


Residuals Plot for XGBoost



XGBoost and Random Forest exhibit the best performance, with tightly clustered residuals, while OLS and Elastic Net display more spread-out residuals, particularly around non-linear patterns. Decision Trees show signs of overfitting, but ensemble methods like Random Forest and XGBoost improve generalization. XGBoost appears to be the optimal choice for capturing complex relationships, as it demonstrates lower residual variance.

An actual vs. predicted plot offers an intuitive visualization of a model's predictive accuracy by displaying how well its predictions align with the true values. Each point represents a prediction; in a perfect model, all points would lie along the diagonal dashed line, indicating that predicted values equal actual values.

- OLS and Elastic Net: These models show moderate clustering around the line but exhibit noticeable scatter, indicating difficulty in capturing certain nuances in the data.
- Decision Tree: Displays closer clustering near the line for lower actual values, but shows some spread at higher values. This suggests that, while it captures some patterns, it may overfit on certain values.

- Random Forest and XGBoost: These models demonstrate tighter clustering near the line, particularly XGBoost, suggesting that these ensemble methods are better at capturing complex patterns and provide more accurate predictions.

	Model	MAE (Train)	MSE (Train)	RMSE (Train)	R2 (Train)	MAE (Test)	MSE (Test)	RMSE (Test)	R2 (Test)
0	OLS	7.85	129.39	11.38	0.47	6.79	102.95	10.15	0.33
1	Elastic Net	9.5	194.92	13.96	0.2	8.15	132.97	11.53	0.13
2	Decision Tree	4.57	63.99	8	0.74	6.49	97.68	9.88	0.36
3	Random Forest	4.19	45.07	6.71	0.82	6.22	82.58	9.09	0.46
4	XGBoost	3.57	28.69	5.36	0.88	5.48	71.98	8.48	0.53

The table displays summarize the performance of the models across various regression metrics, highlighting differences in how well each model generalizes from the training to the test set:

- OLS (Ordinary Least Squares): This model exhibits a relatively high MAE and MSE on the training set, indicating lower accuracy compared to other models. Its R^2 score is low for both training and test sets, with an R^2 of 0.33 on the test set, suggesting poor generalization. The high RMSE on both sets indicates that OLS struggles with larger errors, making it less suitable for capturing complex patterns in the data. However, this model was used solely as a baseline.
- Elastic Net: This regularized linear model exhibits the highest errors and lowest R^2 scores among the models on both the training and test sets. However, this outcome is expected, as Elastic Net was primarily used for feature selection and weighting rather than for predictive accuracy.
- Decision Tree: This model exhibits lower MAE, MSE, and RMSE on the training set but higher values on the test set, indicating potential overfitting. The R2 score on the training set is high (0.74), yet it drops significantly to 0.36 on the test set. This suggests that the model is capturing noise in the training data, which adversely impacts its generalization to new data.
- Random Forest: This ensemble method significantly reduces error metrics on the test set compared to previous models, achieving an RMSE of 9.09 and an R2 of 0.46. The Random Forest model strikes a balance between capturing patterns and avoiding overfitting, as shown by the smaller gap in R2 between the training set (0.82) and test set (0.46). Overall, this model demonstrates improved performance and generalization.
- XGBoost: Among all models, XGBoost achieves the lowest MAE, MSE, and RMSE on the test set, indicating the best predictive accuracy. Its R2 score of 0.53 on the test set is also the highest, suggesting that it explains more variability in the data than other models. The strong performance across all metrics makes XGBoost the most robust model in this comparison.

Appendix L. Inferential Statistic

Inferential Statistics in data science allows analysts to make predictions about a population based on sample data, using probability distributions to identify trends and patterns. These distributions are classified as Discrete (finite outcomes, such as the Bernoulli and Binomial distributions for binary outcomes or counts of success) and continuous (infinite possible values within a range, such as the Normal distribution for natural data variations or the Exponential distribution for time intervals). Key distributions include the Bernoulli, Uniform, Binomial, Normal, Poisson, and Exponential distributions, each tailored to specific data scenarios and essential for making accurate inferences.

Binomial Distribution

The **Binomial Distribution** describes the number of successes in a fixed number of independent trials, each with the same probability of success. The probability of getting exactly k successes in n independent trials is given by the binomial probability formula:

$$P(X = k) = \binom{n}{k} \cdot p^k \cdot (1 - p)^{n-k}$$

Where:

- n = number of trials
- k = number of successes
- p = probability of success on a single trial
- $(1 - p)$ = probability of failure

Common Statistics	Formula
Mean	np
Mode	$p(n + 1) - 1 \leq x \leq p(n + 1)$
Range	0 to n
Standard Deviation	$\sqrt{np(1 - p)}$

Criteria for a Binomial variable: - Binary outcomes: Only two unique values. - Natural success/failure interpretation: One value can be assigned as success, the other as failure. - Events Independence: Each event should be independent of others. - Consistent probability: The success probability should be stable across trials.

For our dataset, the is_free column has been selected as an ideal candidate for binomial modeling because it meets the fundamental requirements for a binomial setup:

- Binary outcomes: The is_free column is binary, containing only two unique values, 0 and 1. This structure allows us to define clear outcomes: 1 can be treated as a “success” (indicating a free event), while 0 represents a “failure” (indicating a paid event).
- Natural success/failure interpretation: For binomial analysis, it’s essential to have a column that naturally defines “success” versus “failure.” In this case, a free event can be viewed as a success, while a paid event represents a failure.
- Events Independence: To ensure the events in the dataset are independent, rows representing multiple dates for the same event must be removed. If an event

occurs on different dates, treating each occurrence as an independent trial would violate the independence assumption, as these repeated events could potentially influence each other.

- Consistent probability: To verify whether the probability of an event being free remains consistent across different segments of the dataset, such as across seasons, we can calculate the proportion of free events for each season. If the probability of a free event varies noticeably by season, it suggests that a seasonal analysis may be more appropriate, as the binomial model assumes a stable probability of success for all trials.

From EDA section we know that, the proportions of free events by season are as follows:

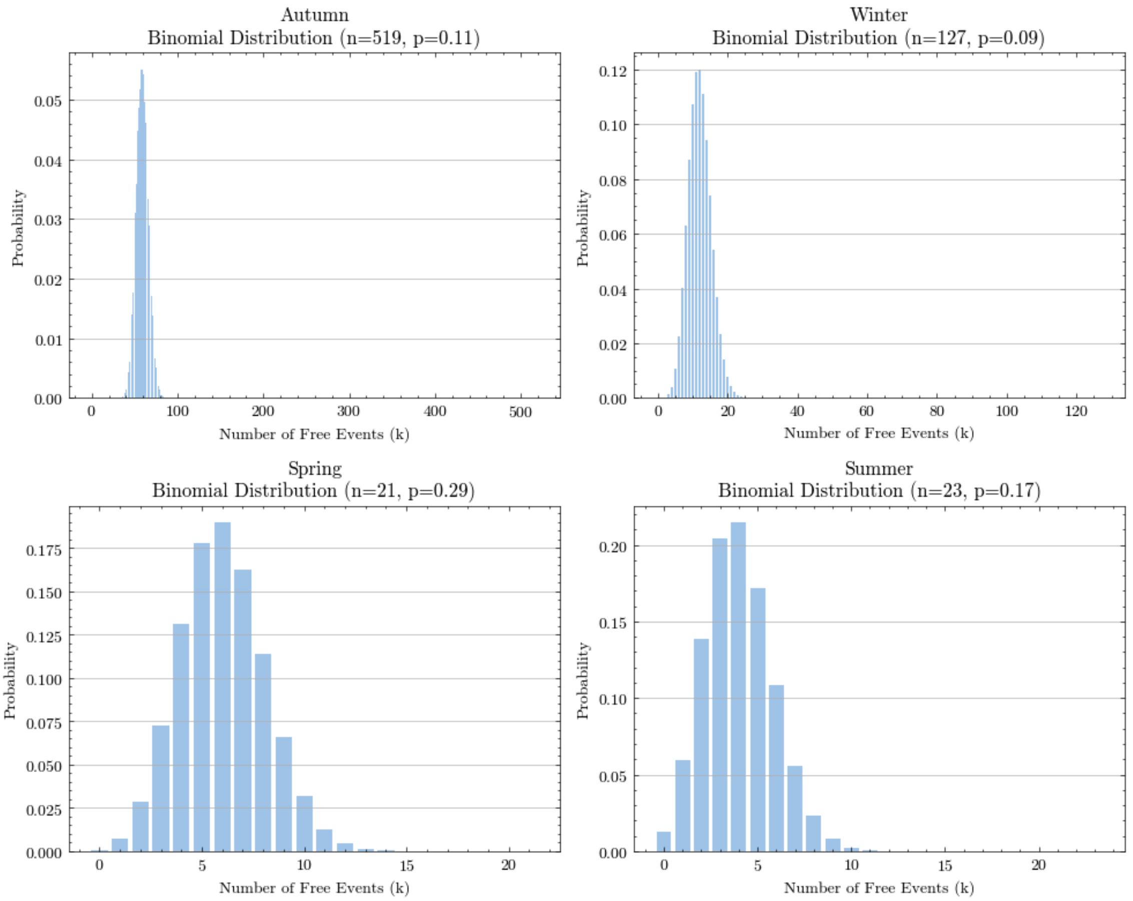
- Autumn: 11.3%
- Spring: 28.6%
- Summer: 16.7%
- Winter: 9.4%

These values indicate significant variation in the probability of free events across seasons, with spring displaying the highest proportion of free events at nearly 28.6%, while autumn and winter show much lower proportions, around 9.4%. This variability suggests that analysing the dataset by season could yield more accurate results, as each season exhibits a different likelihood of free events.

Binomial PMF

The Probability Mass Function (PMF) is used with discrete random variables and provides the probability of a specific outcome or value. For a discrete random variable X, the PMF is denoted as P(X = k), giving the probability that X takes on a specific value k.

In our analysis, we focus on the number of free events for each season separately: - n: Total number of events in the specific season - k: The specific number of free events we are interested in - p: Probability of an event being free in that season.



Autumn
Binomial Distribution (n=519, p=0.11)

Winter
Binomial Distribution (n=127, p=0.09)

Spring
Binomial Distribution (n=21, p=0.29)

Summer
Binomial Distribution (n=23, p=0.17)

The binomial distributions for each season reveal distinct patterns in the likelihood of free events. In autumn and winter, with high event counts but low probabilities of free events, the distributions are tightly concentrated near zero, indicating that free events are rare and the binomial model captures this well. Spring shows the highest probability of free events, resulting in a broader, more symmetric distribution that reflects greater accessibility, while summer displays a moderate spread, indicating some likelihood of free events but less than spring.

Binomial CDF

The Cumulative Density Function (CDF) applies to both discrete and continuous random variables and provides the cumulative probability up to a specific value. For a random variable X, the CDF, denoted as F(k) = P(X ≤ k), gives the probability that X is less than or equal to a certain value k.

Suppose we are analysing a specific period, such as spring, with 10 (n) scheduled events in total. We can use the binomial cumulative distribution function (CDF) to determine the probability of having a certain number of free events during this period.

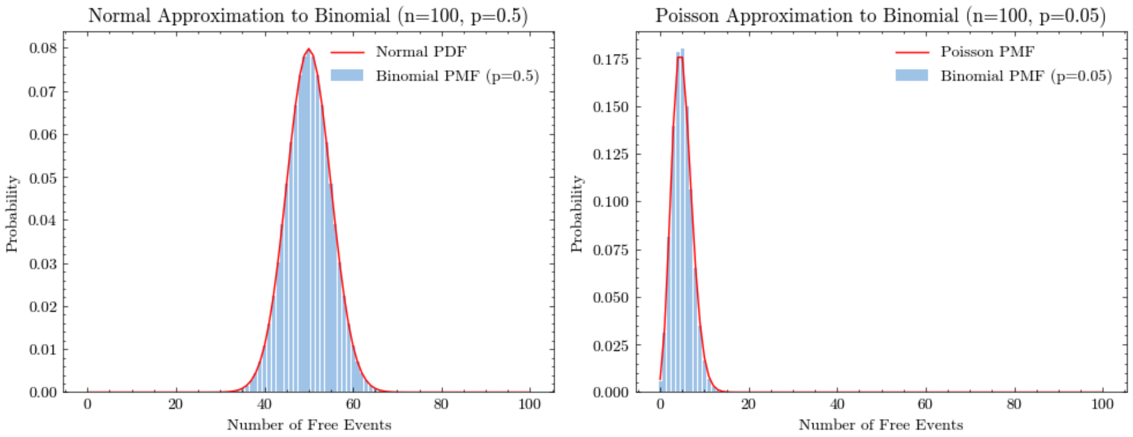
For example, to find the probability of having more than 3 (k) free events out of these 10 events, we calculate the probability of 3 or fewer free events using the CDF and subtract it from 1. This gives us the probability of having more than 3 free events. In this example, the probability of having more than 3 free events out of 10 scheduled events is approximately 0.312, or 31.2%.

Probability in Spring of having more than 3 free events out of these 10 events = 0.31

Binomial Approximation

Approximating a binomial distribution with either a normal or Poisson distribution is often employed when the number of trials n becomes large, making exact probability calculations challenging and computationally intensive. These approximations enable simpler and faster probability estimation without the need to repeatedly compute the binomial formula.

- Normal Approximation: This approach is preferred when p is moderate (around 0.5), creating a balanced, bell-shaped binomial distribution that aligns well with the shape of the normal distribution. Because the normal distribution is symmetric and centred around its mean, a binomial distribution with a moderate p also becomes roughly symmetric, making the normal approximation an effective fit. The reason is related to the Central Limit Theorem in statistics and this phenomenon is also known as De Moivre — Laplace theorem.
- Poisson Approximation: The Poisson approximation is used when p is small, as the binomial distribution in such cases becomes highly skewed, concentrating probabilities at lower counts of successes (given that a small p implies fewer likely successes). The Poisson distribution, which models low-probability events across many trials and is inherently skewed, is a better fit than the symmetric normal distribution when p is small and n is large.



The mean and standard deviation of a binomial distribution with n trials and success probability p represent the expected number of successes and the variability around that expectation. The mean, $ = np $, provides the expected number of successes, while the standard deviation, $ = $, measures the spread of outcomes. These formulas are essential for normal approximation, as they allow us to construct a bell-shaped curve that approximates the binomial distribution.

The Poisson approximation is applied to the binomial distribution when p is small (indicating rare events) and n is large, with the product $\lambda = np$ held moderate. This approximation is effective because the Poisson distribution is specifically suited for modeling rare events, with its mean λ corresponding directly to the expected number of successes in the binomial distribution. Under these conditions, the skewed shape of the Poisson distribution closely matches that of the binomial distribution.

Poisson Distribution

The Poisson Distribution describes the probability of a given number of events occurring in a fixed interval of time or space when these events happen independently, with a constant mean rate. It is commonly used to model counts of rare or random events over a fixed period.

The probability of observing exactly k events in a fixed interval, given the average rate λ (lambda), is given by the Poisson probability formula:

$$P(X = k) = \frac{\lambda^k e^{-\lambda}}{k!}$$

Where:

- λ = the average rate of events (mean number of events in the interval)
- k = the number of events occurring within the interval
- e = Euler's number, approximately 2.71828.

Common Statistics	Formula
Mean	λ
Mode	$\lfloor\lambda\rfloor$ or $\lfloor\lambda\rfloor - 1$ (if λ is an integer)
Range	0 to ∞
Standard Deviation	$\sqrt{\lambda}$

Criteria for a Poisson Variable: - Discrete Values: An event either occurs or does not; partial occurrences are not possible. - Non-Overlapping Events: Multiple events do not happen simultaneously. - Events Independence: The occurrence of one event does not influence the occurrence of other events. - Constant Mean Rate: The average (mean) number of events remains constant over time.

In our dataset, the count of events over time has been identified as an ideal candidate for Poisson modeling because it meets the fundamental requirements for a Poisson setup:

- Discrete Values: The data consists of event counts within specific time intervals (e.g., daily or monthly). Events either occur or do not within each time period, making the data inherently discrete and suitable for Poisson analysis. For example, each day or month serves as an interval for counting free or paid events.
- Non-Overlapping Events: Poisson modeling requires that events do not occur simultaneously within an interval. In our dataset, each event is counted independently, with no overlap in classification.
- Events Independence: For accurate Poisson modeling, each interval (such as a day) should reflect individual events that do not influence one another. If the same event appears on multiple dates, it is essential to verify that each occurrence is treated independently to maintain the independence assumption, where one event does not affect the probability of another.
- Constant Mean Rate: Poisson modeling assumes a relatively stable average event rate (mean) over time. In our dataset, we can assess whether the mean number of events (such as free or paid events) remains consistent across different time intervals (e.g., daily or monthly).

To verify if the event counts over time fit a Poisson distribution, we first calculate the average event rate (mean) across different intervals, such as daily and monthly, to establish a baseline occurrence rate. Next, we examine the stability of these mean rates across intervals by comparing daily and monthly averages, ensuring they remain relatively consistent. Finally, we measure the variance in event counts and compare it to the mean for each interval. A variance close to the mean supports the Poisson assumption, as it indicates a stable rate of events without external fluctuations.

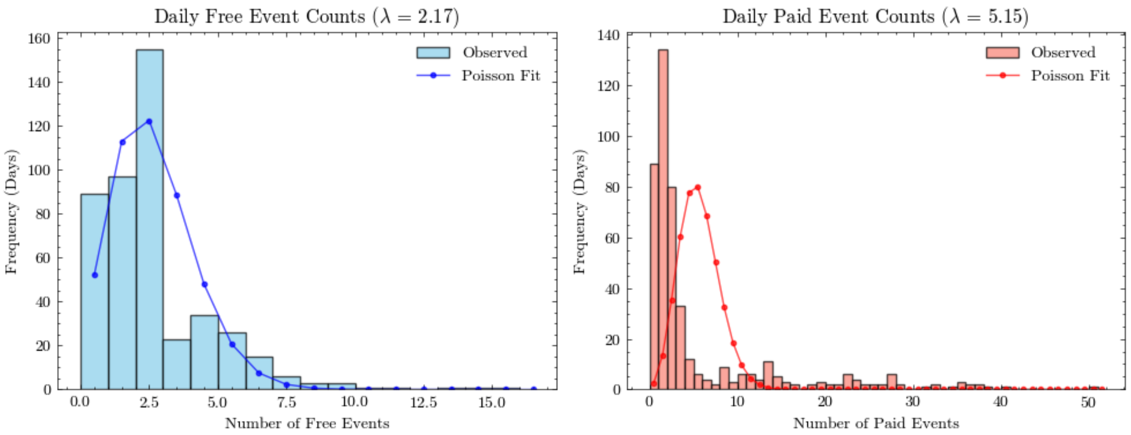
Interval	Mean Free Events	Variance Free Events	Variance-to-Mean Ratio Free	Mean Paid Events	Variance Paid Events	Variance-to-Mean Ratio Paid
Daily	2.16886	4.58021	2.11181	5.14912	72.3865	14.058
Monthly	47.0952	3543.39	75.2388	111.81	42824.8	383.015

Although the high variance-to-mean ratios generally suggest that our data may not perfectly follow a Poisson distribution, the daily free events exhibit a relatively lower ratio ~2.11 compared to other subsets. This value is closer to the Poisson ideal of 1, indicating that daily free events may approximate a Poisson distribution more closely than other event counts in the dataset.

Poisson PMF

In our analysis, we focus on the daily counts of free and paid events separately: - λ (lambda): The mean number of free/paid events per day - k: The specific count of free events we are interested in.

Note: the expected Poisson probabilities have been scaled to the number of observations, making them comparable to the observed frequencies in the histogram.



While the variance-to-mean ratio suggests some deviation from an ideal Poisson distribution, the visual alignment indicates that daily free event counts may still be well-approximated by a Poisson model. This could be attributed to minor fluctuations around the average rate that do not significantly disrupt the overall Poisson-like pattern. Therefore, the Poisson model may be useful for analysing daily free event counts, even if it is not a statistically perfect fit.

Poisson CDF

Using the Poisson cumulative distribution function (CDF) allows us to estimate the probability of observing specific counts of daily free events based on this approximation. For instance, with an average rate (λ) of 2.17 free events per day, we can use the Poisson CDF to calculate the probability of observing more than a target number of free events.

For example, to determine the likelihood of having more than 3 free events in a day, we calculate the cumulative probability of 3 or fewer free events and subtract it from 1. In this example, the probability of having more than 3 free events on a given day, given an average rate of 2.17, is approximately 0.174, or 17.4%.

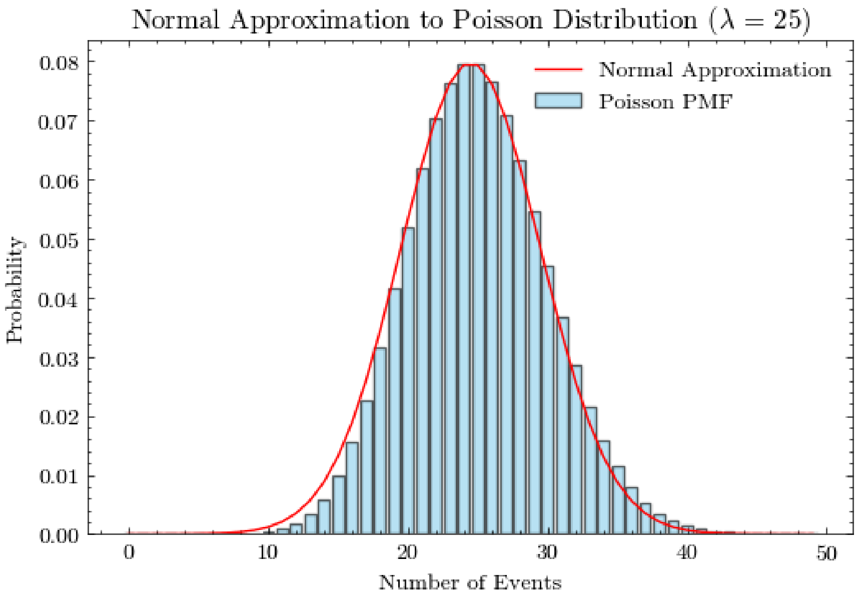
Despite minor statistical deviations, this method provides meaningful insights into daily free event counts, helping to anticipate the frequency of days with higher-than-average counts.

Likelihood of having more than 3 free events in a day = 0.17

Poisson Approximation

With larger values of λ, the Poisson distribution starts to look symmetric and resembles a normal distribution. This is because, with increasing λ, the discrete spikes of the Poisson distribution smooth out, concentrating around the mean and creating a bell-shaped curve. To use the normal approximation for a Poisson distribution, we apply this approach when the mean (λ) is large, generally when λ>20.

Since Poisson values are discrete and normal values are continuous, we use a continuity correction when calculating probabilities. The normal approximation can simplify calculations, but it slightly deviates from the actual Poisson probabilities.



To approximate a Poisson distribution with a normal distribution when the mean, denoted by λ, is large, we set the normal distribution's mean to λ and standard deviation to $\sqrt{\lambda}$. Since Poisson is discrete while normal is continuous, we apply a continuity correction by adjusting values by 0.5. F. For example, to find P(X > k), we use P(X > k + 0.5) in the normal distribution.

Normal Distribution

The Normal Distribution, also known as the Gaussian distribution, is a continuous probability distribution characterized by its bell-shaped curve. It is widely used in statistics to model natural phenomena and represents how data points are distributed around a mean. The distribution is defined by two parameters: the mean (μ) and the standard deviation (σ).

The probability density function (PDF) of a normal distribution is given by:

$$P(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Where:

- μ = mean of the distribution (center of the curve)
- σ = standard deviation (spread of the distribution)
- x = variable of interest
- e = Euler’s number, approximately 2.71828

Common Statistics	Formula
Mean	The location parameter μ.
Median	The location parameter μ.
Mode	The location parameter μ.
Range	$-\infty$ to ∞.
Standard Deviation	The scale parameter σ.

Criteria for a Normal Variable: - Continuous Values: The variable can take on any real value within a range. - Symmetric Data: The data should be distributed evenly around the mean. - Bell-Shaped Curve: The histogram of the data should resemble a bell curve. - Central Limit Theorem: For large sample sizes, the sampling distribution of the sample mean will be approximately normal, regardless of the shape of the population distribution.

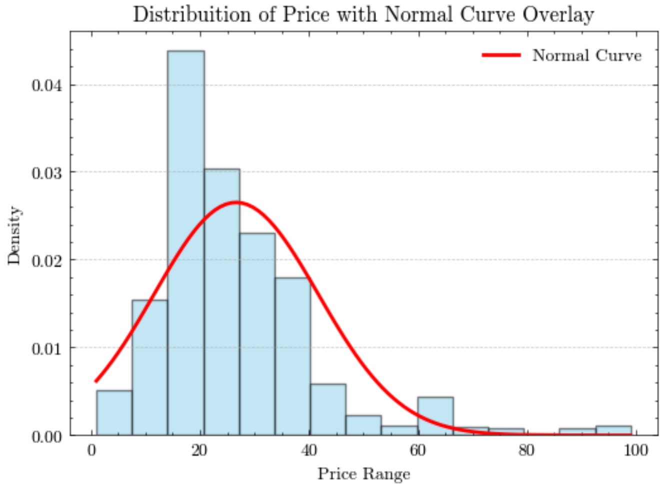
When selecting a variable for normal distribution analysis, it is essential to choose one that can benefit from standardization and probability calculations, such as through the use of z-scores. Price is an ideal candidate for this analysis because it is continuous and provides valuable insights into the likelihood of certain pricing ranges. For example, understanding how often prices exceed a specific threshold can inform business decisions or help set expectations for potential visitors regarding event affordability.

To ensure a meaningful and relevant analysis, as we did for the model data, we are filtering for paid events only. This approach prevents the skew that could be introduced by free events (price 0), which would artificially lower the average price and reduce the variability in the data. To be consistent with the model prediction, will use the same granularity in the model dataframe.

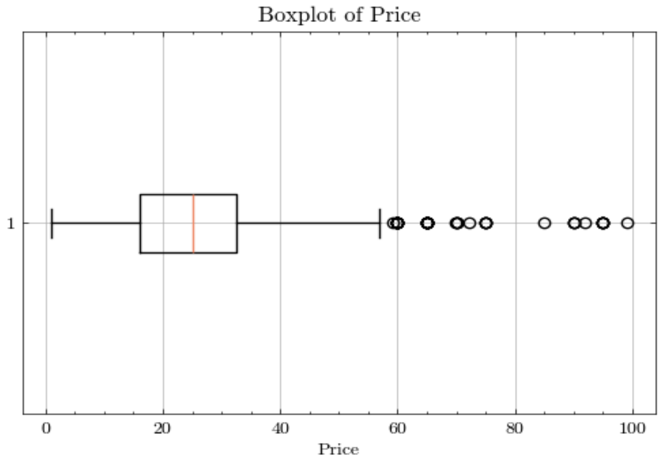
```
count    811.000000
mean      26.659174
```

```
std      15.057864
min       1.000000
25%      16.000000
50%      25.000000
75%      32.500000
max      99.000000
Name: price, dtype: float64
```

The broad spread, highlighted by the standard deviation and the range from 1 to 99 units, suggests a right-skewed distribution with some extreme values influencing the overall variability.

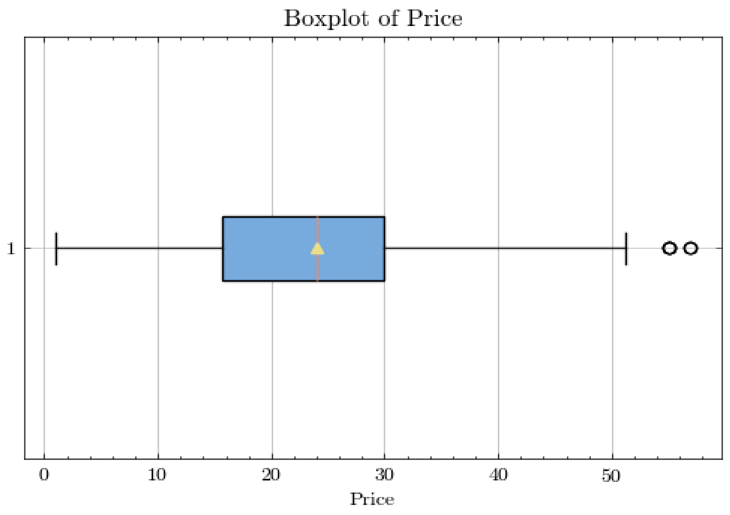


The histogram confirms that, although the data spans a wide range, it deviates from a perfect normal distribution. The curve highlights that the actual distribution is right-skewed, with a higher density of lower prices and a tail extending towards higher prices, reinforcing the earlier observation of non-normality.

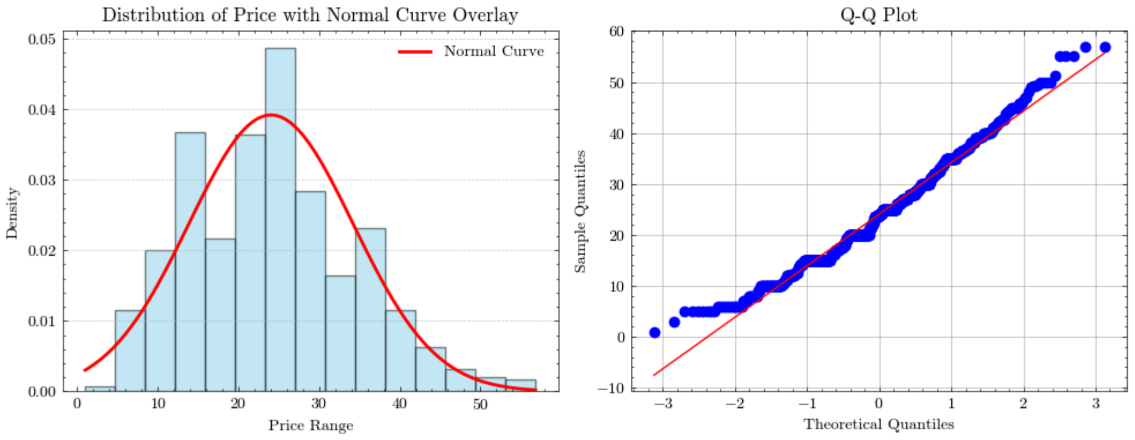


The boxplot provides a clear visualization of the distribution and highlights potential outliers, confirming the presence of extreme values at the higher end of the distribution.

Temporarily removing these outliers for further analysis can help assess the underlying distribution more accurately and determine whether it approaches normality without the influence of these extreme points.



The boxplot of price without outliers shows a clearer, more symmetric distribution, focusing on typical event prices. The median is now approximately equal to the mean, suggesting a more balanced distribution.



The combined plots demonstrate that, after removing outliers, the price distribution more closely approximates normality. The histogram aligns well with the normal curve, and the Q-Q plot shows that most data points follow the reference line closely, with only minor deviations at the tails. This suggests that the data is sufficiently normal for conducting normality-based statistical tests.

Normality Test

The Shapiro-Wilk test is a statistical test used to assess the normality of a dataset. It evaluates the null hypothesis that the data is drawn from a normal distribution by calculating a statistic that measures how well the data points align with a normal distribution:

- A high p-value (e.g., > 0.05) indicates that the data does not significantly deviate from normality, so the null hypothesis of normality is not rejected.
- A low p-value (e.g., ≤ 0.05) suggests that the data deviates from a normal distribution, leading to the rejection of the null hypothesis.

```
Statistic: 0.980834224563862, p-value: 1.764932678436155e-08 Sample does not
appear to be normally distributed.
```

The Shapiro-Wilk test result for the price variable (without outliers) gives a test statistic of 0.98 and a p-value of approximately 1.76e-08. Despite improvements from outlier removal, the very low p-value < 0.05) indicates that we reject the null hypothesis of normality, suggesting that the price distribution still deviates from a perfectly normal distribution.

To attempt transforming the price variable and potentially achieve a more normal distribution, we will apply a Box-Cox transformation, which can often handle skewed data more effectively than a simple log transformation, and re-evaluate the distribution.

```
Statistic: 0.9940596135778376, p-value: 0.004096870907762449
Sample does not appear to be normally distributed.
```

The Box-Cox transformation has significantly improved the normality of the price distribution, with a higher test statistic (0.994) and a p-value of 0.004, which, while still indicating a slight deviation from perfect normality, suggests a much better fit.

The Shapiro-Wilk test is particularly powerful for detecting deviations from normality in smaller sample sizes. However, it may become overly sensitive for very large datasets, where minor deviations from normality might result in a low p-value even if the data is approximately normal for practical purposes.

Z-Score

A **z-score** is a statistical measurement that describes how many standard deviations a data point is from the mean of a dataset. It is calculated using the formula:

$$Z = \frac{X - \mu}{\sigma}$$

where:

- X is the data point,
- μ is the mean
- σ is the standard deviation.

Z-scores are useful for determining the relative position of a data point within a distribution and are commonly used to find probabilities.

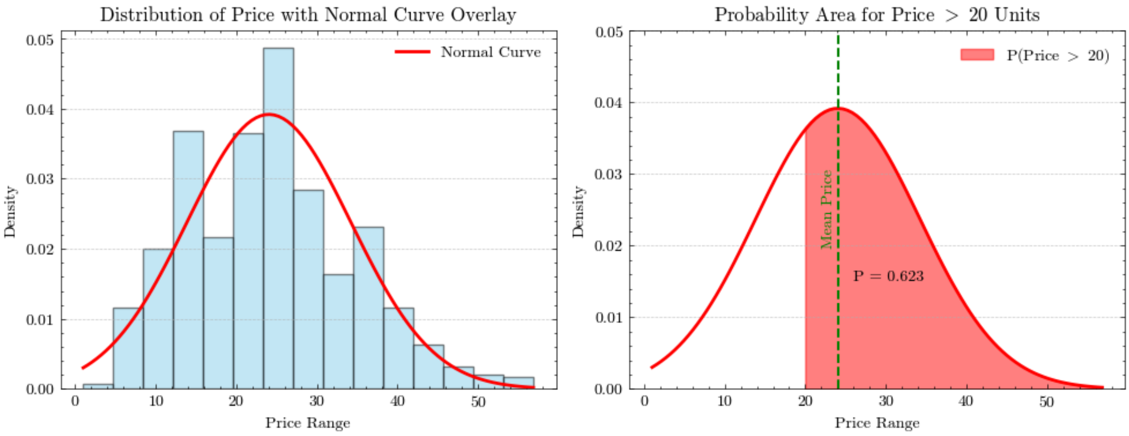
Using z-scores assumes a normal distribution, so if the data is not perfectly normal, the results may be less precise. However, if the data is approximately normal or has been transformed (e.g., using the Box-Cox transformation), z-scores can still provide reasonable estimates.

For large sample sizes, the Central Limit Theorem supports the use of z-scores for analysing means. However, for highly skewed data or when normality is questionable, non-parametric methods or bootstrapping may yield more reliable results.

To determine the probability that the price of an event exceeds a specific value, we can use z-scores to calculate for example the probability of finding an event with a price greater than 20 units. The z-score is computed to determine how many standard deviations the target price is from the mean of the transformed dataset. Then, the cumulative distribution function (CDF) is used to find the probability of a value being less than or equal to this z-score.

```
Z-score for price target 20: -0.31
Probability of finding an event with a price greater than 20 units: 0.62
```

The z-score for a target price of 20 units is approximately -0.31, indicating that the price is 0.31 standard deviations below the mean of the Box-Cox transformed dataset. The probability of finding an event with a price greater than 20 units is approximately 62.3%, suggesting that most event prices lie above this value.



The visual representation in the second subplot shows the area under the normal curve corresponding to prices greater than 20 units, with the shaded red area representing this probability. This insight can help in understanding how typical or rare it is for events to exceed a specific price threshold.