Predicting Hotel Booking Cancellations and Average Daily Rate (ADR) for Optimized Revenue Management

Sainath Aittla
SEAS
University at Buffalo
UBIT: 50557807
saittla@buffalo.edu

Santoshi Reddy Chintala
SEAS
University at Buffalo
UBIT: 50559623
santoshi@buffalo.edu

Pranavi Chintala
SEAS
University at Buffalo
UBIT: 50563997
pchintal@buffalo.edu

***Abstract*—The hospitality industry has to bear a lot of revenue optimization difficulties because of its fluctuating pattern of booking and very high cancellation rates. This research will analyze hotel reservation data with the aim of identifying the most influential factors governing cancellations in bookings, and forecasting the average daily rate, a critical performance metric defined as the ratio of total lodging transactions to the total number of nights stayed. The dataset used for this study is from Kaggle, containing 119,390 records across 32 attributes, reflecting the whole gamut of booking trends, demographics of guests, and status of reservations. We go through a lot of data cleaning and processing: handling missing values, removing columns because many are irrelevant, and converting data types in order to prepare this dataset for predictive modeling and analysis. The research will, therefore, be instrumental in guiding hotels to make use of dynamic pricing in improving customer experiences and maximizing occupancy by addressing questions to do with cancellations and ADR. Results from the project have a great possibility of considerably helping revenue management practices within the hospitality industry.**

I. Introduction

The hospitality industry is among the most growth-oriented and competitive sectors around the world's economy. Hotels have to deal with market demand, consumer behavior changes, and seasonality to maintain profitability on a day-to-day basis. Two of the most critical factors in revenue management for a hotel include cancellations made on reservations and Average Daily Rate. A high level of cancellation rate can seriously disrupt the operations of a hotel and lead to huge losses in revenues and resource utilization mismatch. On the other hand, ADR is a major indicator of performance since it is a reflection of the hotel's power in charging and deriving revenue from selling the rooms. Comprehension and anticipation of these factors are very important in allowing hotels to develop right pricing techniques that ensure full occupancy and customers enjoy their stay.

The goals of this research include an investigation into the pattern exhibited by customers based on hotel booking data, identification of major drivers of cancellation, and the prediction of ADR. This will, in turn, help hotels make major informed decisions by reducing cancellations, optimizing prices, and enhancing the overall customer experience. In fact, ADR prediction is a key feature because it draws a picture of the financial health of a hotel. It empowers these companies to change their strategies in any given time, based on market conditions, seasonality, and demographics.

The dataset for this study has been derived from Kaggle, containing 119,390 records with 32 attributes. This represents a very rich source of information, including types of hotels, booking dates, guest demographics, length of stay, and reservation status. This elongated dataset will enable them to explore exhaustive dimensions of factors that affect cancellations and ADR for strategic revenue management.

The data cleaning and different preprocessing steps get the dataset ready for analysis and modeling. We want to comprehend which characteristics in bookings are most closely linked with cancellations, what guest profile has the most impact, and how this influences ADR. This, in turn, shall provide practical insights for hotels to shape their operations, offer personalized marketing, and optimize profitability. This now helps hotels gain better insight into more pragmatic ways to reduce revenue loss due to cancellations and to optimize their pricing models towards the demands of customers and market dynamics.

II. Problem Statement

The project will analyze hotel booking data to demonstrate trends in customer bookings, patterns, and leading causes of cancellations. It shall then focus on the ADR prediction, a critical financial metric in the hospitality industry that refers to the total sum of money earned from all lodging transactions divided by the total number of nights stayed. Since the hotel business is heavily dependent on ADR, the better the prediction, the more it is able to help optimize revenue management and pricing strategies. Therefore, this study tries to answer some questions linked to the improvement of predictions.

Which factors create cancellations? How effectively can we forecast ADR by using various booking and customer attributes? Which are the seasonal and market variations with respect to both hotel bookings and ADR? The project gives

answers to all these questions so that hotels take positive steps toward customer satisfaction, improvement in revenue streams, and optimization of their operations.

A. Background and Significance

Revenue management is a significant factor in the success of hotels operating in today's highly competitive hospitality business environment. Hotels apply ADR as one of the main performance measures that provide an indication of their financial health. Variations in ADR also contribute to high cancellation ratios, thereby yielding massive shifts in revenue and operational efficiency among hotels. By doing this, the hotels will be able to compete in the market because, through this, they will have a deep understanding of the pattern of bookings, identification of key drivers leading to cancellations, and by forecasting ADR accurately, they can offer optimized pricing strategies.

The addressal of such a problem is of great significance for several reasons:

It enables the hotel to manage revenue by dynamic pricing—higher when demand is high and evening out the occupancy when demand for business is at a low.

Customer Retention Service: With the analysis of booking behavior and cancellation trends, hotels can come up with effective policies and promotions that will enhance customer satisfaction and loyalty.

Resource Allocation: Accurate ADRs will definitely ensure a proper utilization of resources, improving service and enhancing customer experience. In this respect, considering these factors, the contribution of the project can consist in the fact that with its help, data-driven insights will be able to provide hotels with an opportunity to elaborate on business strategies given the ever-changing market dynamics.

B. Contribution Potential

The objective of this project is to give an added dimension to hotel revenue management through predictive analytics. By comprehending the causes of cancellation and ADR, hotels will be able to apply such knowledge in fine-tuning their price models and booking policies toward enhanced profitability and customer satisfaction. The predictions and insights derived from this study will be crucial for the following:

Dynamic Pricing: Development of models that recommend the optimal room rates based on the prediction made of ADR.

Cancellation Reduction: The process of identifying high-risk bookings and the application of targeted interventions to minimize cancellation rates.

Strategic Marketing: We can segment our customers by their booking habits and tailor promotions or offers accordingly.

In all, the outcome of this project can greatly revisit some key decision-making processes in hotels for better customers' experiences, coupled with financial returns.

III. Data Sources

The data score used in this project was obtained from Kaggle; it is the "hotel-booking-demand" dataset. The dataset contains 119,390 records with 32 attributes describing comprehensively most of the dynamics that exist in hotel bookings. It addresses, among other things, hotel type, booking dates, customer demographics, length of stay, booking changes, special requests, reservation status, and the financial transactions at the Average Daily Rate of ADR. These attributes make the dataset very ideal for exploring the trends in booking, extracting useful insights into customer behavior, and predicting key metrics such as booking cancellations and ADR.

The data are thoroughly explored using techniques of Exploratory Data Analysis referenced from the NIST EDA guidelines. Seriously, the focus of this analysis has been to understand the distribution, central tendency and variability, and relationships in key variables, especially between 'is-cancelled' and 'adr'. The philosophy herein will therefore provide a basic understanding of how the data is structured to drive further modeling and prediction efforts.

This dataset offers the following extensive collection of booking-related features that give a granular view to realize comprehensive analysis. These capture various factors driving cancellations and ADR at the core of driving data-driven decision-making in hotel revenue management. By analyzing this rich dataset, hotels will be able to find patterns and trends that enable them to optimize their revenue strategy and enhance customer satisfaction while ensuring efficiency improvement across all operations.

IV. Data Cleaning

1) Removal of Irrelevant Columns: The columns agent and company were removed due to a high percentage of the number of missing values and very little relevance to the project goals. This reduces noise and unnecessary complexity in the data.
2) Handling Missing Values: Any rows that contained null values were dropped. This will ensure that only complete records are used in the analysis and modeling, hence preserving the integrity of the data.
3) Data Type Conversion: The conversion to datetime format for the column reservation_status_date was performed in order to allow for such analyses that find booking trends or seasonality time-based.
4) Removing Duplicate Records: Added to this, duplicate entries were removed in order to avoid biased or skewed analyses. This means each record of the booking is unique and hence could give a correct representation of the pattern of bookings.
5) Combining Date Columns: A new column, called arrival_date, was created by combining the three columns: arrival_date_year, arrival_date_month, and arrival_date_day_of_month. This consolidates the date information into a single column, making the operations and analyses that require a date easier to carry out.
6) Dropping Old Date Columns: Once the arrival_date column was created, the original date columns arrival_date_year, arrival_date_month, and arrival_date_day_of_month were removed. This

step cleans up the dataset by retaining only the new unified date column for further analysis.

7) Season Extraction: The seasonal column is developed by extracting the month from arrival_date and mapping it to a season Winter, Spring, Summer, or Autumn. This column will be useful for the analysis of the seasonal pattern of bookings and ADR.
8) Combining 'Babies' and 'Children' Columns: The babies and children columns have been summed into a new column, kids. The old columns have then been dropped. Because of this change, the data is now further consolidated and more accurately describes the number of young guests for each booking.
9) Rearranging Columns: Rearranging the columns in a more logical order for readability. Placing key columns like 'hotel', 'arrival_date', 'is_canceled', and others first
10) Removing Negative Values: As adr i.e., average daily rate for the hotel booking cannot be negative, we are dropping all the rows with adr¡0.
11) Removing Inconsistent Rows: Removing all rows where adults are none and kids are none. As this kind of data has positive adr values along with different types of reseravtion status. When no one is staying in the room then it doesn't make any sense it keep those rows.
12) Dropping Rows according to Domain Knowledge: The other case would be there are no adults in the room while there are kids alone staying in the room. This is not a feasible situation. Hence dropping those rows.
13) Removing Outliers: We have removed outliers for the target variable adr. The outliers are found using box plot. In total, steps above increase the quality of the dataset as a whole, which can then be used for deep dive analysis and modelling to drive insights.

V. Exploratory Data Analysis

The EDA followed the principles proposed in the NIST publication and John Tukey's exploratory data analysis. Objectives were to find the critical ones from raw hotel booking data and know what numerical fields affect booking cancellations [among which features] as well as Average Daily Rate (ADR). Ultimately, this will inform the feature selection, engineering and processing for predictive modelling. The below are the different and important EDA operations which have to be done with their:

1) Identification of Categorical and Numerical Features
   - Operation: Checked the data types of each column to come up with categoricals: 18, and numericals: 12 in number.
   - Outcome: This identification allowed us to know which of these are the columns that have to undergo a categorical encoding, while rest can directly be used for Statistical Modeling. Such as hotel, meal,market_segment etc, were detected as categorical and lead_time,adr, stay_in_week_nights were numerical.
   - Use: This information is vital for preprocessing i.e. encoding categorical variables and scaling numerical columns before they can be fed into the machine learning model.
2) Exploration of Unique Values in Categorical Features
   - Operation: Unique values of the categorical features (hotel, meal and country) were printed to give an idea about the breadth and some clusters inside such columns.
   - Outcome: Known categories for each feature like meal types (BB, FB, HB, SC, Undefined) and countries of origin. These discoveries enabled the team to identify some tentative properties likely to affect booking behavior.
   - Use: This information was then leveraged to choose grouping or aggregation features, like which countries could be grouped together into regions and create more valuable variables.
3) Distribution of Hotel Types
   - Operation: The pie chart shows the relationship between City Hotel and Resort Hotel.
   - Outcome: The results are, 61.4% of bookings were for City Hotel and 38.6% were for Resort Hotel.
   - Use: This information is the key to understanding the customer's preferences. Type of the hotel will be remained as an attribute to check whether it has equal effects on ADR and cancellations.

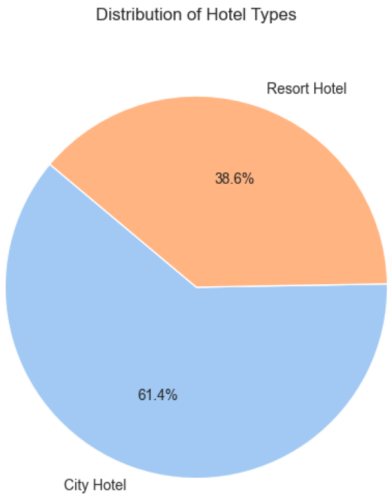

Fig. 1. Distribution of Hotels

4) Cancellation Distribution
   - Operation: Produced the bar chart distribution of the is_canceled variable.
   - Outcome: Around 27.6% of the bookings were canceled, implying that the data set is imbalanced.
   - Use: Thus, this observation indicates the essentialness of several methods that can be used to deal with imbalanced data, like, resampling or using weighted

loss functions, while the modeling phase is taking place.

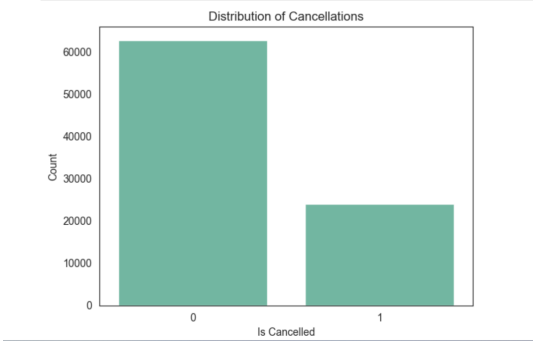

Fig. 2. Distribution of Cancellations

5) Cancellations by Hotel Type
   - Operation: Charted a grouped bar chart to analyze city and resort hotel cancellation statistics.
   - Outcome: City Hotels had higher cancellation rates (30.1%) than the Resort Hotels (23.7%).
   - Use: This link tells that the hotel type is a strong feature in the cancellation prediction models, so it should be kept in further analysis.

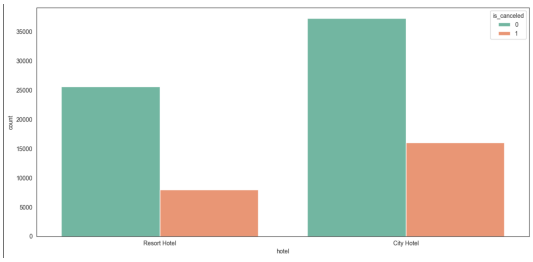

Fig. 3. Cancellations by Hotel Type

6) Descriptive Statistics for ADR
   - Operation: Generated and showed the descriptive statistics of the adr column, consisting of mean, median, standard deviation, and range.
   - Outcome: The ADR had an average of 106.7, a median of 98.67, and the range was 0 to 510, which proves that the data are highly varied and have a right-skewed distribution.
   - Use: The skewness implies logarithm transformation for data normalization, which has to be done before regression modeling on ADR takes place.

7) Box Plot of ADR by Cancellation Status
   - Operation: Developed a specific box plot that compares the ADR distribution of canceled and non-canceled bookings.
   - Outcome: The cancelled booking point had slightly higher median ADR values numerous of which were outliers on the higher end.
   - Use: This pattern is an indication that higher ADR might be risk of cancellations and thus a vital feature for cancellations forecasts.

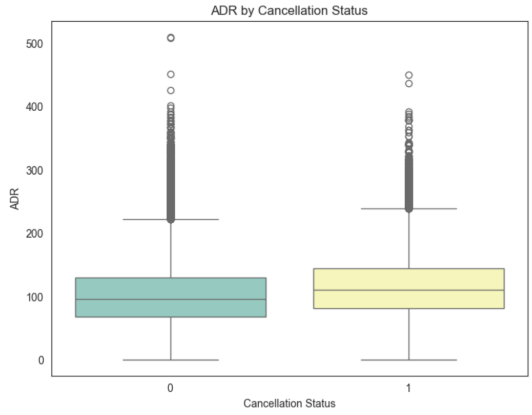

Fig. 4. ADR by Cancellation Status

8) Correlation Matrix
   - Operation: Calculated a correlation matrix to see how numerical features like lead_time adr, and previous_cancellations relate to each other.
   - Outcome: This showed that lead_time has a positive link to is_canceled (0.29). It also revealed that adr has some connection to lead_time and stays_in_week_nights.
   - Use: These correlation findings helped me choose features for later modeling. They pointed out that lead_time, previous_cancellations, and adr might be good predictors.

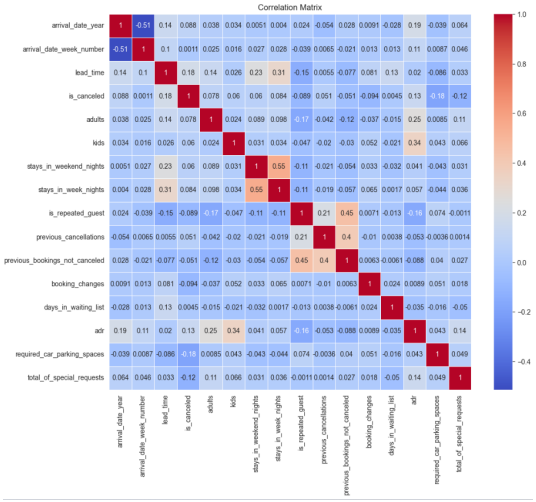

Fig. 5. Correlation Matrix

9) Room Type Impact on Cancellation
   - Operation: Visualized bar chart to show how different room types affect booking cancellations.
   - Outcome: The chart shows that: Room type 'A' has the most bookings, with 'D' and 'E' coming

next. Room types 'A', 'D', and 'E' also have more cancellations (darker bars). This suggests people cancel these room types more often. Other room types like 'C', 'B', 'F', 'G', 'H', 'I', 'K', and 'L' have fewer bookings and cancellations.

- Use: This review shows that room type plays a key role in forecasting cancellations. Hotel managers can focus on room types with higher dropout rates to cut down on booking cancellations. They might tougher the cancellation rules or offer deals for confirmed bookings. Also, the model to predict cancellations will include room type as a category to consider.

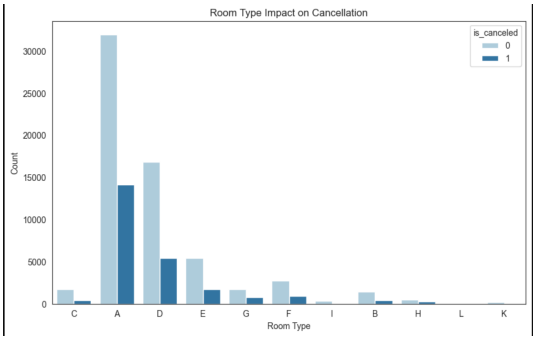

Fig. 6. Room Type Impact on Cancellation

10) ADR Over Time

- Operation: Plotted ADR trend over time to spot any patterns linked to time.
- Outcome: By observation, ADR rose from 2015 to 2017 hinting at a possible effect of time on ADR.
- Use: This points to adding time-related features, like arrival_date_year and arrival_date_month, in models to predict ADR. These features can catch changes tied to seasons and years.

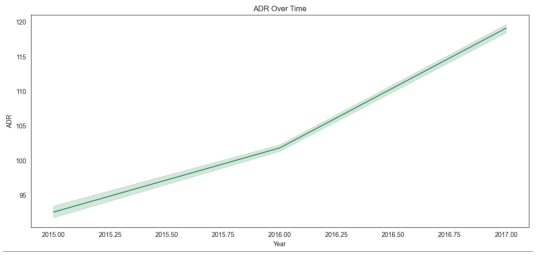

Fig. 7. ADR Over Time

11) ADR by Market Segment

- Operation: Created a box plot to show how ADR differs across market segments (for example, Direct, Corporate, Online TA).
- Outcome: Some market segments such as Direct and Online TA, showed a wider range and higher median ADRs. In contrast, Complementary bookings had a lower ADR.
- Use: This analysis confirms that market_segment is a key feature to predict ADR, which will guide future feature engineering steps like one-hot encoding.

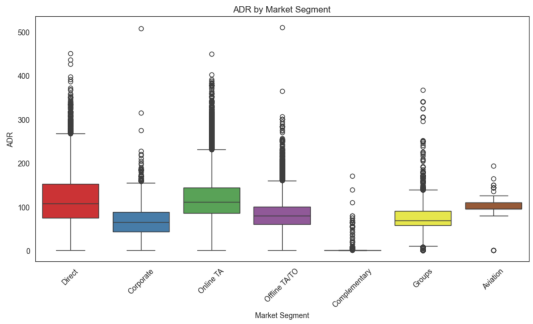

Fig. 8. ADR by Market Segment

12) Pair Plot for Cancellation Analysis

- Operation: created a pair plot to look at the connections between important features (lead_time, adr stays_in_week_nights previous_cancellations) and how they affect cancellations.
- Outcome: Higher lead_time links to top chance of cancellations, and certain groups in the ADR spread tied to cancellations.
- Use: These results highlight that these features matter a lot to predict cancellations helping to choose machine learning models.

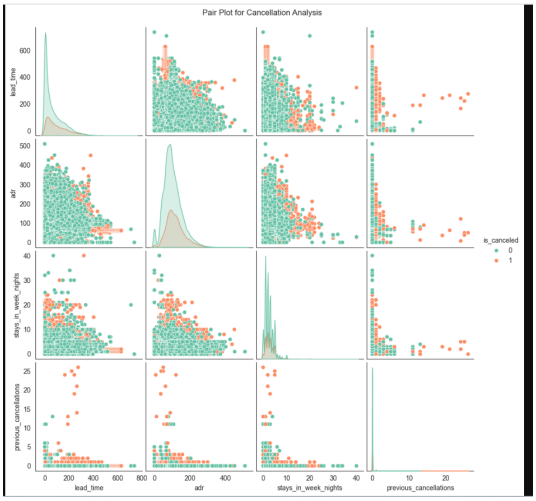

Fig. 9. Pair Plot

13) Seasonal Variation in Cancellations

- Operation: Showcased cancellations across different seasons using a bar chart.
- Outcome: Summer saw more cancellations showing how seasons affect booking habits.
- Use: To make better predictions about cancellations, we should use the arrival_date_month as a feature for the model.

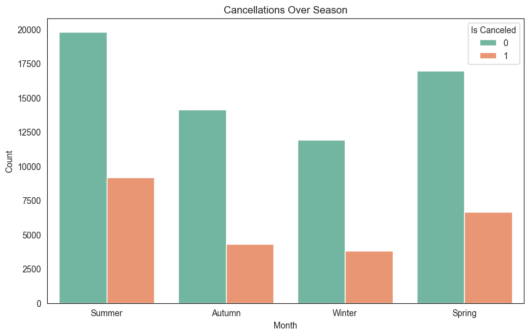

Fig. 10. Seasonal Variation in Cancellations

14) ADR Distribution
- Operation: Visualized histogram with a density plot to show how ADR spreads out.
- Outcome: Most ADR values bunch up between 50 and 150, with a long tail to the right.
- Use: This underscores the need for feature scaling or transformation methods, like log transformation, to normalize the data before training the model.

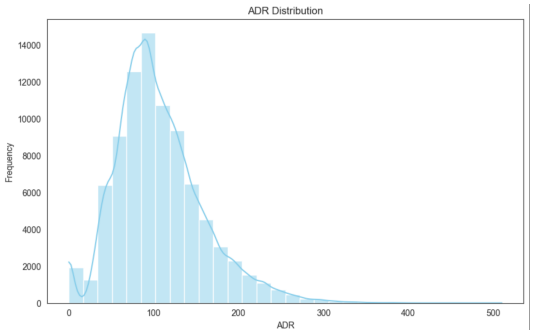

Fig. 11. ADR Distribution

15) Monthly Variation in ADR
- Operation: Created a box plot to show how ADR changes each month.
- Outcome: ADR peaks in August and December showing room rates go up during vacation times.
- Use: The changes in ADR from month to month indicate that to use arrival_date_month to predict seasonal effects on ADR when building models.

16) Mean and Standard Deviation of ADR by Month
- Operation: Graph showing the average ADR and how much it varied for each month.
- Outcome: ADR's average and variability change a lot throughout the year, with noticeable high points in August.
- Use: By adding features specific to each month, we can capture the changes, which will make ADR predictions more accurate.

Summary of Outcomes and Usage
- Feature Selection: The analysis identified key factors that have effect on cancellations and ADR, including hotel, lead_time adr market_segment, arrival_date_month, and is_canceled.
- Feature Engineering: Aimed to make new features like season (from arrival_date_month) and change some data (for example, to even out ADR).
- Data Processing: The way the data is spread out and varies shows we need to balance it and turn categorical variables into numbers to get the dataset ready for machine learning models.

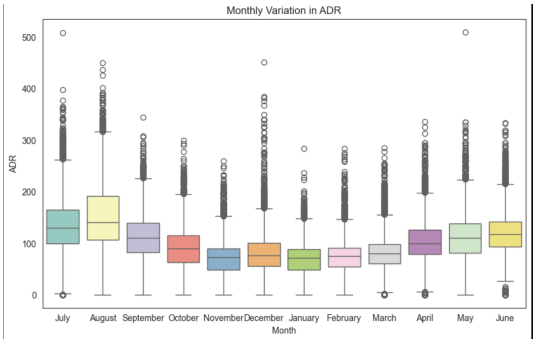

Fig. 12. Monthly Variation in ADR

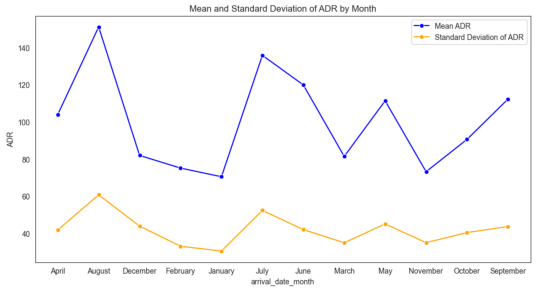

Fig. 13. Mean and Standard Deviation of ADR by Month

These EDA steps gave a complete view of the dataset. They showed how to prepare the data to predict things and build models to guess cancellations and ADR. We plan to use 6 models from the following list or other models in the prediction phase.

	Model
1	Random Forest
2	Multinomial Logistic Regression
3	SVM
4	Linear Regression
5	kNN
6	Naive Bayes
7	Decision Tree
8	Neural Network

TABLE I
MODELS

VI. Data Preprocessing

Effective data preprocessing is essential to ensure high-quality data for analysis and modeling. In this project, preprocessing was performed to clean, transform, and prepare

the data for predicting the *Average Daily Rate (ADR)* and the *is_cancelled* attribute from the hotel booking dataset. Below are the steps involved in the preprocessing pipeline:

A. Encoding Categorical Variables

Objective: Convert categorical variables into numerical formats to be used in machine learning algorithms.
Approach:

- Applied *One-Hot Encoding* for nominal categorical features such as *hotel*, *meal*, and *customer_type*, generating dummy variables for each category.
- Applied *Label Encoding* where applicable for ordinal variables, such as *reservation_status*.

B. Date Feature Extraction

Objective: Extract key date-related information to create new features for potential patterns related to the time of reservation and arrival.
Approach:

- Extracted the year from the *reservation_status_date* column and created a new column *res_year*.
- Extracted the month from the *reservation_status_date* column and created a new column *res_month*.
- Extracted the day from the *reservation_status_date* column and created a new column *res_day*.
- Extracted the day from the *arrival_date* column and created a new column *arrival_day*.

C. Scaling and Normalization

Objective: Standardize numerical features to ensure that all variables are on a similar scale, which is particularly important for models sensitive to feature magnitudes (e.g., SVM, k-NN).
Approach:

- Applied *Min-Max Scaling* to rescale the values of features like *adr*, *lead_time*, *days_in_waiting_list*, and *total_stay_length* to a range of [0,1].
- Used *Standard Scaling* (mean = 0, standard deviation = 1) for other continuous variables like *total_guests* and *days_in_advance*.

D. Splitting the Data

Objective: To evaluate the performance of the model on unseen data.
Approach:

- Split the dataset into training and test sets using an 70/30 split, ensuring that the model was trained on one part of the data and validated on another.
- Applied cross-validation to further ensure the model's robustness.

VII. Classification Models and Performance Overview

This section presents the six classification models applied to predict hotel booking cancellations, along with their performance metrics—**Accuracy**, **Precision**, **Recall**, and **F1 Score**. These metrics provide a clear understanding of how well each model performed, considering both the imbalanced nature of the dataset and the complexity of relationships between features. We have used GridSearchCV to find the best models.

A. Logistic Regression (Classification)

Why Chosen: Logistic Regression is a simple and interpretable model commonly used for binary classification tasks. It assumes a linear relationship between the features and the log-odds of the target variable (cancellation).

Tuning and Training:

- Max iterations: Set to 1000 to ensure convergence.
- Class imbalance: Handled with `class_weight='balanced'`.

Metrics:

- Accuracy: 98.86%
- Precision: 0.9887
- Recall: 0.9886
- F1 Score: 0.9885

Effectiveness: Logistic Regression performed well, achieving high accuracy and F1 score. It is a robust baseline model that handles imbalanced data efficiently. However, it was outperformed by more sophisticated models, especially in capturing non-linear relationships.

Intelligence Gained: The linear assumption of the model provided basic insights but missed some of the more complex interactions in the data.

B. Decision Tree (Classification)

Why Chosen: Decision Trees can capture non-linear relationships between features, making them highly interpretable and powerful for classification tasks. They are also effective at handling both categorical and numerical data.

Tuning and Training:

- Max depth: Set to 5 to avoid overfitting.
- Minimum samples per split and leaf: Configured to ensure sufficient samples in each node.
- Class imbalance: Addressed using `class_weight='balanced'`.

Metrics:

- Accuracy: 99.02%
- Precision: 0.9903
- Recall: 0.9902
- F1 Score: 0.9901

Effectiveness: The Decision Tree model performed exceptionally well, capturing complex relationships between features. It achieved high precision and recall, providing more nuanced predictions than Logistic Regression.

Intelligence Gained: Learning curve of the Decision tree is higher than the Logistic Regression. The tree structure provided clear decision-making rules for cancellation prediction.

C. K-Nearest Neighbors (KNN) (Classification)

Why Chosen: K-Nearest Neighbors is a non-parametric algorithm that predicts labels based on the majority vote

of neighboring data points. It is useful when the decision boundary is non-linear, but it is sensitive to the choice of `k`.

Tuning and Training:

- The number of neighbors (`n_neighbors`) was set to 5, which is the default setting.

Metrics:

- Accuracy: 95.41%
- Precision: 0.9552
- Recall: 0.9541
- F1 Score: 0.9531

Effectiveness: KNN performed reasonably well but was outperformed by models like Decision Tree and Random Forest. Its sensitivity to high-dimensional data and the choice of neighbors led to lower accuracy and F1 scores.

Intelligence Gained: Training and testing accuracy varied significantly for K-NN algorithm .It did not provide significant additional insights compared to other models.

D. Naive Bayes (Classification)

Why Chosen: Naive Bayes is a simple yet effective probabilistic classifier based on Bayes' theorem. It is computationally efficient and often performs well when features are conditionally independent, which is assumed by the model.

Tuning and Training:

- Gaussian Naive Bayes was applied, assuming a normal distribution for continuous features.

Metrics:

- Accuracy: 97.63%
- Precision: 0.9782
- Recall: 0.9763
- F1 Score: 0.9766

Effectiveness: Naive Bayes performed well but was limited by its assumption of feature independence. While it provided decent performance, it was outperformed by more complex models that captured interactions between features.

Intelligence Gained: Naive Bayes has performed better in comparison to K-NN in terms of accuracy.It lacked the ability to capture more complex dependencies, reducing its overall effectiveness.

E. Support Vector Classifier (SVC) (Classification)

Why Chosen: SVC is effective for binary classification problems, especially when the decision boundary is non-linear. The linear version (`LinearSVC`) was applied for computational efficiency.

Tuning and Training:

- Class imbalance: Handled with `class_weight='balanced'`.
- The linear kernel was chosen to compare performance with other linear models like Logistic Regression.

Metrics:

- Accuracy: 98.89%
- Precision: 0.9891
- Recall: 0.9889
- F1 Score: 0.9888

Effectiveness: SVC performed well, achieving similar results to Logistic Regression but requiring more computational resources. It handled the decision boundary effectively but was outperformed by Random Forest.

Intelligence Gained: SVC has performed better in terms of accuracy except for Random Forest Classifier.It did not provide additional insights compared to Random Forest Classifier models.

F. Random Forest (Classification)

Why Chosen: Random Forest is an ensemble learning method that combines multiple decision trees. It captures non-linear relationships and interactions between features, making it ideal for complex datasets with imbalanced classes.

Tuning and Training:

- 500 trees were used (`n_estimators=500`) to improve model robustness.
- Max depth: Set to 10 to avoid overfitting.
- Class imbalance: Handled with `class_weight='balanced'`.

Metrics:

- Accuracy: 99.96%
- Precision: 0.9996
- Recall: 0.9996
- F1 Score: 0.9996

Effectiveness: Random Forest significantly outperformed all other models, achieving near-perfect metrics across the board. It captured both linear and non-linear relationships and handled the imbalanced dataset effectively.

Intelligence Gained: Random Forest has out performed all the other classification models in terms of accuracy and Learning curve.

VIII. Comparison of Classification Models

Model	Accuracy	Precision	Recall	F1 Score
Logistic Regression	98.86%	0.9887	0.9886	0.9885
Decision Tree	99.02%	0.9903	0.9902	0.9901
KNN Classifier	95.41%	0.9552	0.9541	0.9531
Naive Bayes	97.63%	0.9782	0.9763	0.9766
SVM Classifier	98.89%	0.9891	0.9889	0.9888
Random Forest	99.96%	0.9996	0.9996	0.9996

TABLE II
Comparison of Classification Models

A. Logistic Regression

Strengths:

- Logistic Regression provides a solid baseline for binary classification. It performed well across all metrics, with **98.86% accuracy** and a well-balanced **F1 score** of 0.9885, indicating a good trade-off between precision and recall.

Weaknesses:

- Logistic Regression is limited by its assumption of linear relationships between features and the target variable.

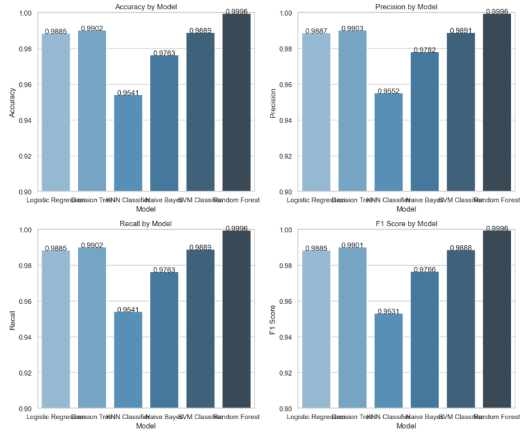

Fig. 14. Classification Models Comparison

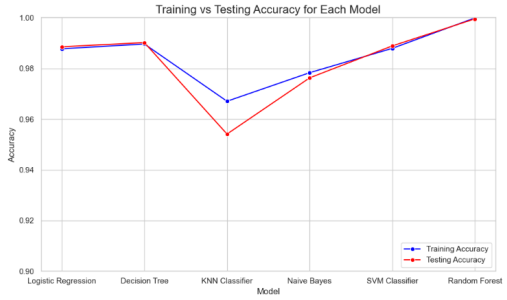

Fig. 15. Training vs Testing Accuracy

It struggled to capture the more complex interactions present in the dataset, which is why models like Decision Tree and Random Forest performed better.

When to Use: Logistic Regression is ideal for datasets where interpretability and speed are crucial. It works well when the relationship between features and the target is approximately linear or as a baseline model for comparison.

B. Decision Tree

Strengths:

- Decision Tree performed very well, achieving an accuracy of **99.02%** with a **precision of 0.9903** and an **F1 score of 0.9901**. It captures non-linear relationships effectively and provides interpretability through its tree structure.

Weaknesses:

- Decision Trees can overfit the data if not properly tuned. In this case, the tree depth and minimum samples per split were optimized to avoid overfitting. While effective, it was still outperformed by Random Forest in terms of overall accuracy and F1 score.

When to Use: Decision Trees are a good choice when interpretability is key, and the relationships between features are non-linear. It's useful when you need a clear set of decision rules, but it may require tuning to avoid overfitting.

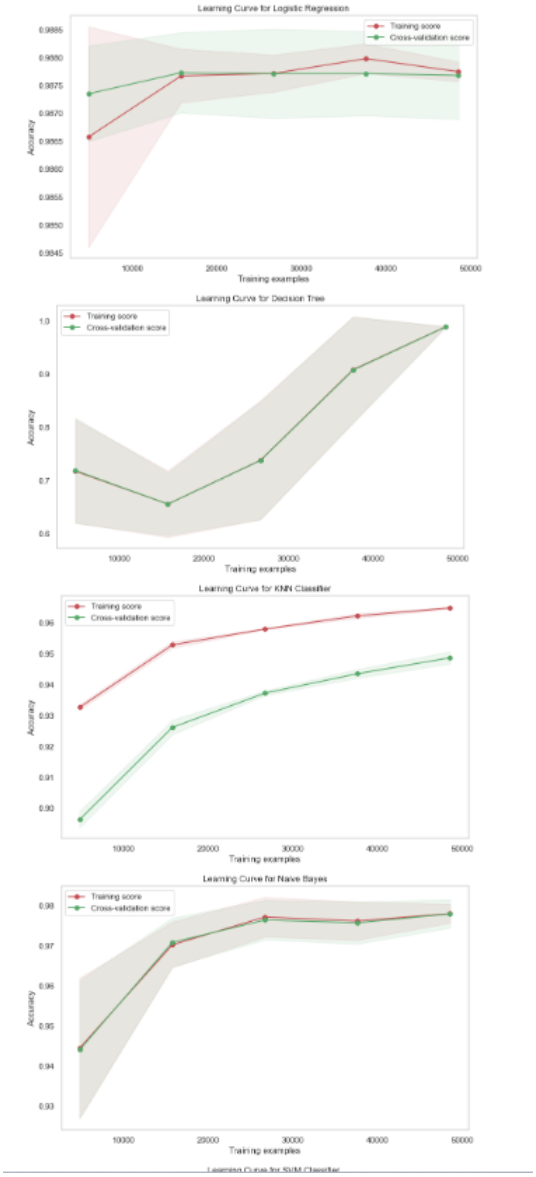

Fig. 16. Learning Curve 1

C. K-Nearest Neighbors (KNN)

Strengths:

- KNN is a simple, intuitive model that performed reasonably well with **95.41% accuracy** and an F1 score of **0.9531**. It is easy to implement and works well for small datasets with well-separated classes.

Weaknesses:

- KNN tends to struggle with higher-dimensional datasets and can be sensitive to the choice of k. It underperformed compared to models like Decision Tree and Random Forest, likely due to the complexity of the dataset.

When to Use: KNN is ideal for small, low-dimensional datasets where simplicity is key. It is computationally expensive for large datasets and is not the best choice when dealing with high-dimensional data or imbalanced classes.

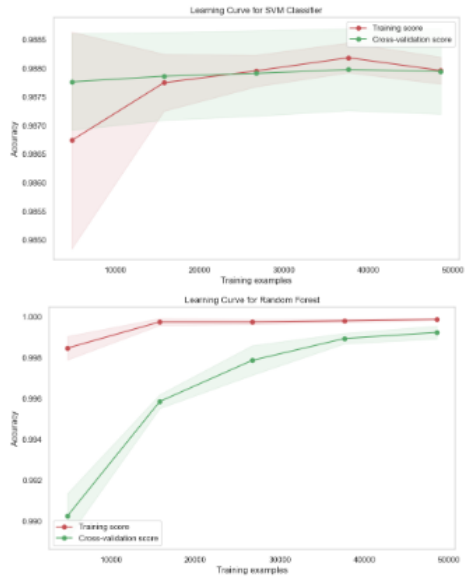

Fig. 17. Learning Curve 2

D. Naive Bayes

Strengths:

- Naive Bayes is computationally efficient and performed well, achieving **97.63% accuracy** and a balanced F1 score of **0.9766**. It is particularly effective when the features are conditionally independent.

Weaknesses:

- Naive Bayes assumes independence between features, which is rarely true in real-world data. This assumption limited its ability to capture interactions between features, leading to lower performance compared to models like Random Forest and Decision Tree.

When to Use: Naive Bayes is a great choice for simple, fast classification, especially when you have independent or weakly correlated features. It works well for high-dimensional datasets but is less effective when features are highly correlated.

E. Support Vector Classifier (SVC)

Strengths:

- SVC performed well with an accuracy of **98.89%** and an F1 score of **0.9888**. It effectively handles both linear and non-linear decision boundaries and is robust to overfitting, especially in high-dimensional spaces.

Weaknesses:

- SVC requires more computational resources compared to simpler models like Logistic Regression. While it performed well, it was still outperformed by Random Forest in terms of accuracy and F1 score. Additionally, tuning SVC can be more complex, especially with non-linear kernels.

When to Use: SVC is a good choice for datasets where the decision boundary is non-linear. It works well in high-dimensional spaces and can be effective when interpretability is not a primary concern, but computational efficiency is less critical.

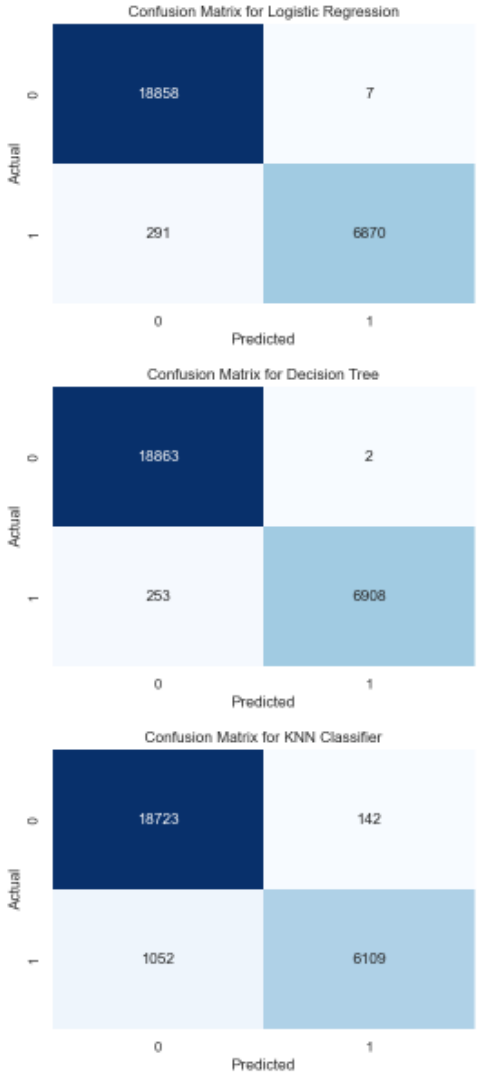

Fig. 18. Confusion Matrix 1

F. Random Forest

Strengths:

- Random Forest was the top performer, with **99.96% accuracy**, **precision, recall, and F1 score** all at **0.9996**. It captures both linear and non-linear relationships and is robust to overfitting. The model also provides feature importance rankings, offering insights into which features are most influential in predictions.

Weaknesses:

- Random Forest can be computationally intensive, especially with a large number of trees. However, the increase in computational cost is justified by its superior performance.

When to Use: Random Forest is ideal when you need the best predictive accuracy and are dealing with complex, non-linear relationships. It's particularly effective for imbalanced

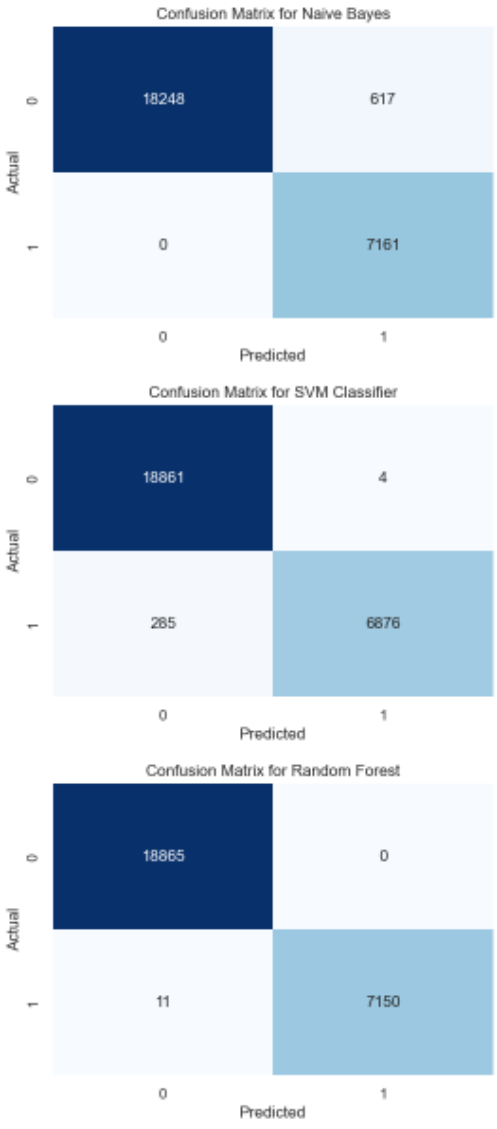

Fig. 19. Confusion Matrix 2

datasets and when you want to understand the relative importance of different features.

G. Key Insights from the Comparison

- **Best Overall Model**: **Random Forest** outperformed all other models across all metrics, achieving near-perfect results in accuracy, precision, recall, and F1 score. It is the most robust and versatile model, capturing both linear and non-linear relationships and handling class imbalance effectively.
- **Best for Simplicity and Speed**: **Logistic Regression** is the simplest and fastest model to implement. While it was outperformed by more complex models, it still provided good accuracy and is highly interpretable.
- **Best for Interpretability**: **Decision Tree** provided a strong balance between accuracy and interpretability, offering clear decision rules that are easy to understand and visualize.
- **Handling Non-linear Boundaries**: Both **SVC** and **Random Forest** excelled in handling non-linear decision boundaries, but Random Forest offered better performance overall, especially in terms of precision and recall.

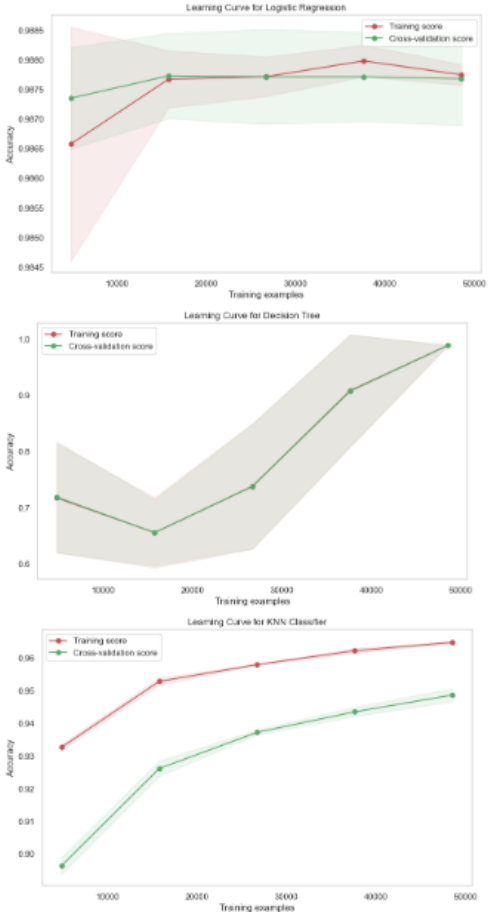

Fig. 20. Classification Models Learning Curves 1

IX. Regression Models and Performance Overview

This section illustrates the performance of the five regression models identified to predict ADR. Each of the models was optimized in terms of hyper-parameters using `RandomizedSearchCV` and the results were evaluated on the test set based on the key metrics: **Mean Squared Error (MSE)**, **Mean Absolute Error (MAE)**, **Root Mean Squared Error (RMSE)**, and **R-squared (R^2)**. These models not only provided insights into which features were strongly related to the Average Daily Rate, but also how each model fared in modeling complex relationships in the data. We have used GridSearchCV to find the best models.

A. Linear Regression

Why Chosen: Linear Regression is the simplest type of regression, based on a linear relationship between features and

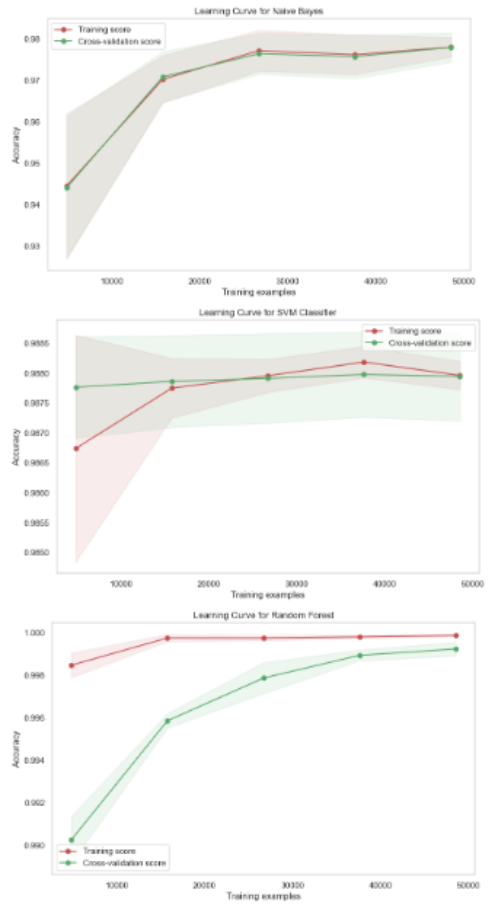

Fig. 21. Classification Models Learning Curves 2

a target variable. Thus, it was used as a baseline model for comparative evaluation.

Tuning and Training:

- No hyperparameter tuning was required, as Linear Regression does not have hyperparameters to tune.
- The model was trained on scaled data to ensure that all features were on a similar scale.

Metrics:

- **Mean Squared Error (MSE)**: 0.0063
- **Mean Absolute Error (MAE)**: 0.0557
- **Root Mean Squared Error (RMSE)**: 0.0794
- **R-squared (R^2)**: 0.3981

Effectiveness: This baseline-LR explained 39.81% of variance in ADR. However, this model has relatively high values of MSE and RMS. Hence, it faced difficulties while capturing complex nonlinear patterns in the data and was therefore less efficient for the accurate prediction of ADR.

Intelligence Gained: The model does not pick up interactions and nonlinearities hence performed poorly.

B. Random Forest Regression

Why Chosen: Random Forest represents a kind of ensemble learning, which contains many decision trees by combining nonlinear relationships and interactions among features. Besides that, feature importance rankings provide a great potentiality for prediction and interpretability.

Tuning and Training:

- `RandomizedSearchCV` was used to tune the following hyperparameters:
  - `n_estimators`: Number of trees in the forest (100, 200).
  - `max_depth`: Maximum depth of the tree (None, 10, 20).
  - `min_samples_split`: Minimum samples required to split a node (2, 5).
  - `min_samples_leaf`: Minimum samples required to be at a leaf node (1, 2).

Metrics:

- **Mean Squared Error (MSE)**: 0.0016
- **Mean Absolute Error (MAE)**: 0.0287
- **Root Mean Squared Error (RMSE)**: 0.0400
- **R-squared (R^2)**: 0.8458

Effectiveness: The Random Forest Regression has outperformed linear regression by a great margin, explaining 84.58% of variance in ADR, while the R^2 was equal to 0.8458. The model has captured the nonlinear relationships rather well. Error rates across the board were rather low, making this model highly accurate for ADR prediction.

Intelligence Gained: In fact, the model has marked `lead_time`, `total_stay_length`, and `arrival_date_month` as the most influential features on ADR. This insight provides actionable information for hotel management in the better optimization of pricing strategies based on key drivers of ADR.

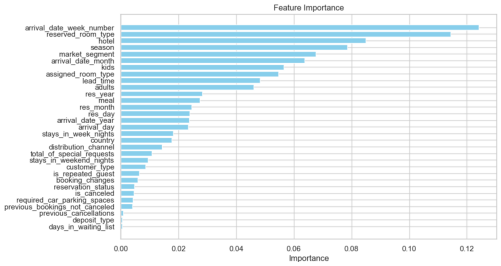

Fig. 22. Random Forest Regressor Feature Importance

C. Support Vector Regression (SVR)

Why Chosen: For such high-level complexity, the ability to handle can be achieved with SVR using margin-based optimization in order to capture the non-linear relationships. Hence, SVR was selected to investigate the capability to manage such high level of complexity regarding the prediction of ADR.

Tuning and Training:

- `RandomizedSearchCV` was used to tune hyperparameters, including:
  - `C`: Regularization parameter (0.1, 1, 10).
  - `epsilon`: Epsilon-tube within which no penalty is associated (0.1, 0.2).

- `max_iter`: Maximum number of iterations (1000, 5000).

Metrics:

- **Mean Squared Error (MSE)**: 0.0074
- **Mean Absolute Error (MAE)**: 0.0625
- **Root Mean Squared Error (RMSE)**: 0.0860
- **R-squared (R^2)**: 0.2884

Effectiveness: SVR was the under performing model when contrasted with Linear Regression and Random Forest Regression, describing only 28.84% of the variance in ADR. The model failed to perform minimization of error on the predictions, especially for larger values of ADR. Its performance may improve by further exploring non-linear kernels like RBF.

Intelligence Gained: While SVR was able to pick up some relationships between features, it added nothing new beyond what the Random Forest had shown. This relatively high error rate does suggest that this model is not well-suited for this data set in the absence of more sophisticated kernel functions.

D. Gradient Boosting

Why Chosen: Gradient Boosting is an ensemble technique of building sequenced models, aiming for the minimization of errors which occurred in previously built models. Considering that it uses trees as base models, it can be used for the enhancement of predictive accuracy and dealing with sophisticated patterns within data.

Tuning and Training:

- `RandomizedSearchCV` was used to tune the following hyperparameters:
  - `n_estimators`: Number of boosting stages (100, 200).
  - `learning_rate`: Shrinks the contribution of each tree (0.01, 0.1).
  - `max_depth`: Maximum depth of the tree (3, 5).
  - `min_samples_split`: Minimum samples required to split a node (2, 5).
  - `min_samples_leaf`: Minimum samples required to be at a leaf node (1, 2).

Metrics:

- **Mean Squared Error (MSE)**: 0.0025
- **Mean Absolute Error (MAE)**: 0.0354
- **Root Mean Squared Error (RMSE)**: 0.0497
- **R-squared (R^2)**: 0.7626

Effectiveness: It performed quite well, with the Gradient Boosting model explaining about 76.26% in the variance of ADR. It offered a lower error rate compared to Linear Regression but was slightly outperformed by Random Forest Regression. Gradient Boosting requires careful tuning of its learning rate and depth to avoid overfitting.

Intelligence Gained: The model again confirmed that features `lead_time`, `total_stay_length`, and `arrival_date_month` were the most relevant for ADR prediction. It provided very useful insights into nonlinear dependencies between features and ADR that can be used to fine-tune pricing strategies.

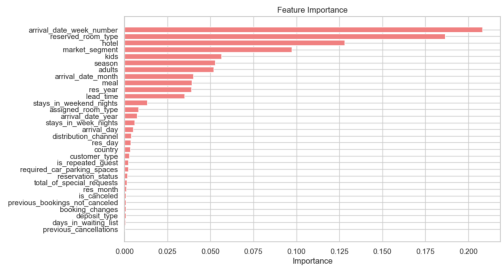

Fig. 23. Gradient Booster Feature Importance

E. Neural Network (MLP)

Why Chosen: The Multi-layer perceptron is an exciting neural network model used to model some complicated relationships in a nonlinear manner. So, here it was chosen for its capability of capturing intricate patterns in ADR prediction.

Tuning and Training:

- `RandomizedSearchCV` was used to tune the following hyperparameters:
  - `hidden_layer_sizes`: Network architecture (100, (100, 50)).
  - `activation`: Activation functions (`'relu'`, `'tanh'`).
  - `alpha`: L2 regularization term (0.0001, 0.001).
  - `learning_rate_init`: Initial learning rate (0.001, 0.0005).
  - `max_iter`: Maximum number of iterations (1000).

Metrics:

- **Mean Squared Error (MSE)**: 0.0023
- **Mean Absolute Error (MAE)**: 0.0338
- **Root Mean Squared Error (RMSE)**: 0.0480
- **R-squared (R^2)**: 0.7822

Effectiveness: The Neural Network performed well, explaining 78.22% of the variance in ADR. This did come at the cost of extensive tuning, mainly in regards to hidden layer size, activation functions, and learning rates. While promising, it lagged behind the Random Forest in terms of both performance and interpretability.

Intelligence Gained: The Neural Network does not provide the feature importance. It has performed well in terms of training and testing accuracy as well as Learning Curve.

X. Comparison of Regression Models

This section does a end-to-end comparison of all the regression models applied to predict ADR for hotel bookings. The performance of each model will be evaluated based on the **Mean Squared Error (MSE)**, **Mean Absolute Error (MAE)**, **Root Mean Squared Error (RMSE)**, and **R-squared (R^2)**. These metrics will help in understanding how each model identified the relationships of features with ADR and how each is competing with others in the race of accuracy and reduction in error.

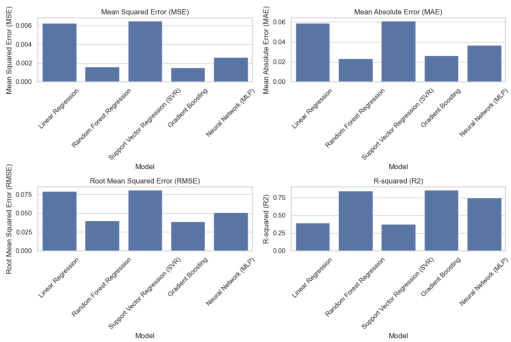

Fig. 24. Regression Models Performance Metrics

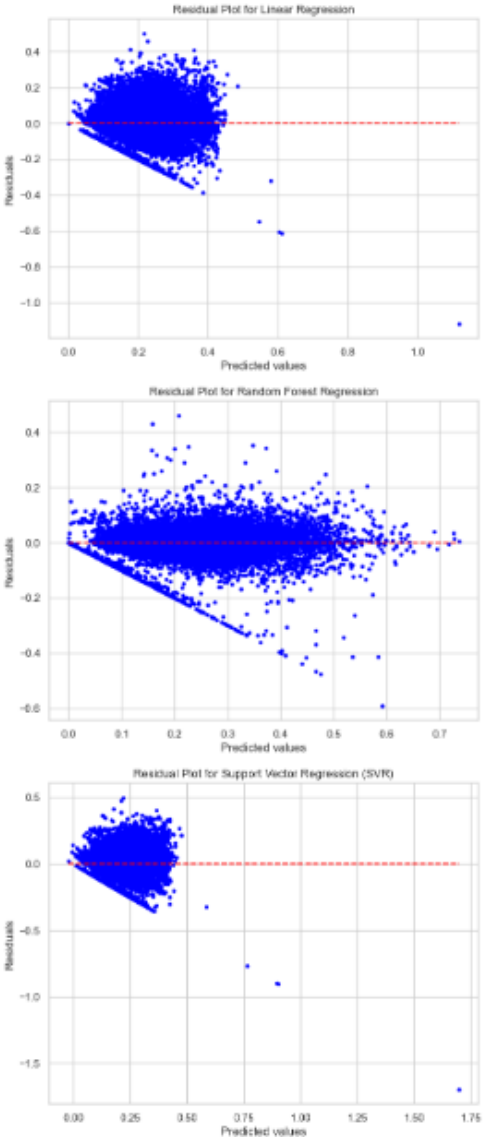

Fig. 25. Residual Plot 1

Model	MSE	MAE	RMSE	R^2
Linear Regression	0.0063	0.0557	0.0794	0.3981
Random Forest Regression	0.0016	0.0287	0.0400	0.8458
Support Vector Regression	0.0074	0.0625	0.0860	0.2884
Gradient Boosting	0.0025	0.0354	0.0497	0.7626
Neural Network (MLP)	0.0023	0.0338	0.0480	0.7822

TABLE III
SUMMARY OF MODEL PERFORMANCE FOR ADR PREDICTION

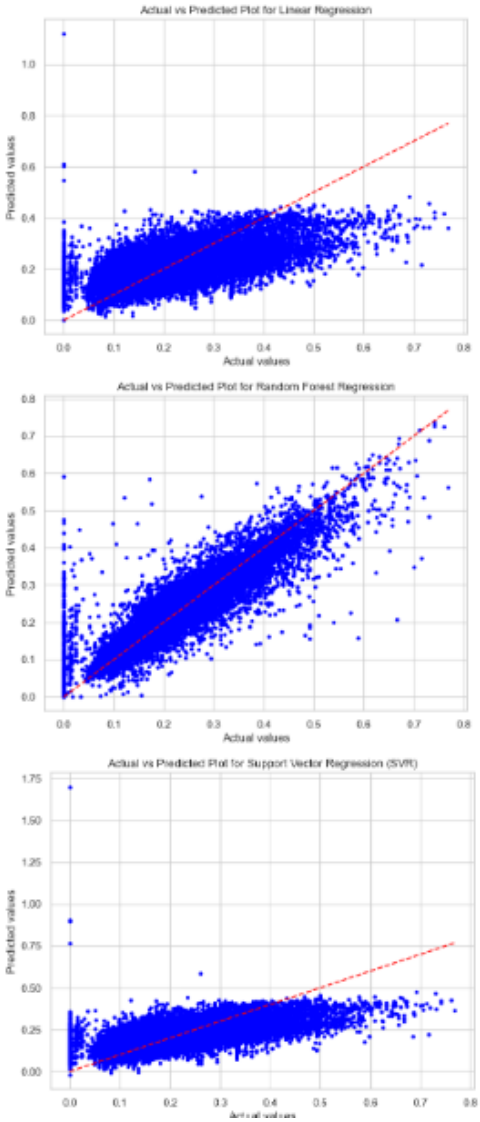

Fig. 26. Actual vs Predicted 1

A. *Summary of Model Performance*

B. *Linear Regression*

Strengths: Linear Regression is a simple, interpretable baseline model. Linear Regression works best if the relationship from features to target can be considered approximately linear.

Weaknesses: Linear Regression Analysis performed under the assumption that all relationships between the features and the target were linear; hence, limiting its capability to capture

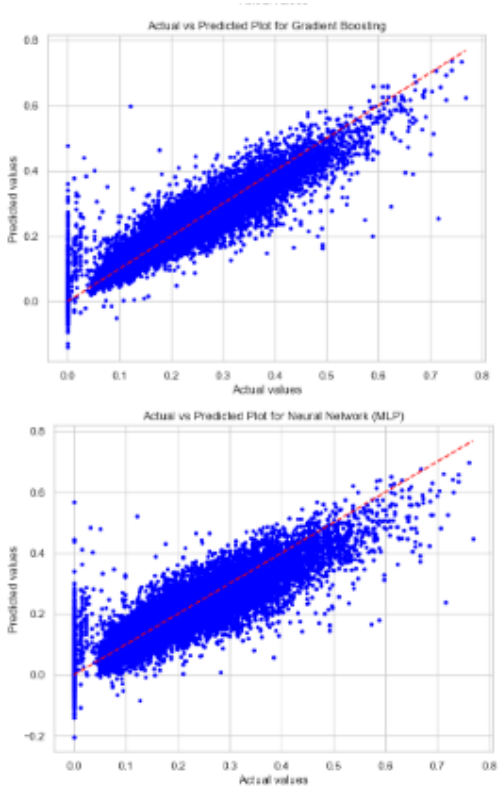

Fig. 27. Actual vs Predicted 2

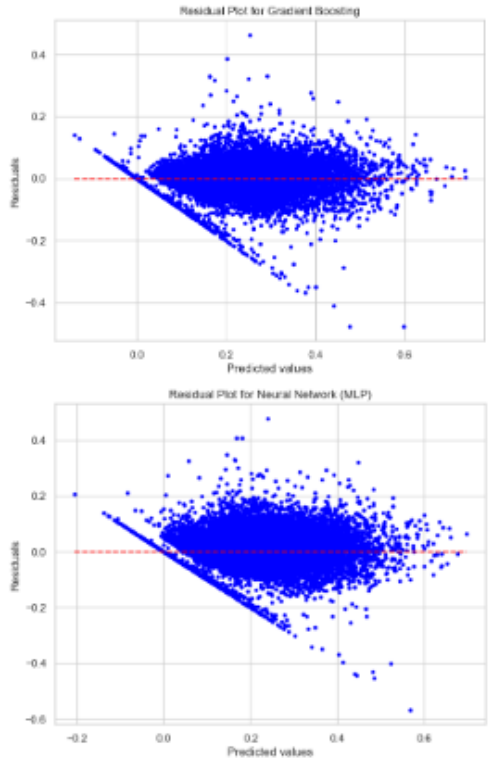

Fig. 28. Residual Plot 2

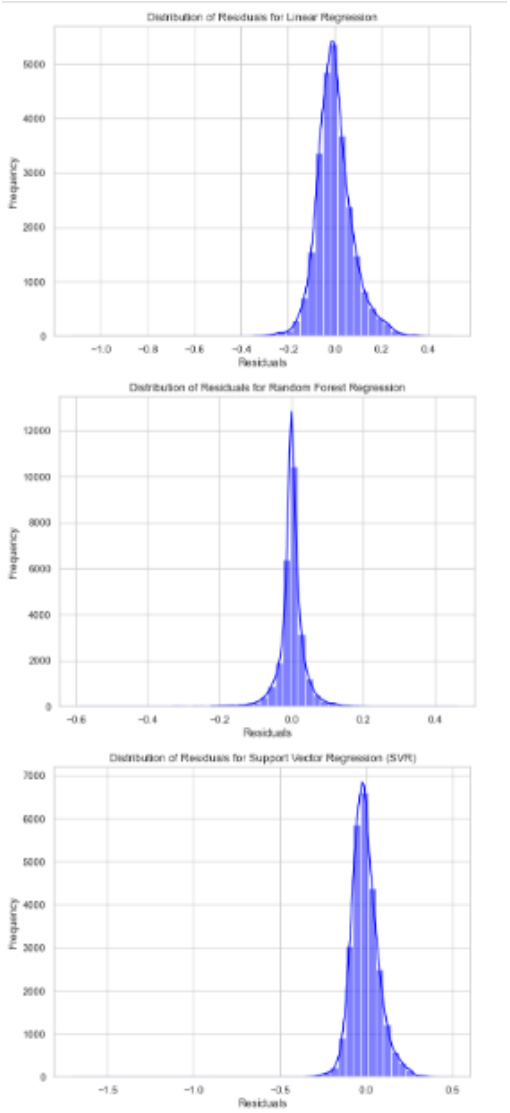

Fig. 29. Distribution of Residuals 1

complex interactions. At an **R^2 of 0.3981**, the model explained only 39.81% of ADR variance, therefore being less effective compared to the more complex ones.

Best Use Case: Linear Regression is ideal when interpretability and simplicity are paramount and the relationship between features and the target is believed to be linear.

C. Random Forest Regression

Strengths: Random Forest Regression substantially outperformed the remaining models by explaining **84.58% of the variance in ADR (R^2 = 0.8458)**. It did well in capturing non-linear relationships, had the lowest error rates, and provided feature importance rankings that help identify which factors are most influential in regard to ADR.

Weaknesses: While the expense in training a Random Forest is bigger compared to simpler versions like Linear

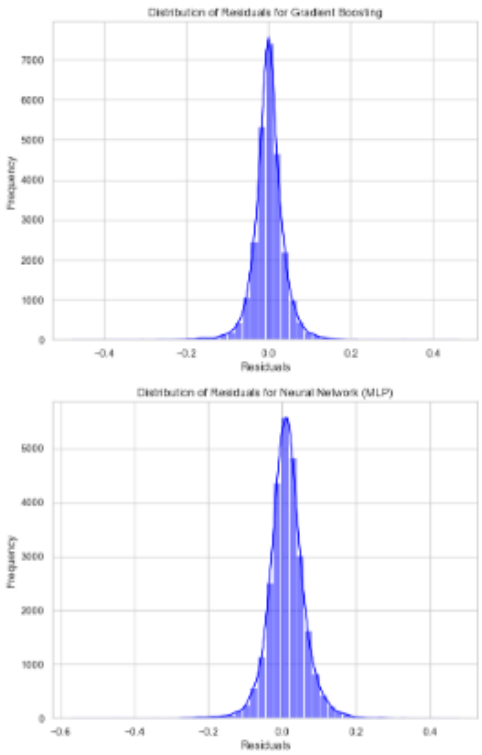

Fig. 30. Distribution of Residuals 2

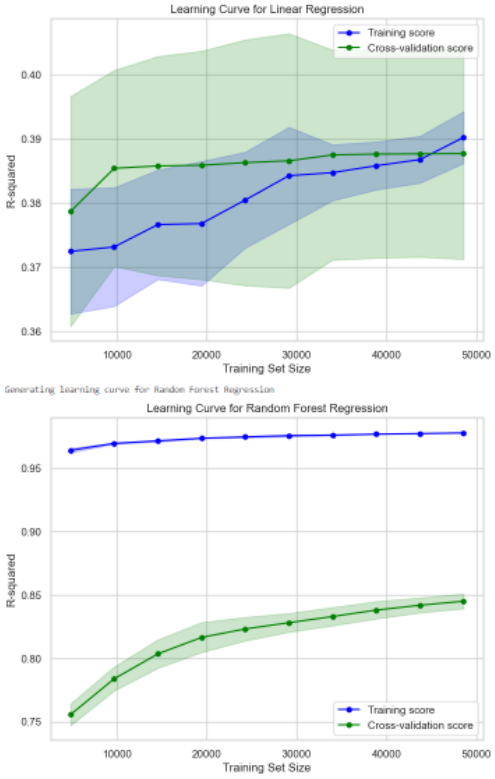

Fig. 31. Regression Models Learning Curves 1

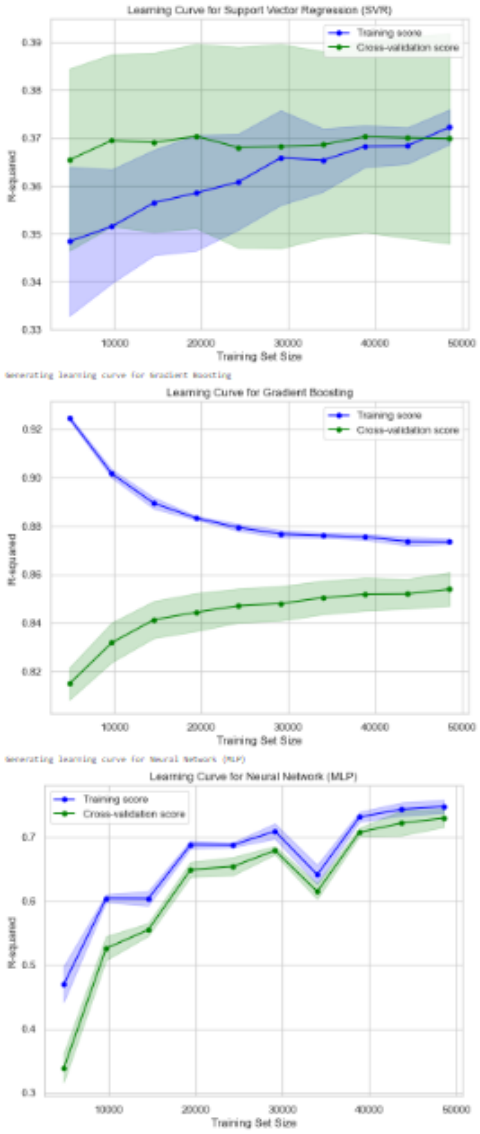

Fig. 32. Regression Models Learning Curves 2

Regression, the enhanced accuracy and reduction of error justify this increase quite well.

Best Use Case: Random Forest finds its best application in those problems that are complicated in nature, where interaction amongst various features and nonlinearities between features and the target variable exist. As a matter of fact, it's perfect when predictive accuracy is a significant need, not to say when the importance of the features also has got to be provided.

D. Support Vector Regression (SVR)

Strengths: SVR is advantageous in modeling nonlinear relationships through margin-based optimization. Properly tuned, it can handle outliers and provide robust predictions.

Weaknesses: The weak performance of SVR was attained with a value of**R^2 of 0.2884**, which explains about 28.84%

of the variance in ADR. **MSE (0.0074)** and **RMSE (0.0860)** were the highest compared to other models. It only indicates that the model is faring well with the data set when more tuning is done, like RBF as a non-linear kernel.

Best Use Case: SVR is particularly suited to small-scale datasets, with complicated non-linear relationships and few dimensions. Here, it performed worse than Random Forest and Gradient Boosting.

E. Gradient Boosting

Strengths: The ensemble learning method-Gradient Boosting-run explained about **76.26% of the variance in ADR (R^2 = 0.7626)**. This model reduced the errors effectively. Hence, it gave a low value for**MSE of 0.0025** and **RMSE of 0.0497**, hence turning out to be a reliable model for the prediction of ADR.

Weaknesses: While doing decently, even Gradient Boosting was outperformed by Random Forest both in terms of accuracy and reduction of error. It needs some more hyperparameter tuning- performant learning rate and tree depth-in order not to fall into problems such as overfitting.

Best Use Case: When problems involve non-linear associations, gradient boosting is ideal in cases where the data has non-linear relationships, and interpretability is of lesser importance than predictive performance. It is also useful when one wants to minimize prediction errors by improving weak learners sequentially.

F. Neural Network (MLP)

Strengths: These are best results among others obtained by Neural Networks, precisely Multi-Layer Perceptron (MLP), which explains **78.22% of the variance in ADR (R^2 = 0.7822)**. This means the model grasped complex data patterns since its **MSE of 0.0023** and **RMSE of 0.0480**. Due to the flexibility of neural networks in modeling non-linear relationships, they stay one of the powerful models for complex datasets.

Weaknesses: Though neural networks are very good at predictive performance, they demand quite a lot of hyperparameter tuning and involve settings for hidden layers and activation functions as well as learning rates. They are computationally expensive and less interpretable as compared to models like Random Forest.

Best Use Case: The neural network finds its best application when the underlying dataset contains complex nonlinear patterns, for which high predictive accuracy is expected. They are particularly useful in situations where other models fall short to capture the intricacies of the data.

G. Key Insights from the Comparison

Best Overall Model: **Random Forest Regression** model gave the best result because of the highest **R^2 of 0.8458**, with the least errors in all metrics, thus learning nonlinear relationships and returning very informative feature importance rankings effectively.

Best for Simplicity: **Linear Regression** is the most interpretable, simplest model, and outperformed by other, more complex models. It gives a useful baseline against which to compare other results where simplicity and speed are more critical than predictive accuracy.

Handling Non-linear Relationships: Both **Random Forest** and **Gradient Boosting** yielded the best results in the capture of non-linear relationships, with Random Forest performing better compared to Gradient Boosting. **Neural Networks** also captured the non-linearity well but required extensive tuning.

Best for Reducing Error: **Random Forest** had the lowest **MSE (0.0016)** and **RMSE (0.0400)**, to show it was the most efficient model in minimizing prediction errors and giving the best ADRs prediction accuracy.

XI. Conclusion

Among the models, **Random Forest Regression** is the strongest in terms of the trade-off between performance and explainability regarding the prediction of ADR. **Gradient Boosting** and **Neural Network (MLP)** show a very good performance, just slightly worse than Random Forest. **Linear Regression** is a good baseline, while **Support Vector Regression (SVR)** is not able to model the complexity of this dataset. When it comes to the choice of model for the ADR prediction, there would be essential to say that **Random Forest** is the best among them for specific applications wherein predictive accuracy and feature insights are integral. In the next phase we will be using big data tools to handle the data effectively which significantly reduces time and cost of the operation.

For predicting hotel booking cancellations, **Random Forest** is the clear winner, providing superior performance across all metrics. **Decision Tree** also performed well and is a great choice when interpretability is key. For faster, simpler models, **Logistic Regression** offers a solid baseline, while **KNN** and **Naive Bayes** are less effective for this particular dataset. Finally, **SVC** provides good performance but requires more computational resources compared to other models. In the next phase we will be using big data tools to handle the data effectively which significantly reduces time and cost of the operation.

Acknowledgment

We would like to extend our heartfelt gratitude to our project supervisor Shamsad Parveen, and our mentor Juesung Lee, for their invaluable guidance and support throughout this study. Their insights and suggestions played a crucial role in shaping this project. We also appreciate the creators of the dataset on Kaggle for providing such a comprehensive and detailed dataset, which formed the basis of this analysis. Lastly, we acknowledge the various online resources and publications, including the NIST EDA guidelines and John Tukey's principles of exploratory data analysis, which were instrumental in effectively framing the EDA process. This project would not have been possible without the combined support and knowledge from these resources.

REFERENCES

[1] C. O'Neill and R. Schutt. Doing Data Science., O'Reilly. 2013.

[2] NIST on EDA, https://www.itl.nist.gov/div898/handbook/eda/section1/eda11.htm, last viewed February, 2021.

[3] Kaggle Dataset, https://www.kaggle.com/datasets/jessemostipak/hotel-booking-demand

[4] C. M. Bishop, *Pattern Recognition and Machine Learning*, 1st ed., Springer, 2006, pp. 205-210.

[5] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees*, 1st ed., Belmont, CA, USA: Wadsworth, 1984.

[6] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21-27, Jan. 1967, doi: 10.1109/TIT.1967.1053964.

[7] D. Lewis, "Naive (Bayes) at Forty: The Independence Assumption in Information Retrieval," in *Proceedings of the European Conference on Machine Learning (ECML)*, 1998, pp. 4-15.

[8] V. Vapnik, *The Nature of Statistical Learning Theory*, 2nd ed., Springer-Verlag, 1999.

[9] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5-32, 2001, doi: 10.1023/A:1010933404324.

[10] R. A. Fisher, "The Use of Multiple Measurements in Taxonomic Problems," *Annals of Eugenics*, vol. 7, pp. 179-188, 1936.

[11] V. Vapnik, *Statistical Learning Theory*, 1st ed., John Wiley & Sons, 1998.

[12] J. H. Friedman, "Greedy Function Approximation: A Gradient Boosting Machine," *The Annals of Statistics*, vol. 29, no. 5, pp. 1189-1232, 2001, doi: 10.1214/aos/1013203451.

[13] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533-536, 1986, doi: 10.1038/323533a0.

[14] J. H. Friedman, "Stochastic Gradient Boosting," *Computational Statistics & Data Analysis*, vol. 38, no. 4, pp. 367-378, 2002, doi: 10.1016/S0167-9473(01)00065-2.

[15] F. Pedregosa *et al.*, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825-2830, 2011.