SUFFOLK
UNIVERSITY
BOSTON

Student Name: Melody Masunda

Course: Predictive Analytics and Machine Learning (ISOM 835)

Instructor: Professor Hasan Arslan

Date: 5 May 2025

Project Type: Individual Project

Project Title: Predicting Hotel Booking Cancellations Using Machine Learning: A Comparative Analysis of Random Forest and Neural Networks

Final Report

1. Introduction

Hotel booking cancellations can cause substantial disruptions to revenue planning and operational efficiency in the hospitality industry. As a predictive analytics student, I set out to build a machine learning model that can help forecast booking cancellations before they occur. This allows hotels to proactively manage overbooking, reduce revenue loss, and make more informed decisions related to staffing and inventory.

In this project, I applied two predictive models, Random Forest and a Neural Network, to analyze historical booking data and identify the key drivers of cancellations. The project followed the full data science workflow: from data exploration and cleaning, to feature engineering, model training, evaluation, and insight generation. My goal was not only to develop a high-performing model, but also to ensure that the results were interpretable and applicable in real-world hotel operations.

Dataset Description

I used the Hotel Booking Demand dataset, which was obtained from Kaggle. This dataset contains over 119,000 hotel bookings made between 2015 and 2017 across two hotel types: a city hotel and a resort hotel.

Each record includes information on:

- Guest behavior (e.g., repeat guest status, number of special requests)
- Booking specifics (e.g., lead time, number of nights, number of guests)
- Market and channel details (e.g., distribution channel, market segment, deposit type)
- Financial variables (e.g., average daily rate (ADR))
- The target variable: is_canceled (1 = canceled, 0 = not canceled)

Before training the models, I performed data cleaning by handling missing values, detecting and evaluating outliers, and applying appropriate feature transformations (e.g., calculating total nights). I used one-hot encoding to convert categorical variables and scaled the numerical features to prepare the data for machine learning.

2. Exploratory Data Analysis (EDA)

To better understand the structure and behavior of the dataset before modeling, I conducted a thorough exploratory data analysis using summary statistics, data types, distributions, and visualizations for both numerical and categorical variables.

Dataset Structure and Overview

Using basic inspection methods such as df.info(), df.shape, and df.describe(), I confirmed that the dataset contains 119,390 rows and 32 columns, including a mix of numerical and

categorical variables. The target variable **is_canceled** is binary and indicates whether a booking was canceled (1) or not (0).

- Data Types: The dataset includes both int64, float64, and object types.
- Index and Columns: Columns were retrieved and stored as a list for easy reference.
- Summary Statistics: df.describe() was used for numerical summaries, and df.describe(include=['object']) was used to explore categorical values (e.g., hotel type, meal, deposit type).

Missing Values
Using df.isnull().sum(), I identified missing values in the following columns:

- Children
- Country
- Agent
- company

These were addressed in the data cleaning phase through imputation using appropriate domain-informed strategies (e.g., filling children with 0 and country with the mode).

Visualizations

Histograms for Numerical Features
I plotted histograms to examine the distribution of numerical features such as lead_time, adr (average daily rate), booking_changes, days_in_waiting_list and total_of_special_requests

Insights:

- **‘lead_time' is heavily right-skewed:** Most bookings have short lead times, but some have very long lead times.
- **'stays_in_weekend_nights' and 'stays_in_week_nights':** The majority of stays involve a few nights, with peaks around 0-2 nights for weekends and 1-3 nights for weekdays.
- **'adr' (average daily rate):** Has a slightly skewed distribution, indicating a range of room prices with some potential outliers on the higher end.
- **'adults', 'children', 'babies':** Most bookings involve 2 adults, with fewer bookings involving children or babies.
- **'previous_cancellations':** Most bookings have 0 previous cancellations, but there are some with 1 or more cancellations.

Bar Charts for Categorical Features
I used bar plots to visualize distributions of key categorical variables, including hotel, deposit_type, market_segment and is_repeated_guest.

Insights:

- **Hotel Type:** Most bookings are for City Hotels, followed by Resort Hotels.
- **Arrival Date:** Bookings are distributed across all months, with peaks observed during the summer months (July and August) and a slight increase in April and May as well as October. There is a dip in November.
- **Customer Type:** The majority of bookings are made by Transient customers (individuals), followed by Transient-Party. Contract and Group bookings are less common.
- **Meal Type:** Most bookings include Bed & Breakfast (BB), followed by Half board (HB). Full board (FB) and Undefined meals are less frequent.
- **Country:** Portugal is the top country of origin for bookings, followed by Great Britain and France. Other countries have significantly fewer bookings.
- **Market Segment:** Most bookings are made through the Online TA (Travel Agents) segment, followed by Offline TA/TO and Groups. Direct and Corporate bookings are less common.
- **Distribution Channel:** The majority of bookings are made through the TA/TO (Travel Agents/Tour Operators) channel.
- **Reserved Room Type:** Room type A is the most frequently booked, followed by Room type D. Other room types have lower booking frequencies.
- **Assigned Room Type:** Similar to reserved room type, Room type A is the most commonly assigned, followed by Room type D. There are instances where the assigned room type differs from the reserved room type, indicating potential room upgrades or changes.
- **Deposit Type:** Most bookings are made with No Deposit, followed by Non Refund. Refundable deposits are less common.
- **Reservation Status:** The majority of bookings are either Canceled or Check-Out.

3. Preprocessing

Before training predictive models, I performed several data preprocessing steps to ensure the dataset was clean, consistent, and ready for machine learning. These steps included handling missing values, feature engineering, removing irrelevant or redundant columns, encoding categorical variables, and scaling numerical features.

Missing Values Handling

The dataset was generally clean but required minor imputation. Missing values were identified in the following columns: children, country, agent, and company.

- children was filled with 0, assuming no children were present when the value was missing.
- country was filled with the most frequent value (mode).
- agent and company contained substantial missing values and were dropped entirely from the dataset due to their low predictive value and high sparsity.

Feature Engineering and Reduction
I created a new feature, **total_nights,** by summing **stays_in_week_nights** and **stays_in_weekend_nights**, capturing the total length of stay. The original two columns were then dropped to simplify the dataset, in alignment with the Principle of Parsimony.

Additionally, the columns **reservation_status** and **reservation_status_date** were removed to prevent data leakage, as they represent information only available after a booking has been completed or canceled.

Encoding Categorical Variables
Categorical features were converted into numerical format using one-hot encoding, with drop_first=True to avoid multicollinearity. This transformation enabled compatibility with machine learning models while maintaining a clean and compact feature set.

Feature Scaling
Numerical features were standardized using StandardScaler to bring them onto a common scale. This was especially important for the Neural Network model, which is sensitive to feature magnitude. Scaling was applied after the train-test split to prevent data leakage from test data into the training process.

These preprocessing steps ensured that the dataset was free from inconsistencies, simplified through intelligent feature engineering, and optimized for supervised learning models. By reducing dimensionality and standardizing inputs, I created a clean, interpretable, and high-performing dataset for machine learning.

4. Business Questions

- **Question 1:** What are the primary factors driving booking cancellations?

Explanation: Understanding why cancellations occur is crucial for revenue management and operational planning. By analyzing factors which influence booking cancellations hotel managers can identify high-risk bookings and potentially implement targeted strategies to reduce cancellations and improve occupancy forecasting.

- **Question 2:** Do repeat guests exhibit significantly different booking behaviors (e.g., lead time, length of stay, special requests) or generate higher ADR compared to new guests?

Explanation: This helps Marketing and Loyalty Program Managers to assess the value and characteristics of loyal customers. Understanding if repeat guests book earlier/later (lead_time), stay longer (stays_in_weekend_nights + stays_in_week_nights), require more services (total_of_special_requests), or pay different rates (adr) can inform strategies for customer retention, targeted marketing campaigns, and evaluating the success of loyalty initiatives.

- **Question 3:** Can we accurately predict whether a booking will be canceled in advance using machine learning models?

Explanation: By training models like Random Forest and Neural Networks, we aim to develop a reliable early warning system for potential cancellations, enabling the hotel to reduce lost revenue and improve occupancy forecasting.

5. Modeling

To predict hotel booking cancellations, I developed and evaluated two supervised learning models: Random Forest Classifier and a Neural Network (MLPClassifier). Both models were trained using the top 15 most important features, selected based on feature importance from the initial Random Forest. This approach followed the Principle of Parsimony, ensuring simplicity without sacrificing performance.

Model evaluation focused on recall for cancellations (class 1), in order to minimize missed cancellations and support better overbooking and staffing strategies.

Model Performance Comparison Table

Metric	Random Forest (Top 15 Features)	Neural Network (MLP)
Accuracy	0.89	0.86
Recall (Canceled)	**0.81**	0.77
Precision (Canceled)	0.89	0.84
F1-Score (Canceled)	0.85	0.80
AUC Score	**0.958**	0.932
False Negatives	**1,710**	2,095

The Random Forest model with 15 features outperformed the Neural Network in key metrics related to cancellation prediction, especially recall and AUC. While both models showed strong performance, the Random Forest offered better reliability in identifying cancellations, fewer false negatives, and greater interpretability, making it the preferred model for operational decision-making in a hotel context.

6. Insights

This project set out to analyze hotel booking behavior and develop machine learning models capable of predicting cancellations. By focusing on high-impact features and evaluating both Random Forest and Neural Network classifiers, I derived actionable insights for hotel management, marketing, and operations.

Key Insights from Modeling

The best-performing model, a Random Forest trained on the top 15 features, achieved:

- Accuracy: 89%
- Recall for canceled bookings: 0.81
- AUC score: 0.958

These metrics indicate that the model is highly effective at identifying bookings that are likely to be canceled. It correctly flagged over 81% of actual cancellations, making it suitable as an early warning tool for hotel operations.

The Neural Network model also performed well, with an AUC of 0.932 and a recall of 0.77 for cancellations. However, the Random Forest had a lower false negative rate and greater interpretability, making it the preferred model in this case.

Answers to Business Questions

a. What are the primary factors driving booking cancellations?

The most influential predictors included lead time, deposit type, booking source (agent/distribution channel), customer type, and market segment. Bookings with long lead times, no deposit, and made through online travel agents were significantly more likely to be canceled. These findings can inform targeted overbooking policies, stricter cancellation rules, or customer-specific engagement strategies.

b. Do repeat guests behave differently than new guests?

Yes. Repeat guests tend to book with shorter lead times, make more special requests, and cancel less frequently. They also show slightly higher ADR (average daily rate), suggesting that they are more valuable and reliable. This supports further investment in loyalty programs, personalized experiences, and exclusive offers.

c. Can we predict cancellations accurately in advance?

Yes. The Random Forest model's strong recall and AUC indicate that cancellations can be predicted with a high degree of accuracy. This capability enables proactive decision-making across revenue management, marketing, and front-desk operations.

Implications for Hotel Decision-Making

Based on the model's outputs and feature insights, hotels can:

- Apply predictive scores in real-time to flag high-risk bookings
- Proactively contact guests likely to cancel for confirmations or incentives
- Set overbooking thresholds dynamically, based on cancellation risk
- Refine loyalty and marketing strategies, focusing on the behavior of repeat vs. new guests
- Review deposit policies for segments that cancel frequently

Limitations and Considerations

- **Temporal Bias:** The dataset spans 2015–2017; cancellation patterns may differ today, especially post-pandemic.

- **Ethical Concerns:** Over-reliance on cancellation predictions may unfairly target certain countries, booking agents, or market segments. Regular bias audits are recommended.
- **Model Complexity:** While Random Forests offer feature interpretability, the Neural Network operates as a black box and may be less explainable to non-technical stakeholders.
- **Business Integration:** The success of predictive analytics depends not just on accuracy, but on whether the hotel has systems and processes ready to act on predictions.
- **Data scope:** The model is only as good as the data available. If guest sentiment, economic trends, or pandemic-related variables are missing, the prediction might be limited.

By combining machine learning with domain understanding, this project demonstrates that booking cancellations can be anticipated and managed more strategically. The insights generated provide a foundation for smarter forecasting, revenue protection, and enhanced guest experience, aligning predictive analytics with operational impact.

7. Ethics & Interpretability

While predictive models like Random Forests and Neural Networks offer powerful tools for forecasting booking cancellations, their use raises several ethical considerations. One key concern is potential bias in the data. For example, if certain market segments, countries, or booking channels historically show higher cancellation rates due to external factors (e.g., economic conditions, travel restrictions), the model may unintentionally flag bookings from those groups as high risk, potentially leading to unfair treatment such as stricter booking terms or limited availability.

To minimize this risk, it's important to regularly audit model predictions across demographic and booking source features to ensure that no group is disproportionately penalized. Moreover, relying solely on cancellation likelihood for business decisions must be balanced with customer experience considerations.

In terms of interpretability, the Random Forest model provides relatively transparent insights. Feature importance scores make it possible to explain why a particular booking was flagged as high risk. This supports accountability and builds trust with stakeholders such as revenue managers, front office teams, and marketing departments.

However, the Neural Network model, while effective, functions more like a "black box." It is less interpretable, making it harder to justify individual predictions. For real-world deployment, preference should be given to models that balance performance with transparency, especially in customer-facing decisions.

Appendix (code, visuals)

Code Snippets

Checking missing values

```
# Count missing values in each column

missing = df.isnull().sum()

# Filter only columns with missing values
missing = missing[missing > 0].sort_values(ascending=False)

# Print result
print("📌 Columns with Missing Values:\n")
print(missing)
```

Handling Missing Values

df['children'] = df['children'].fillna(0)
df['country'] = df['country'].fillna(df['country'].mode()[0])
df.drop(['agent', 'company', 'reservation_status', 'reservation_status_date'], axis=1, inplace=True)

Feature Engineering

df['total_nights'] = df['stays_in_week_nights'] + df['stays_in_weekend_nights']
df.drop(['stays_in_week_nights', 'stays_in_weekend_nights'], axis=1, inplace=True)

One-Hot Encoding and Scaling

df_encoded = pd.get_dummies(df, drop_first=True)
scaler = StandardScaler()
X_scaled = scaler.fit_transform(df_encoded.drop('is_canceled', axis=1))

Checking Outliers in Numerical Columns

```
numeric_cols = df.select_dtypes(include=['int64', 'float64']).columns

# Create boxplots for all numeric columns (excluding is_canceled)
for col in numeric_cols:
    if col != 'is_canceled':
        plt.figure(figsize=(8, 3))
        sns.boxplot(data=df, x=col)
        plt.title(f'Boxplot of {col}')
        plt.show()
```

```
# Loop through each numeric column and compute IQR-based outliers
numeric_cols = df.select_dtypes(include=['int64', 'float64']).columns
numeric_cols = df.select_dtypes(include=['int64',
'float64']).columns.tolist()
numeric_cols.remove('is_canceled')
for feature in numeric_cols:
    Q1 = df[feature].quantile(0.25)
    Q3 = df[feature].quantile(0.75)
    IQR = Q3 - Q1

    # Define bounds
    upper_bound = Q3 + 1.5 * IQR
    lower_bound = Q1 - 1.5 * IQR

    # Find outliers
    outliers = df[(df[feature] > upper_bound) | (df[feature] <
lower_bound)]

    print(f"Feature: {feature}")
    print(f"Number of Outliers: {len(outliers)}")
    print(f"Percentage of Outliers: {len(outliers) / len(df) * 100:.2f}%")
    print("-" * 40)
```

Random Forest Model

```
# 1. Extract feature importances from previous full-feature Random Forest
importances = rf.feature_importances_
features = X_train.columns

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, roc_auc_score,
confusion_matrix, ConfusionMatrixDisplay

# Create and sort feature importance DataFrame
feature_importance_df = pd.DataFrame({'Feature': features, 'Importance':
importances})
feature_importance_df = feature_importance_df.sort_values(by='Importance',
ascending=False)

# Plot top 20 for reference
```

```
plt.figure(figsize=(10, 6))
sns.barplot(data=feature_importance_df.head(20), x='Importance',
y='Feature')
plt.title("Top 20 Feature Importances")
plt.tight_layout()
plt.show()

# 2. Get top 15 features
top_features = feature_importance_df['Feature'].head(15).tolist()
top_indices = [X_train.columns.get_loc(col) for col in top_features]

# 3. Subset scaled feature sets
X_train_top = X_train_scaled[:, top_indices]
X_test_top = X_test_scaled[:, top_indices]

# 4. Train Random Forest on top 15 features
rf_top15 = RandomForestClassifier(random_state=42)
rf_top15.fit(X_train_top, y_train)

# 5. Predict and evaluate
top15_preds = rf_top15.predict(X_test_top)
top15_probs = rf_top15.predict_proba(X_test_top)[:, 1]

# Print classification report and AUC
print("🔍 Random Forest (Top 15 Features):")
print(classification_report(y_test, top15_preds))
print("AUC Score:", roc_auc_score(y_test, top15_probs))

# Confusion Matrix
cm_top15 = confusion_matrix(y_test, top15_preds)
disp = ConfusionMatrixDisplay(confusion_matrix=cm_top15,
display_labels=['Not Canceled', 'Canceled'])
disp.plot(cmap='Greens')
plt.title("Random Forest - Confusion Matrix (Top 15 Features)")
plt.show()
```

Random Forest Regularization and Performance Enhancement

```
#Grid Search to Maximize Recall for Cancellations
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import make_scorer, recall_score

# Define a custom scorer for recall of class 1 (canceled bookings)
recall_canceled = make_scorer(recall_score, pos_label=1)
```

```
# Define hyperparameter grid
param_grid = {
    'n_estimators': [100, 200],
    'max_depth': [10, 15, 20],
    'min_samples_split': [5, 10],
    'max_features': ['sqrt', 'log2']
}

# Initialize GridSearchCV
grid_search = GridSearchCV(
    estimator=RandomForestClassifier(random_state=42),
    param_grid=param_grid,
    scoring=recall_canceled,
    cv=3,
    verbose=2,
    n_jobs=-1
)

# Fit grid search on the top-15-feature dataset
grid_search.fit(X_train_top, y_train)

# Get best model
best_rf = grid_search.best_estimator_

# Predict and evaluate
best_preds = best_rf.predict(X_test_top)
best_probs = best_rf.predict_proba(X_test_top)[:, 1]

from sklearn.metrics import classification_report, roc_auc_score,
confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt

print("🔍 Tuned Random Forest (Max Recall for Cancellations):")
print(classification_report(y_test, best_preds))
print("AUC Score:", roc_auc_score(y_test, best_probs))

# Confusion Matrix
cm_best = confusion_matrix(y_test, best_preds)
disp = ConfusionMatrixDisplay(confusion_matrix=cm_best,
display_labels=['Not Canceled', 'Canceled'])
disp.plot(cmap='Oranges')
plt.title("Random Forest - Confusion Matrix (Tuned for Recall)")
plt.show()
```

Train/Test Split

```
from sklearn.model_selection import train_test_split

# Separate features and target
X = df_encoded.drop('is_canceled', axis=1)
y = df_encoded['is_canceled']

# Train/test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
```

Scale Features for Neural Network

```
from sklearn.preprocessing import StandardScaler

# Initialize the scaler
scaler = StandardScaler()

# Fit on training data, transform both train and test
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

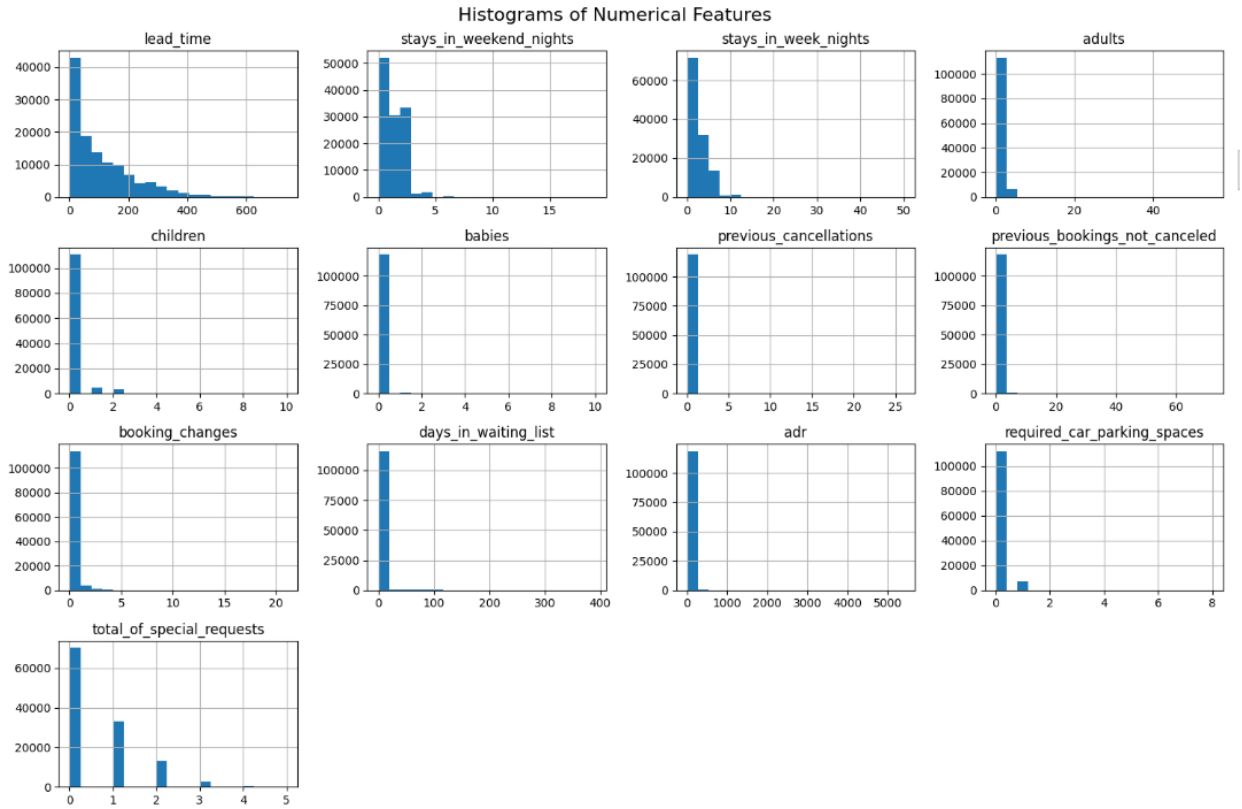
Neural Network

```
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import classification_report, roc_auc_score,
confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt

# Initialize and train the Neural Network
mlp = MLPClassifier(hidden_layer_sizes=(100,), max_iter=500,
random_state=42)
mlp.fit(X_train_top, y_train)

# Make predictions
mlp_preds = mlp.predict(X_test_top)
mlp_probs = mlp.predict_proba(X_test_top)[:, 1]  # Probabilities for AUC

# Evaluate model
print("Neural Network (MLP) Classification Report:")
print(classification_report(y_test, mlp_preds))
print("AUC Score:", roc_auc_score(y_test, mlp_probs))

# Confusion matrix
```

```
mlp_cm = confusion_matrix(y_test, mlp_preds)
disp = ConfusionMatrixDisplay(confusion_matrix=mlp_cm,
display_labels=['Not Canceled', 'Canceled'])
disp.plot(cmap='Purples')
plt.title("Neural Network - Confusion Matrix")
plt.show()
```

Histograms for Numerical Features

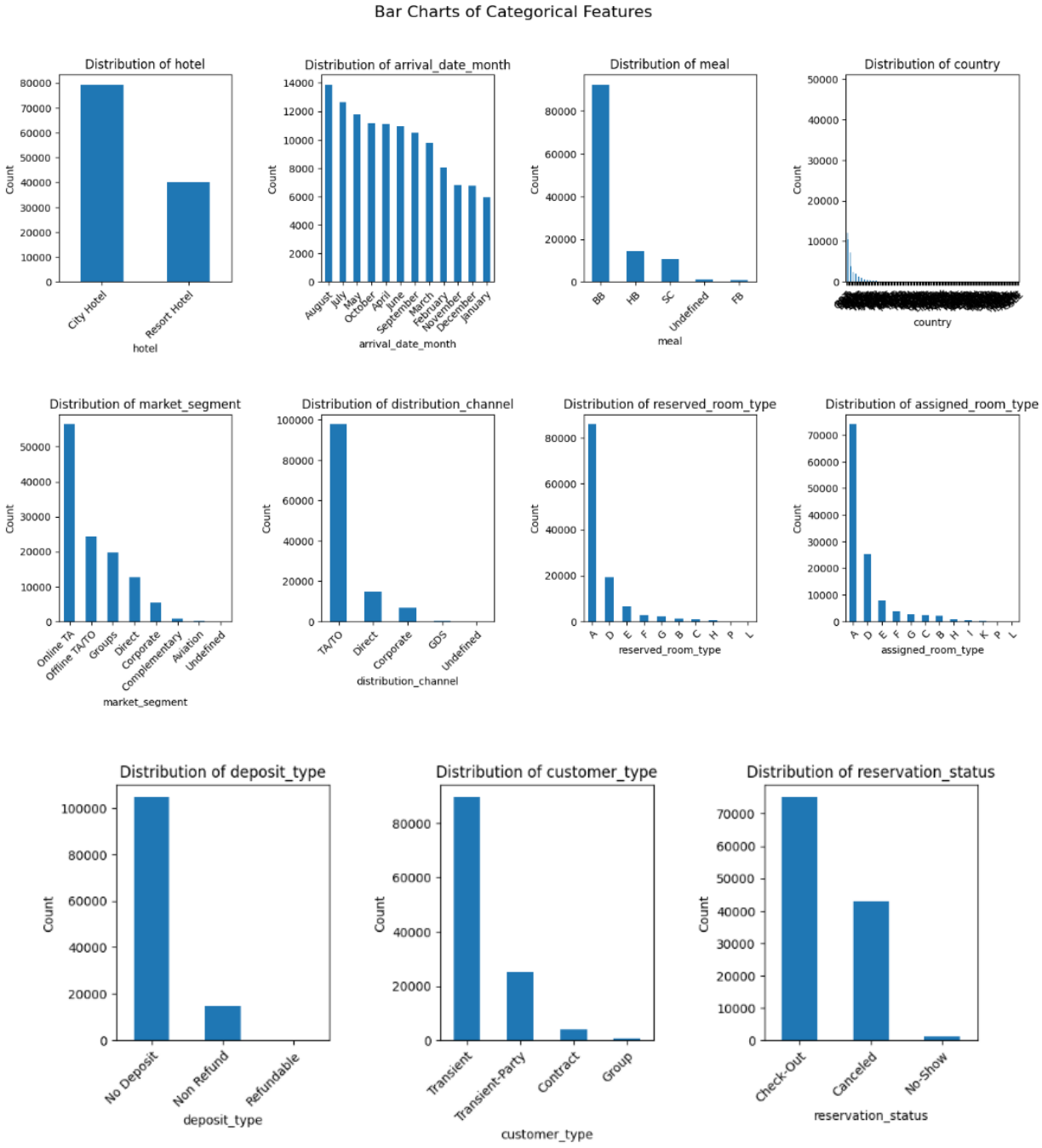
```
# Create histograms
df[numerical_features].hist(figsize=(15, 10), bins=20)
plt.suptitle('Histograms of Numerical Features', fontsize=16)
plt.tight_layout()
plt.show()
```



Bar Charts for Categorical features

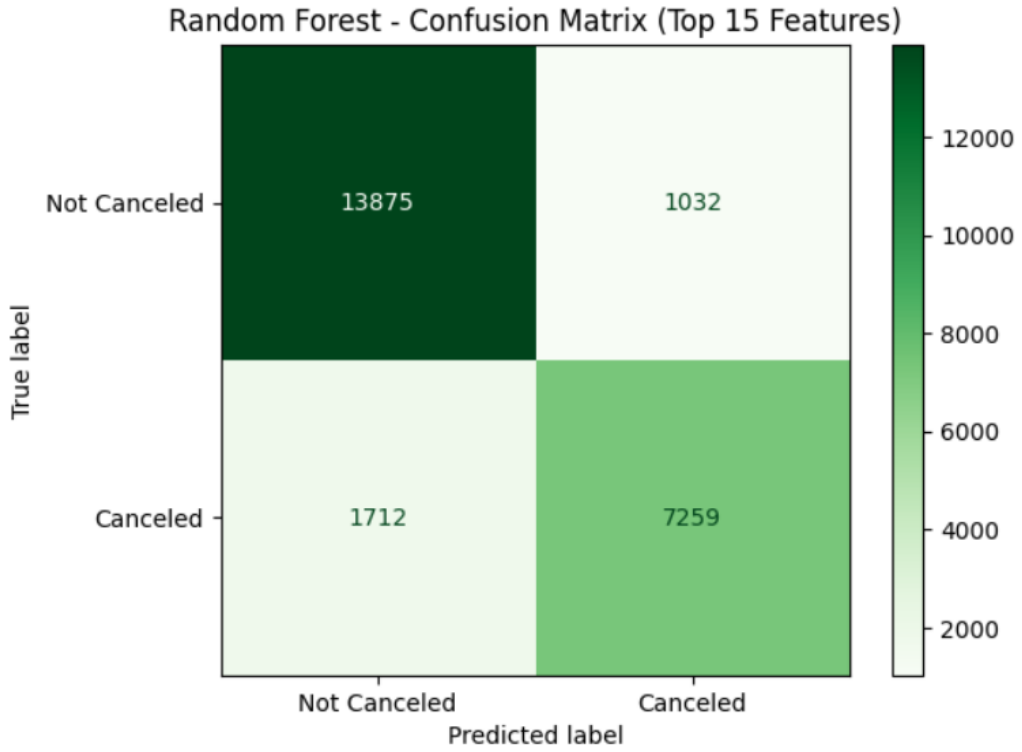
```
fig, axes = plt.subplots(num_rows, num_cols, figsize=(15, num_rows *
5))  # Adjust figsize
fig.suptitle('Bar Charts of Categorical Features', fontsize=16)

# Iterate through categorical features and create bar charts
```

```
for i, feature in enumerate(categorical_features):
    row = i // num_cols
    col = i % num_cols

    # Handle single row case
    ax = axes[row, col] if num_rows > 1 else axes[col]

    #The following 2 lines ensures you're drawing on the correct plot.
    df[feature].value_counts().plot(kind='bar', ax=ax)  # Bar chart
    plt.sca(ax)

    ax.set_title(f'Distribution of {feature}')
    ax.set_xlabel(feature)
    ax.set_ylabel('Count')

    # Rotate x-axis labels if needed
    plt.setp(ax.get_xticklabels(), rotation=45, ha="right",
rotation_mode="anchor")

# Adjust layout to prevent overlapping
plt.tight_layout(pad=3.0)  # Adjust pad for spacing
plt.show()
```


Bar Charts of Categorical Features
Distribution of hotel
Count
hotel
City Hotel
Resort Hotel
Distribution of arrival_date_month
arrival_date_month
August
July
May
October
April
June
September
March
February
November
December
January
Distribution of meal
meal
BB
HB
SC
Undefined
FB
Distribution of country
country
Distribution of market_segment
market_segment
Online TA
Offline TA/TO
Groups
Direct
Corporate
Complementary
Aviation
Undefined
Distribution of distribution_channel
distribution_channel
TA/TO
Direct
Corporate
GDS
Undefined
Distribution of reserved_room_type
reserved_room_type
Distribution of assigned_room_type
assigned_room_type
Distribution of deposit_type
deposit_type
No Deposit
Non Refund
Refundable
Distribution of customer_type
customer_type
Transient
Transient-Party
Contract
Group
Distribution of reservation_status
reservation_status
Check-Out
Canceled
No-Show

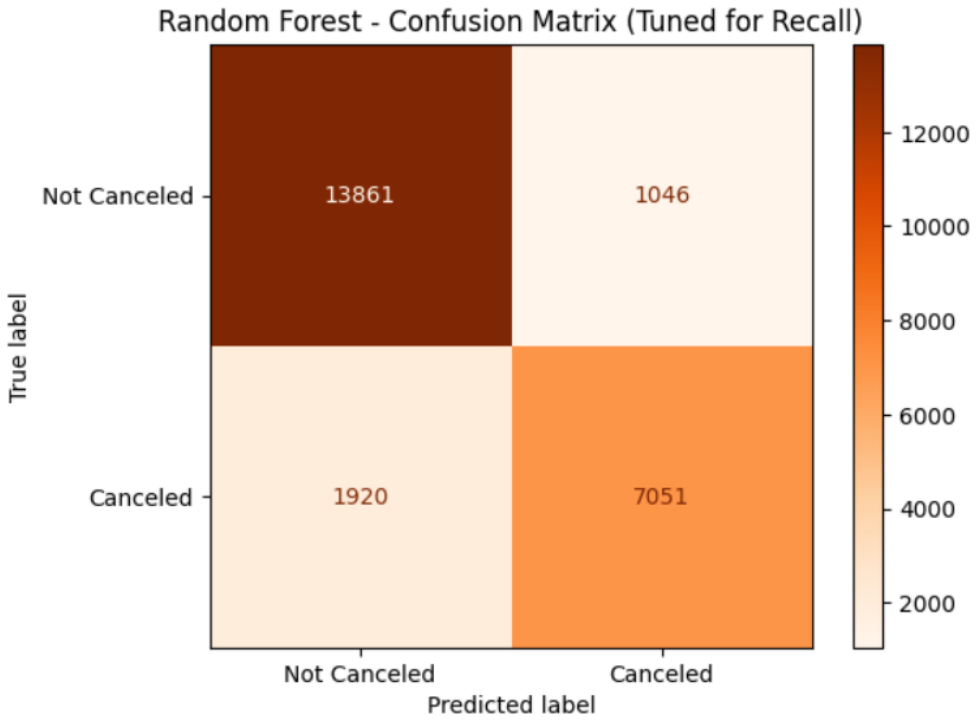
	precision	recall	f1-score	support
0	0.89	0.93	0.91	14907
1	0.88	0.81	0.84	8971
accuracy			0.89	23878
macro avg	0.88	0.87	0.88	23878
weighted avg	0.88	0.89	0.88	23878

AUC Score: 0.9523338123333044



	precision	recall	f1-score	support
0	0.88	0.93	0.90	14907
1	0.87	0.79	0.83	8971
accuracy			0.88	23878
macro avg	0.87	0.86	0.86	23878
weighted avg	0.88	0.88	0.87	23878

AUC Score: 0.9495501433003075



	precision	recall	f1-score	support
0	0.87	0.91	0.89	14907
1	0.84	0.77	0.80	8971
accuracy			0.86	23878
macro avg	0.85	0.84	0.85	23878
weighted avg	0.86	0.86	0.86	23878

AUC Score: 0.9321945170150425