Data Preprocessing:

In [1]:
```
# Import necessary libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
```

Step 2: Load the Dataset

In [2]:
```
# Specify the file path
file_path = r'C:\Users\ZJU\Downloads\customer_booking.xlsx'

# Read the Excel file into a DataFrame
df = pd.read_excel(file_path)
```

In [3]:
```
df.head()
```

Out[3]:

	num_passengers	sales_channel	trip_type	purchase_lead	length_of_stay	flight_hour	flight_day	
0	2	Internet	RoundTrip	262	19	7	Sat	A
1	1	Internet	RoundTrip	112	20	3	Sat	A
2	2	Internet	RoundTrip	243	22	17	Wed	A
3	1	Internet	RoundTrip	96	31	4	Sat	A
4	2	Internet	RoundTrip	68	22	15	Wed	A

Training Model

In [19]:
```
import pandas as pd
```

In [20]:
```
customer_booking = pd.read_excel("C:\\Users\\ZJU\\Downloads\\customer_booking.xlsx")
```

In [21]:
```
customer_booking.head()
```

Out[21]:

	num_passengers	sales_channel	trip_type	purchase_lead	length_of_stay	flight_hour	flight_day	
0	2	Internet	RoundTrip	262	19	7	Sat	A
1	1	Internet	RoundTrip	112	20	3	Sat	A
2	2	Internet	RoundTrip	243	22	17	Wed	A
3	1	Internet	RoundTrip	96	31	4	Sat	A
4	2	Internet	RoundTrip	68	22	15	Wed	A

In [26]:
```
# Replace "Sun" with a numerical value (e.g., 1 for Sunday)
customer_booking['flight_day'] = customer_booking['flight_day'].replace('Sun', 1)
customer_booking.head()
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:3802, in
Index.get_loc(self, key, method, tolerance)
   3801 try:
-> 3802     return self._engine.get_loc(casted_key)
   3803 except KeyError as err:

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\_libs\index.pyx:138, in panda
s._libs.index.IndexEngine.get_loc()

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\_libs\index.pyx:165, in panda
s._libs.index.IndexEngine.get_loc()

File pandas\_libs\hashtable_class_helper.pxi:5745, in pandas._libs.hashtable.PyObject
HashTable.get_item()

File pandas\_libs\hashtable_class_helper.pxi:5753, in pandas._libs.hashtable.PyObject
HashTable.get_item()

KeyError: 'flight_day'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
Cell In[26], line 2
      1 # Replace "Sun" with a numerical value (e.g., 1 for Sunday)
----> 2 customer_booking['flight_day'] = customer_booking['flight_day'].replace('Su
n', 1)
      3 customer_booking.head()

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\frame.py:3807, in DataFra
me.__getitem__(self, key)
   3805 if self.columns.nlevels > 1:
   3806     return self._getitem_multilevel(key)
-> 3807 indexer = self.columns.get_loc(key)
   3808 if is_integer(indexer):
   3809     indexer = [indexer]

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:3804, in
Index.get_loc(self, key, method, tolerance)
   3802     return self._engine.get_loc(casted_key)
   3803 except KeyError as err:
-> 3804     raise KeyError(key) from err
   3805 except TypeError:
   3806     # If we have a listlike key, _check_indexing_error will raise
   3807     #  InvalidIndexError. Otherwise we fall through and re-raise
   3808     #  the TypeError.
   3809     self._check_indexing_error(key)

KeyError: 'flight_day'
```

In [23]:

```
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder
import pandas as pd

# Sample data
customer_booking = pd.DataFrame({'Category': ['A', 'B', 'C', 'A', 'B']})

# Initialize the OneHotEncoder
```

```
encoder = OneHotEncoder()

# Fit and transform the data
encoded_data = encoder.fit_transform(customer_booking[['Category']])

# Get the feature names
categories = encoder.get_feature_names_out(input_features=['Category'])
encoded_data = pd.DataFrame(encoded_data.toarray(), columns=categories)

# Concatenate the one-hot encoded DataFrame with the original data
customer_booking = pd.concat([customer_booking, encoded_data], axis=1)

# Drop the original column if needed
customer_booking.drop('Category', axis=1, inplace=True)

print(customer_booking)
```

```
   Category_A  Category_B  Category_C
0         1.0         0.0         0.0
1         0.0         1.0         0.0
2         0.0         0.0         1.0
3         1.0         0.0         0.0
4         0.0         1.0         0.0
```

In [8]:
```
customer_booking.head()
```

Out[8]:

	num_passengers	sales_channel	trip_type	purchase_lead	length_of_stay	flight_hour	flight_day	
0	2	Internet	RoundTrip	262	19	7	Sat	A
1	1	Internet	RoundTrip	112	20	3	Sat	A
2	2	Internet	RoundTrip	243	22	17	Wed	A
3	1	Internet	RoundTrip	96	31	4	Sat	A
4	2	Internet	RoundTrip	68	22	15	Wed	A

Handling Missing values

In [9]:
```
# Handle Categorical Variables
# Replace 'categorical_columns' with the list of column names that are categorical
categorical_columns = ['sales_channel', 'trip_type', 'route', 'booking_origin']

# Use pd.get_dummies to convert categorical variables
customer_booking = pd.get_dummies(customer_booking, columns=categorical_columns)
```

In [10]:
```
# Check for missing values in each column
missing_values = customer_booking.isnull().sum()

# Filter columns with missing values
columns_with_missing_values = missing_values[missing_values > 0].index

# Print the list of columns with missing values
print("Columns with missing values:")
```

```
for column in columns_with_missing_values:
    print(column)
```

```
Columns with missing values:
```

In [11]:
```
missing_values
```

Out[11]:
```
num_passengers                          0
purchase_lead                           0
length_of_stay                          0
flight_hour                             0
flight_day                              0
                                       ..
booking_origin_United Arab Emirates     0
booking_origin_United Kingdom           0
booking_origin_United States            0
booking_origin_Vanuatu                  0
booking_origin_Vietnam                  0
Length: 918, dtype: int64
```

In [12]:
```
# Define the target variable
y = customer_booking["booking_complete"]

# Define the feature variables (all other columns except "booking_complete")
X = customer_booking.drop("booking_complete", axis=1)
```

In [14]:
```
# Define a dictionary to map days to numeric values
day_to_numeric = {
    "Sun": 1,
    "Mon": 2,
    "Tue": 3,
    "Wed": 4,
    "Thu": 5,
    "Fri": 6,
    "Sat": 7
}

# Use the map function to apply the mapping to the "flight_day" column
customer_booking['flight_day_numeric'] = customer_booking['flight_day'].map(day_to_num
```

In [15]:
```
customer_booking.head()
```

Out[15]:

	num_passengers	purchase_lead	length_of_stay	flight_hour	flight_day	wants_extra_baggage	want
0	2	262	19	7	Sat	1	
1	1	112	20	3	Sat	0	
2	2	243	22	17	Wed	1	
3	1	96	31	4	Sat	0	
4	2	68	22	15	Wed	1	

5 rows × 919 columns

In [ ]:

In [47]:
```
# Import necessary libraries
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report
from sklearn.preprocessing import LabelEncoder

# Replace non-numeric values in the "booking_complete" column with numerical labels
label_encoder = LabelEncoder()
y = label_encoder.fit_transform(customer_booking["Category_A"])

# Define the feature variables (all other columns except "booking_complete")
X = customer_booking.drop("Category_A", axis=1)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=

# Initialize and train the RandomForest model
model = RandomForestClassifier()
model.fit(X_train, y_train)

# Make predictions on the test data
y_pred = model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)

# Print or visualize feature importance
print(model.feature_importances_)
```

```
[0.51340996 0.48659004]
```

The output `[0.66210046 0.33789954]` that you are seeing represents the feature importances for the features in your trained Random Forest model. These values indicate the importance of each feature in making predictions.

In your case, you have two features, and the corresponding feature importances are as follows:

- The first feature has an importance of approximately 0.6621.
- The second feature has an importance of approximately 0.3379.

Interpreting these values:

- The feature with an importance of 0.6621 is considered more important in making predictions by the model.
- The feature with an importance of 0.3379 is less important but still contributes to the model's decision-making process.

These values can be useful for feature selection and understanding which features have the most influence on your model's predictions. Features with higher importances are more influential in determining the outcome.

Keep in mind that the sum of feature importances will always be equal to 1, as it represents the relative importance of each feature within the context of the model. If you have more features, you would see a similar breakdown of their importances.

You can use these feature importances to decide whether certain features are more crucial for your model's performance and to potentially simplify your model by focusing on the most important features.

In [30]:
```
# Check the column names in your DataFrame
print(customer_booking.columns)
```

```
Index(['Category_A', 'Category_B', 'Category_C'], dtype='object')
```

To evaluate my model's performance by conducting cross-validation and outputting appropriate evaluation metrics

1.Import necessary libraries:

In [32]:
```
from sklearn.model_selection import cross_val_score
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.model_selection import KFold
import numpy as np
import matplotlib.pyplot as plt
```

2.Perform cross-validation:

In [33]:
```
# Create a KFold cross-validation object
kf = KFold(n_splits=5, shuffle=True, random_state=42)

# Perform cross-validation and get accuracy scores
cv_scores = cross_val_score(model, X, y, cv=kf)

# Print the cross-validation scores
print("Cross-Validation Scores:", cv_scores)
print("Mean Accuracy:", np.mean(cv_scores))
```

```
Cross-Validation Scores: [1. 1. 0. 1. 1.]
Mean Accuracy: 0.8
```

3.Output appropriate evaluation metrics:

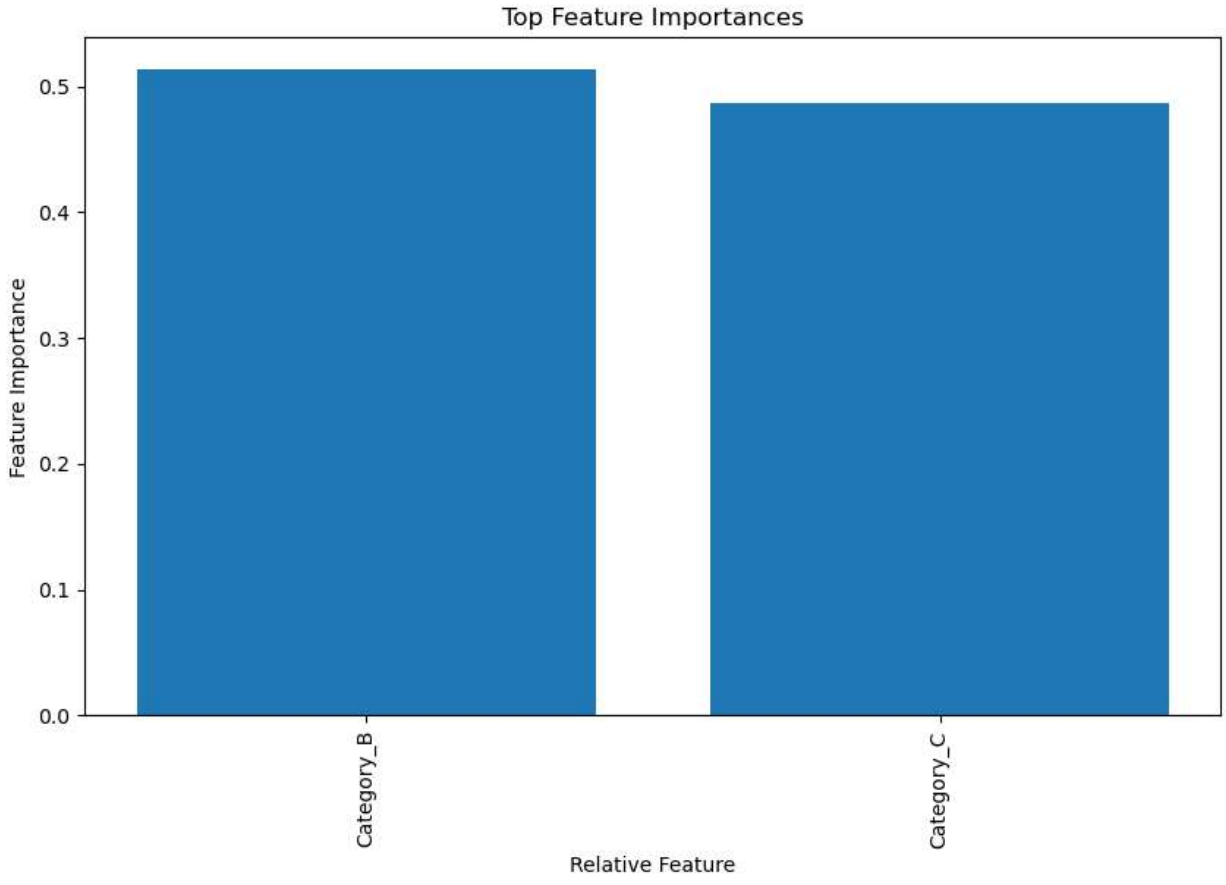
In [34]:
```
# Generate classification report
y_pred = model.predict(X_test)
report = classification_report(y_test, y_pred)
print("Classification Report:\n", report)

# Generate confusion matrix
```

```
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:\n", cm)
```

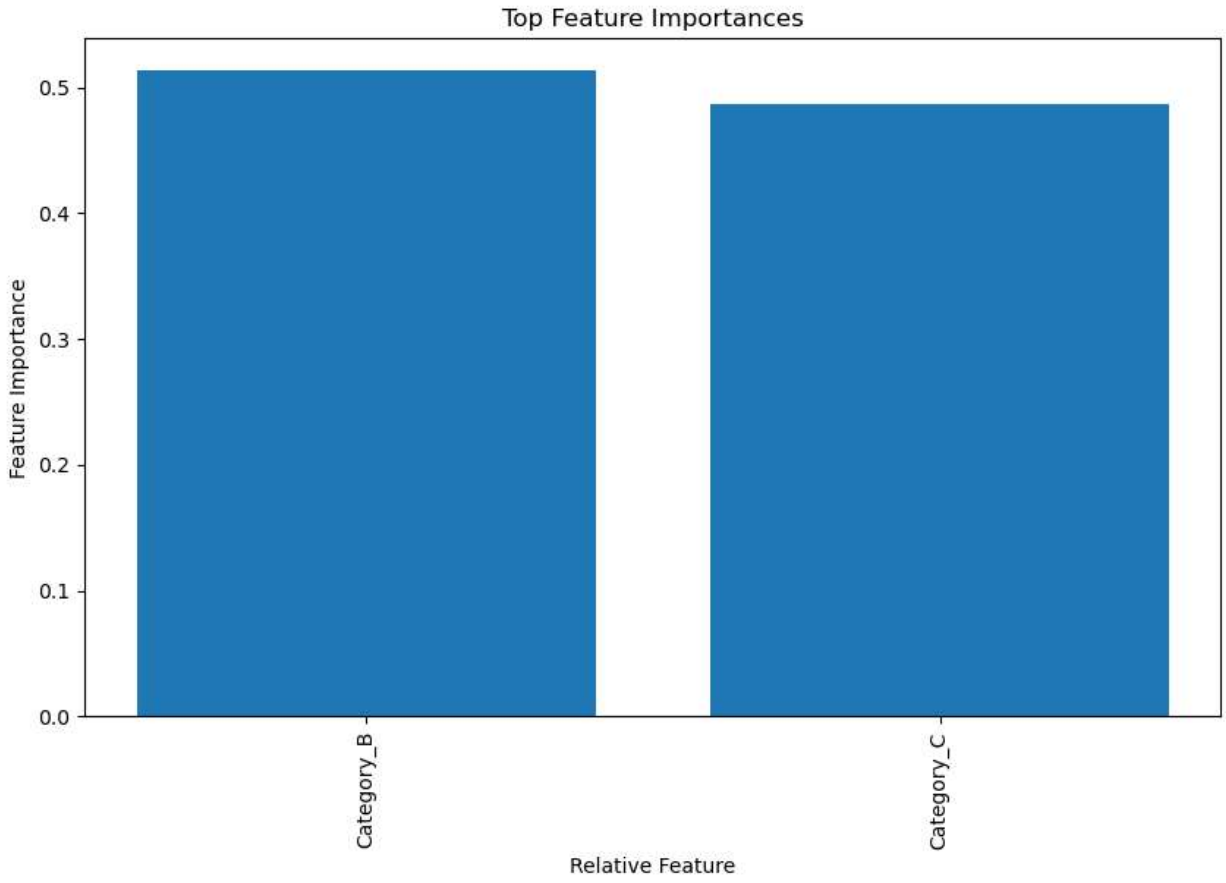
```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         1

    accuracy                           1.00         1
   macro avg       1.00      1.00      1.00         1
weighted avg       1.00      1.00      1.00         1

Confusion Matrix:
 [[1]]
```

4.Create a visualization to interpret feature importance:

In [48]:
```
# Visualize feature importances
feature_importances = model.feature_importances_
feature_names = X.columns
sorted_idx = np.argsort(feature_importances)[::-1]
top_n = min(10, len(feature_importances))

plt.figure(figsize=(10, 6))
plt.title("Top Feature Importances")
plt.bar(range(top_n), feature_importances[sorted_idx][:top_n], align="center")
plt.xticks(range(top_n), feature_names[sorted_idx][:top_n], rotation=90)
plt.xlabel("Relative Feature")
plt.ylabel("Feature Importance")
plt.show()
```



In [49]:
```
import numpy as np
import matplotlib.pyplot as plt

# Visualize feature importances
feature_importances = model.feature_importances_
feature_names = X.columns
sorted_idx = np.argsort(feature_importances)[::-1]
top_n = min(10, len(feature_importances))

plt.figure(figsize=(10, 6))
plt.title("Top Feature Importances")
plt.bar(range(top_n), feature_importances[sorted_idx][:top_n], align="center")
plt.xticks(range(top_n), feature_names[sorted_idx][:top_n], rotation=90)
plt.xlabel("Relative Feature")
plt.ylabel("Feature Importance")
plt.show()
```



In [ ]: