Task 2

Predictive modeling of customer bookings

This Jupyter notebook includes some code to get you started with this predictive modeling task. We will use various packages for data manipulation, feature engineering and machine learning.

Exploratory data analysis

First, we must explore the data in order to better understand what we have and the statistical properties of the dataset.

In [1]:
```
import pandas as pd
```

In [3]:
```
df = pd.read_csv("customer_booking.csv", encoding="ISO-8859-1")
df.head()
```

Out[3]:

	num_passengers	sales_channel	trip_type	purchase_lead	length_of_stay	flight_hour	flight_day	route	booking_origin	wa
0	2	Internet	RoundTrip	262	19	7	Sat	AKLDEL	New Zealand	
1	1	Internet	RoundTrip	112	20	3	Sat	AKLDEL	New Zealand	
2	2	Internet	RoundTrip	243	22	17	Wed	AKLDEL	India	
3	1	Internet	RoundTrip	96	31	4	Sat	AKLDEL	New Zealand	
4	2	Internet	RoundTrip	68	22	15	Wed	AKLDEL	India	

The `.head()` method allows us to view the first 5 rows in the dataset, this is useful for visual inspection of our columns

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 14 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   num_passengers         50000 non-null  int64  
 1   sales_channel          50000 non-null  object 
 2   trip_type              50000 non-null  object 
 3   purchase_lead          50000 non-null  int64  
 4   length_of_stay         50000 non-null  int64  
 5   flight_hour            50000 non-null  int64  
 6   flight_day             50000 non-null  object 
 7   route                  50000 non-null  object 
 8   booking_origin         50000 non-null  object 
 9   wants_extra_baggage    50000 non-null  int64  
 10  wants_preferred_seat   50000 non-null  int64  
 11  wants_in_flight_meals  50000 non-null  int64  
 12  flight_duration        50000 non-null  float64
 13  booking_complete       50000 non-null  int64  
dtypes: float64(1), int64(8), object(5)
memory usage: 5.3+ MB
```

The `.info()` method gives us a data description, telling us the names of the columns, their data types and how many null values we have. Fortunately, we have no null values. It looks like some of these columns should be converted into different data types, e.g. flight_day.

To provide more context, below is a more detailed data description, explaining exactly what each column means:

- `num_passengers` = number of passengers travelling
- `sales_channel` = sales channel booking was made on
- `trip_type` = trip Type (Round Trip, One Way, Circle Trip)
- `purchase_lead` = number of days between travel date and booking date
- `length_of_stay` = number of days spent at destination
- `flight_hour` = hour of flight departure
- `flight_day` = day of week of flight departure
- `route` = origin -> destination flight route
- `booking_origin` = country from where booking was made
- `wants_extra_baggage` = if the customer wanted extra baggage in the booking
- `wants_preferred_seat` = if the customer wanted a preferred seat in the booking

- `wants_in_flight_meals` = if the customer wanted in-flight meals in the booking
- `flight_duration` = total duration of flight (in hours)
- `booking_complete` = flag indicating if the customer completed the booking

Before we compute any statistics on the data, lets do any necessary data conversion

In [5]:
```
df["flight_day"].unique()
```

Out[5]:
```
array(['Sat', 'Wed', 'Thu', 'Mon', 'Sun', 'Tue', 'Fri'], dtype=object)
```

In [6]:
```
mapping = {
    "Mon": 1,
    "Tue": 2,
    "Wed": 3,
    "Thu": 4,
    "Fri": 5,
    "Sat": 6,
    "Sun": 7,
}

df["flight_day"] = df["flight_day"].map(mapping)
```

In [7]:
```
df["flight_day"].unique()
```

Out[7]:
```
array([6, 3, 4, 1, 7, 2, 5])
```

In [9]:
```
print(df.describe())
```

```
       num_passengers  purchase_lead  length_of_stay  flight_hour  \
count    50000.000000   50000.000000    50000.00000  50000.00000
mean         1.591240      84.940480       23.04456      9.06634
std          1.020165      90.451378       33.88767      5.41266
min          1.000000       0.000000        0.00000      0.00000
25%          1.000000      21.000000        5.00000      5.00000
50%          1.000000      51.000000       17.00000      9.00000
75%          2.000000     115.000000       28.00000     13.00000
max          9.000000     867.000000      778.00000     23.00000

         flight_day  wants_extra_baggage  wants_preferred_seat  \
count  50000.000000         50000.000000          50000.000000
mean       3.814420             0.668780              0.296960
std        1.992792             0.470657              0.456923
min        1.000000             0.000000              0.000000
25%        2.000000             0.000000              0.000000
50%        4.000000             1.000000              0.000000
75%        5.000000             1.000000              1.000000
max        7.000000             1.000000              1.000000

       wants_in_flight_meals  flight_duration  booking_complete
count           50000.000000     50000.000000      50000.000000
mean                0.427140         7.277561          0.149560
std                 0.494668         1.496863          0.356643
min                 0.000000         4.670000          0.000000
25%                 0.000000         5.620000          0.000000
50%                 0.000000         7.570000          0.000000
75%                 1.000000         8.830000          0.000000
max                 1.000000         9.500000          1.000000
```

The `.describe()` method gives us a summary of descriptive statistics over the entire dataset (only works for numeric columns). This gives us a quick overview of a few things such as the mean, min, max and overall distribution of each column.

From this point, you should continue exploring the dataset with some visualisations and other metrics that you think may be useful. Then, you should prepare your dataset for predictive modelling. Finally, you should train your machine learning model, evaluate it with performance metrics and output visualisations for the contributing variables. All of this analysis should be summarised in your single slide.

In [10]:
```
df.isnull().sum()
```

Out[10]:
```
num_passengers           0
sales_channel            0
trip_type                0
purchase_lead            0
length_of_stay           0
flight_hour              0
flight_day               0
route                    0
booking_origin           0
wants_extra_baggage      0
wants_preferred_seat     0
wants_in_flight_meals    0
flight_duration          0
booking_complete         0
dtype: int64
```

In [11]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix, roc_auc_score
import matplotlib.pyplot as plt
import seaborn as sns
```

In [16]:
```
print(df.columns.tolist())
```

```
['num_passengers', 'sales_channel', 'trip_type', 'purchase_lead', 'length_of_stay', 'flight_hour', 'flight_day',
'route', 'booking_origin', 'wants_extra_baggage', 'wants_preferred_seat', 'wants_in_flight_meals', 'flight_durat
ion', 'booking_complete']
```

In [17]:
```
# Strip leading/trailing spaces and convert to lowercase
df.columns = df.columns.str.strip().str.lower()
```

In [19]:
```
print(df.columns.tolist())
```

```
['num_passengers', 'sales_channel', 'trip_type', 'purchase_lead', 'length_of_stay', 'flight_hour', 'flight_day',
'route', 'booking_origin', 'wants_extra_baggage', 'wants_preferred_seat', 'wants_in_flight_meals', 'flight_durat
ion', 'booking_complete']
```

In [21]:
```
X = df.drop('booking_complete', axis=1)
y = df['booking_complete']
```

In [23]:
```
# Basic shape and structure
print(df.shape)
print(df.columns)

# First few rows
df.head()

# Data types and missing values
df.info()
df.isnull().sum()

# Summary statistics
df.describe(include='all')
```

```
(50000, 14)
Index(['num_passengers', 'sales_channel', 'trip_type', 'purchase_lead',
       'length_of_stay', 'flight_hour', 'flight_day', 'route',
       'booking_origin', 'wants_extra_baggage', 'wants_preferred_seat',
       'wants_in_flight_meals', 'flight_duration', 'booking_complete'],
      dtype='object')
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 14 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   num_passengers         50000 non-null  int64
 1   sales_channel          50000 non-null  object
 2   trip_type              50000 non-null  object
 3   purchase_lead          50000 non-null  int64
 4   length_of_stay         50000 non-null  int64
 5   flight_hour            50000 non-null  int64
 6   flight_day             50000 non-null  object
 7   route                  50000 non-null  object
 8   booking_origin         50000 non-null  object
 9   wants_extra_baggage    50000 non-null  int64
 10  wants_preferred_seat   50000 non-null  int64
 11  wants_in_flight_meals  50000 non-null  int64
 12  flight_duration        50000 non-null  float64
 13  booking_complete       50000 non-null  int64
dtypes: float64(1), int64(8), object(5)
memory usage: 5.3+ MB
```

Out[23]:

	num_passengers	sales_channel	trip_type	purchase_lead	length_of_stay	flight_hour	flight_day	route	booking_origi
count	50000.000000	50000	50000	50000.000000	50000.00000	50000.00000	50000	50000	5000
unique	NaN	2	3	NaN	NaN	NaN	7	799	10
top	NaN	Internet	RoundTrip	NaN	NaN	NaN	Mon	AKLKUL	Austral
freq	NaN	44382	49497	NaN	NaN	NaN	8102	2680	1787
mean	1.591240	NaN	NaN	84.940480	23.04456	9.06634	NaN	NaN	Na
std	1.020165	NaN	NaN	90.451378	33.88767	5.41266	NaN	NaN	Na
min	1.000000	NaN	NaN	0.000000	0.00000	0.00000	NaN	NaN	Na
25%	1.000000	NaN	NaN	21.000000	5.00000	5.00000	NaN	NaN	Na
50%	1.000000	NaN	NaN	51.000000	17.00000	9.00000	NaN	NaN	Na
75%	2.000000	NaN	NaN	115.000000	28.00000	13.00000	NaN	NaN	Na
max	9.000000	NaN	NaN	867.000000	778.00000	23.00000	NaN	NaN	Na

In [27]:
```
df['booking_complete'].value_counts()
```

Out[27]:
```
booking_complete
0    42522
1     7478
Name: count, dtype: int64
```

In [28]:
```
## Local Steps to Complete the Task
#Sample Your Data
#To avoid memory issues, start by sampling 10,000 rows:
df_sampled = df.sample(n=10000, random_state=42)
```

Simplify High-Cardinality Columns Reduce route and booking_origin to the top 5 values and label the rest as 'Other':

In [29]:
```
for col in ['route', 'booking_origin']:
    top_cats = df_sampled[col].value_counts().nlargest(5).index
    df_sampled[col] = df_sampled[col].apply(lambda x: x if x in top_cats else 'Other')
```

In [30]:
```
from sklearn.preprocessing import LabelEncoder
for col in df_sampled.select_dtypes(include='object'):
    df_sampled[col] = LabelEncoder().fit_transform(df_sampled[col])
```

Train Your Model

In [31]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix, roc_auc_score

X = df_sampled.drop('booking_complete', axis=1)
y = df_sampled['booking_complete']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)

print(confusion_matrix(y_test, rf.predict(X_test)))
print(classification_report(y_test, rf.predict(X_test)))
print("ROC AUC:", roc_auc_score(y_test, rf.predict_proba(X_test)[:, 1]))
```

```
[[2515   40]
 [ 413   32]]
              precision    recall  f1-score   support

           0       0.86      0.98      0.92      2555
           1       0.44      0.07      0.12       445

    accuracy                           0.85      3000
   macro avg       0.65      0.53      0.52      3000
weighted avg       0.80      0.85      0.80      3000

ROC AUC: 0.7199454693374964
```

In [32]:
```
#Feature Importance
pd.DataFrame({
    'Feature': X.columns,
    'Importance': rf.feature_importances_
}).sort_values(by='Importance', ascending=False).head(10)
```

Out[32]:

	Feature	Importance
3	purchase_lead	0.219036
5	flight_hour	0.160288
4	length_of_stay	0.149129
12	flight_duration	0.103746
6	flight_day	0.102194
8	booking_origin	0.091893
0	num_passengers	0.054420
11	wants_in_flight_meals	0.029409
10	wants_preferred_seat	0.025336
7	route	0.024190

Model Evaluation with Cross-Validation Use Stratified K-Fold Cross-Validation to ensure the class imbalance is respected during splits:

In [33]:
```
from sklearn.model_selection import cross_val_score, StratifiedKFold

cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
cv_scores = cross_val_score(rf, X, y, cv=cv, scoring='roc_auc')

print("Cross-Validated AUC Scores:", cv_scores)
print("Mean AUC:", cv_scores.mean())
```

```
Cross-Validated AUC Scores: [0.73670061 0.72084662 0.71621721 0.6940347  0.72139977]
Mean AUC: 0.7178397808019287
```

In [35]:
```
y_pred = rf.predict(X_test)
```

In [36]:
```
y_prob = rf.predict_proba(X_test)[:, 1]
```

In [37]:
```
# Train the model
rf.fit(X_train, y_train)

# Make predictions
y_pred = rf.predict(X_test)
y_prob = rf.predict_proba(X_test)[:, 1]  # For ROC AUC
```

In [38]:
```
from sklearn.metrics import accuracy_score, precision_score, recall_score

print("Accuracy:", accuracy_score(y_test, y_pred))
print("Precision:", precision_score(y_test, y_pred))
print("Recall:", recall_score(y_test, y_pred))
```

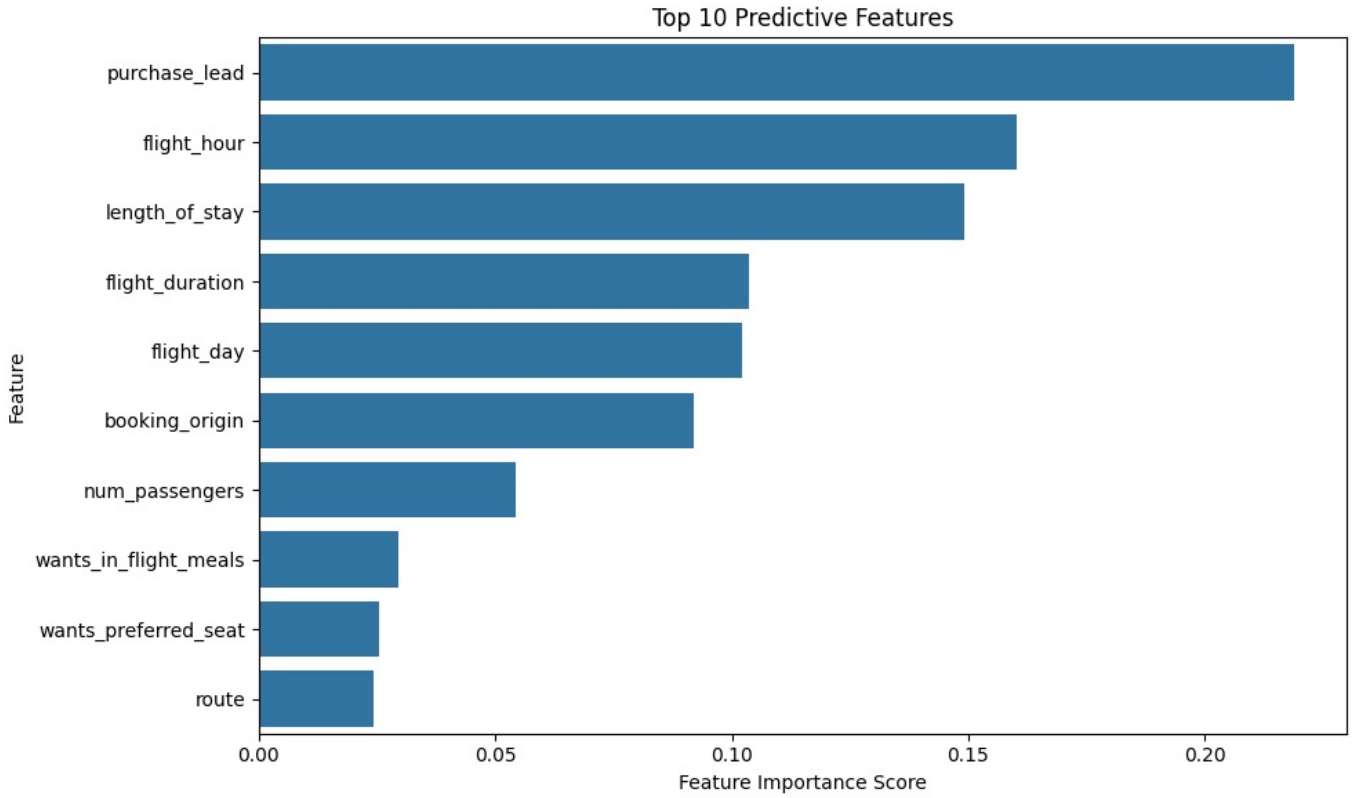
```
Accuracy: 0.849
Precision: 0.4444444444444444
Recall: 0.07191011235955057
```

In [40]:
```
import pandas as pd

feature_importances = pd.DataFrame({
    'Feature': X.columns,
    'Importance': rf.feature_importances_
}).sort_values(by='Importance', ascending=False)
```

In [41]:
```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(10, 6))
sns.barplot(x='Importance', y='Feature', data=feature_importances.head(10))
plt.title('Top 10 Predictive Features')
plt.xlabel('Feature Importance Score')
plt.tight_layout()
plt.show()
```


Top 10 Predictive Features
purchase_lead
flight_hour
length_of_stay
flight_duration
flight_day
booking_origin
num_passengers
wants_in_flight_meals
wants_preferred_seat
route
Feature
0.00
0.05
0.10
0.15
0.20
Feature Importance Score

In [ ]: