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
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df = pd.read_csv(r"C:\Users\Käyttäjä\Downloads\customer_booking.csv", encoding="ISO-8859-1")
df.head()
```

Out[2]:

	num_passengers	sales_channel	trip_type	purchase_lead	length_of_stay	flight_hour	flight_day	route	booking_origin	wants_extra
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 [3]:
```
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 [4]:
```
df["flight_day"].unique()
```

Out[4]:
```
array(['Sat', 'Wed', 'Thu', 'Mon', 'Sun', 'Tue', 'Fri'], dtype=object)
```

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

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

In [6]:
```
df["flight_day"].unique()
```

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

In [7]:
```
df.describe()
```

Out[7]:

	num_passengers	purchase_lead	length_of_stay	flight_hour	flight_day	wants_extra_baggage	wants_preferred_seat	wants_in_f
count	50000.000000	50000.000000	50000.00000	50000.00000	50000.000000	50000.000000	50000.000000	50
mean	1.591240	84.940480	23.04456	9.06634	3.814420	0.668780	0.296960	
std	1.020165	90.451378	33.88767	5.41266	1.992792	0.470657	0.456923	
min	1.000000	0.000000	0.00000	0.00000	1.000000	0.000000	0.000000	
25%	1.000000	21.000000	5.00000	5.00000	2.000000	0.000000	0.000000	
50%	1.000000	51.000000	17.00000	9.00000	4.000000	1.000000	0.000000	
75%	2.000000	115.000000	28.00000	13.00000	5.000000	1.000000	1.000000	
max	9.000000	867.000000	778.00000	23.00000	7.000000	1.000000	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 [16]:
```
print(df.columns.tolist())
```

```
['num_passengers', 'sales_channel', 'trip_type', 'purchase_lead', 'length_of_stay', 'flight_hour', 'flight_da
y', 'booking_origin', 'wants_extra_baggage', 'wants_preferred_seat', 'wants_in_flight_meals', 'flight_duratio
n', 'booking_complete']
```

In [18]:
```
label_encoders = {}
for col in df.select_dtypes(include="object").columns:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    label_encoders[col] = le
```

In [20]:
```
X = df.drop("booking_complete", axis=1)
y = df["booking_complete"]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = RandomForestClassifier(random_state=42, class_weight="balanced")
model.fit(X_train, y_train)
```

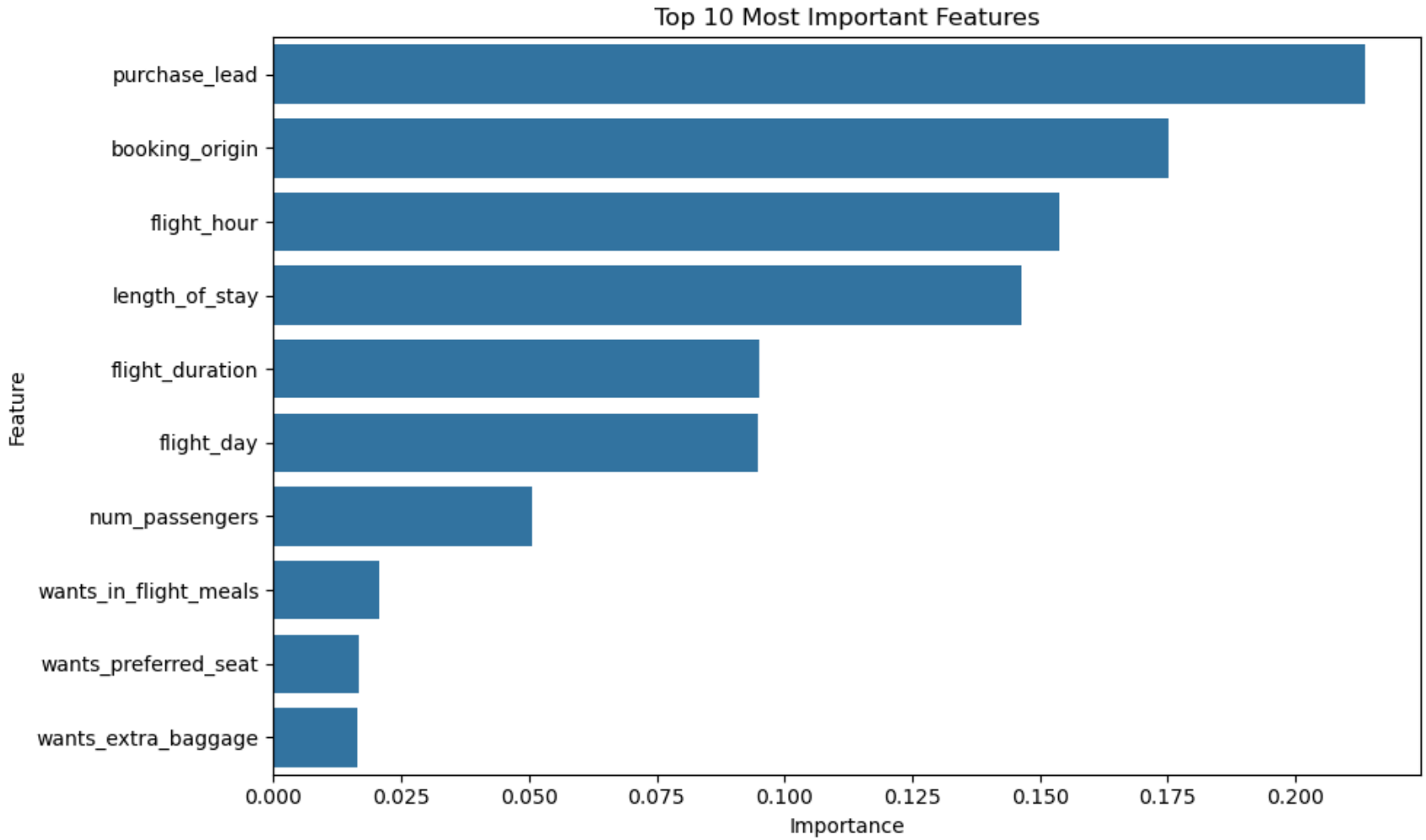
Out[20]:
```
RandomForestClassifier

RandomForestClassifier(class_weight='balanced', random_state=42)
```

In [21]:
```
y_pred = model.predict(X_test)
print(classification_report(y_test, y_pred))
print("Cross-validation score:", cross_val_score(model, X, y, cv=5).mean())
```

```
              precision    recall  f1-score   support

           0       0.86      0.99      0.92      8520
           1       0.53      0.09      0.15      1480

    accuracy                           0.85     10000
   macro avg       0.70      0.54      0.54     10000
weighted avg       0.81      0.85      0.81     10000

Cross-validation score: 0.7127999999999999
```

In [22]:

```
importances = model.feature_importances_
feature_names = X.columns

feat_imp = pd.Series(importances, index=feature_names).sort_values(ascending=False)

plt.figure(figsize=(10, 6))
sns.barplot(x=feat_imp[:10], y=feat_imp[:10].index)
plt.title("Top 10 Most Important Features")
plt.xlabel("Importance")
plt.ylabel("Feature")
plt.tight_layout()
plt.savefig("feature_importance.png")  # Saves the image for your PowerPoint
plt.show()
```


Top 10 Most Important Features
purchase_lead
booking_origin
flight_hour
length_of_stay
flight_duration
flight_day
num_passengers
wants_in_flight_meals
wants_preferred_seat
wants_extra_baggage
Feature
0.000
0.025
0.050
0.075
0.100
0.125
0.150
0.175
0.200
Importance

In [ ]: