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.

```
import pandas as pd
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
df = pd.read_csv("/content/drive/MyDrive/datasets/customer_booking.csv", encoding="ISO-8859-1")
df.head()
```

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

Next steps: Generate code with df View recommended plots New interactive sheet

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

```
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

```
df["flight_day"].unique()
```

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

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

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

```
df["flight_day"].unique()
```

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

```
df.describe()
```

	num_passengers	purchase_lead	length_of_stay	flight_hour	flight_day	wants_extra_baggage	wants_preferred_se
count	50000.000000	50000.000000	50000.00000	50000.00000	50000.000000	50000.000000	50000.000(
mean	1.591240	84.940480	23.04456	9.06634	3.814420	0.668780	0.296(
std	1.020165	90.451378	33.88767	5.41266	1.992792	0.470657	0.456(
min	1.000000	0.000000	0.00000	0.00000	1.000000	0.000000	0.000(
25%	1.000000	21.000000	5.00000	5.00000	2.000000	0.000000	0.000(
50%	1.000000	51.000000	17.00000	9.00000	4.000000	1.000000	0.000(
75%	2.000000	115.000000	28.00000	13.00000	5.000000	1.000000	1.000(
max	9.000000	867.000000	778.00000	23.00000	7.000000	1.000000	1.000(

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.

Double-click (or enter) to edit

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import OneHotEncoder
from sklearn.metrics import classification_report, roc_auc_score
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
import matplotlib.pyplot as plt
import seaborn as sns


# Split features and target
X = df.drop("booking_complete", axis=1)
y = df["booking_complete"]


# Identify column types
categorical_cols = X.select_dtypes(include=["object"]).columns.tolist()
numerical_cols = X.select_dtypes(include=["int64", "float64"]).columns.tolist()


# Preprocessing pipeline
preprocessor = ColumnTransformer(
    transformers=[
        ("cat", OneHotEncoder(handle_unknown="ignore"), categorical_cols)
    ],
    remainder="passthrough"
)


# Full pipeline with RandomForest
model = Pipeline(steps=[
    ("preprocessor", preprocessor),
    ("classifier", RandomForestClassifier(n_estimators=50, random_state=42))
])


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


# Train the model
model.fit(X_train, y_train)
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/compose/_column_transformer.py:1667: FutureWarning:
The format of the columns of the 'remainder' transformer in ColumnTransformer.transformers_ will change in version 1
At the moment the remainder columns are stored as indices (of type int). With the same ColumnTransformer configurati
To use the new behavior now and suppress this warning, use ColumnTransformer(force_int_remainder_cols=False).

  warnings.warn(
```

```
Pipeline
  preprocessor: ColumnTransformer
    cat                 remainder
    OneHotEncoder       passthrough
  RandomForestClassifier
```

```
# Predict and evaluate on test set
y_pred = model.predict(X_test)
y_proba = model.predict_proba(X_test)[:, 1]


# Print classification report and ROC AUC
```

```
# Print classification report and ROC AUC
report = classification_report(y_test, y_pred)
roc_auc = roc_auc_score(y_test, y_proba)
print("Classification Report:\n", report)
print("ROC AUC Score (Test Set):", roc_auc)
```

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

           0       0.87      0.98      0.92      8520
           1       0.51      0.14      0.22      1480

    accuracy                           0.85     10000
   macro avg       0.69      0.56      0.57     10000
weighted avg       0.82      0.85      0.82     10000

ROC AUC Score (Test Set): 0.7806748429767796
```

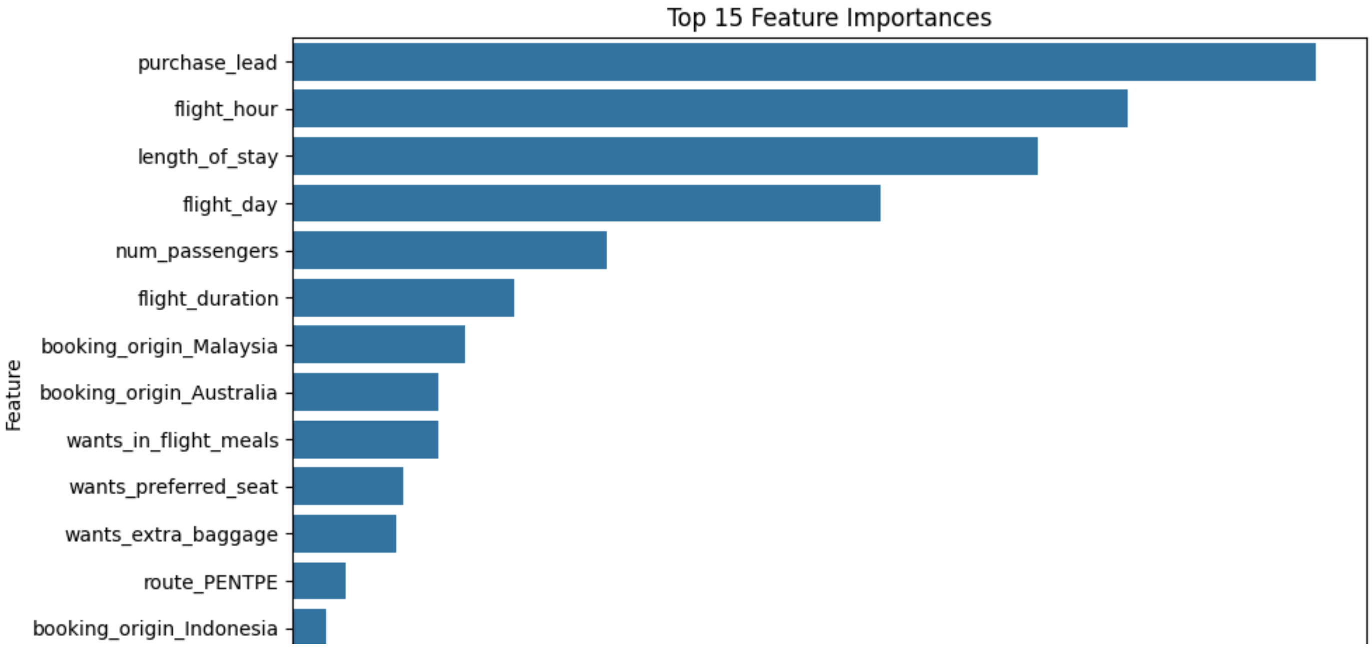
```
# Cross-validation (3-fold) ROC AUC
cv_scores = cross_val_score(model, X, y, cv=3, scoring='roc_auc')
print("Cross-Validated ROC AUC Scores:", cv_scores)
print("Mean CV ROC AUC Score:", cv_scores.mean())
```

```
Cross-Validated ROC AUC Scores: [0.62570269 0.46649309 0.74353042]
Mean CV ROC AUC Score: 0.6119087331944005
```

```
# Feature importances
# Refit model to access features easily (same as trained above)
ohe = model.named_steps["preprocessor"].named_transformers_["cat"]
ohe_feature_names = ohe.get_feature_names_out(categorical_cols)
all_feature_names = np.concatenate([ohe_feature_names, numerical_cols])


importances = model.named_steps["classifier"].feature_importances_
feat_importances = pd.Series(importances, index=all_feature_names).sort_values(ascending=False)


# Plot top 15 features
plt.figure(figsize=(10, 6))
sns.barplot(x=feat_importances.values[:15], y=feat_importances.index[:15])
plt.title("Top 15 Feature Importances")
plt.xlabel("Importance")
plt.ylabel("Feature")
plt.tight_layout()
plt.show()
```


Top 15 Feature Importances
Feature
purchase_lead
flight_hour
length_of_stay
flight_day
num_passengers
flight_duration
booking_origin_Malaysia
booking_origin_Australia
wants_in_flight_meals
wants_preferred_seat
wants_extra_baggage
route_PENTPE
booking_origin_Indonesia