British Airline Feedback Analyzing using Python

Presented By: Katie Nguyen

Crawling data from different sources: Using bs4 module

- Using bs4 module to collect 1000 data from 10 pages of the base URL.
- Results:

```
Total number of rows: 1000
Field names: , reviews
Not Verified
False    741
True     259
dtype: int64
```

```python
base_url = "https://www.airlinequality.com/airline-reviews/british-airways"
pages = 10
page_size = 100

reviews = []

# for i in range(1, pages + 1):
for i in range(1, pages + 1):

    print(f"Scraping page {i}")

    # Create URL to collect links from paginated data
    url = f"{base_url}/page/{i}/?sortby=post_date%3ADesc&pagesize={page_size}"

    # Collect HTML data from this page
    response = requests.get(url)

    # Parse content
    content = response.content
    parsed_content = BeautifulSoup(content, 'html.parser')
    for para in parsed_content.find_all("div", {"class": "text_content"}):
        reviews.append(para.get_text())

    print(f"   ---> {len(reviews)} total reviews")
```

```
Scraping page 1
   ---> 100 total reviews
Scraping page 2
   ---> 200 total reviews
Scraping page 3
   ---> 300 total reviews
Scraping page 4
   ---> 400 total reviews
Scraping page 5
   ---> 500 total reviews
Scraping page 6
   ---> 600 total reviews
Scraping page 7
   ---> 700 total reviews
Scraping page 8
   ---> 800 total reviews
Scraping page 9
   ---> 900 total reviews
Scraping page 10
   ---> 1000 total reviews
```

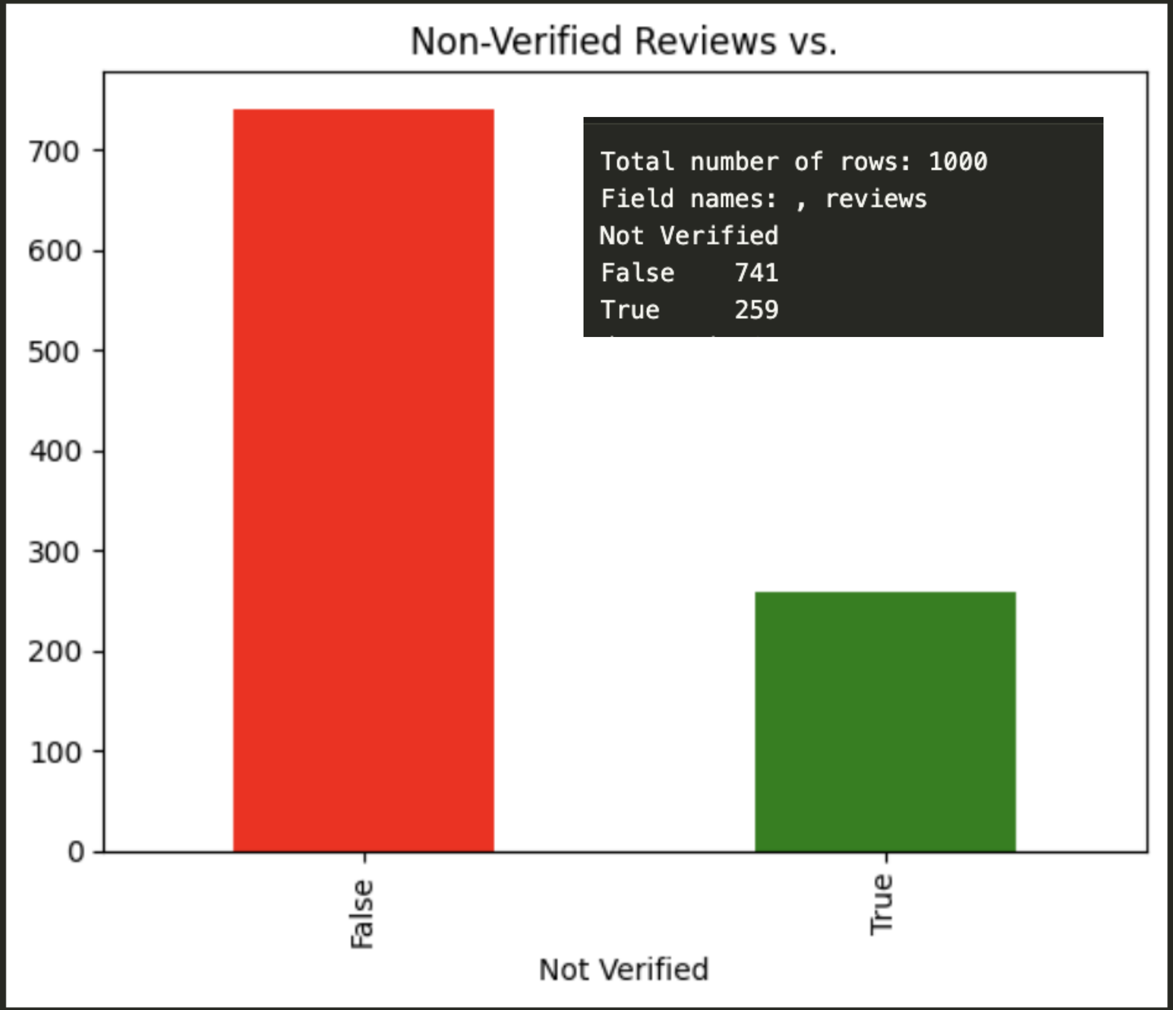
1. Number “Verified” vs. ”Not Verified” feedback:

Using the above Python script, I have obtained the Number of Verified Feedback (Red) and the Number of “Not Verified” feedback (Green).

There are 741 “Verified” feedback and 259 “Not Verified” feedback, in total of 1000 reviews.

```
import matplotlib.pyplot as plt
import seaborn as sns

#Bar chart for Verifield vs. Not Verified
df.groupby("Not Verified").size().plot(kind="bar", color=["red", "green"])
plt.title("Non-Verified Reviews vs.")
plt.show()
```

✓ 0.0s

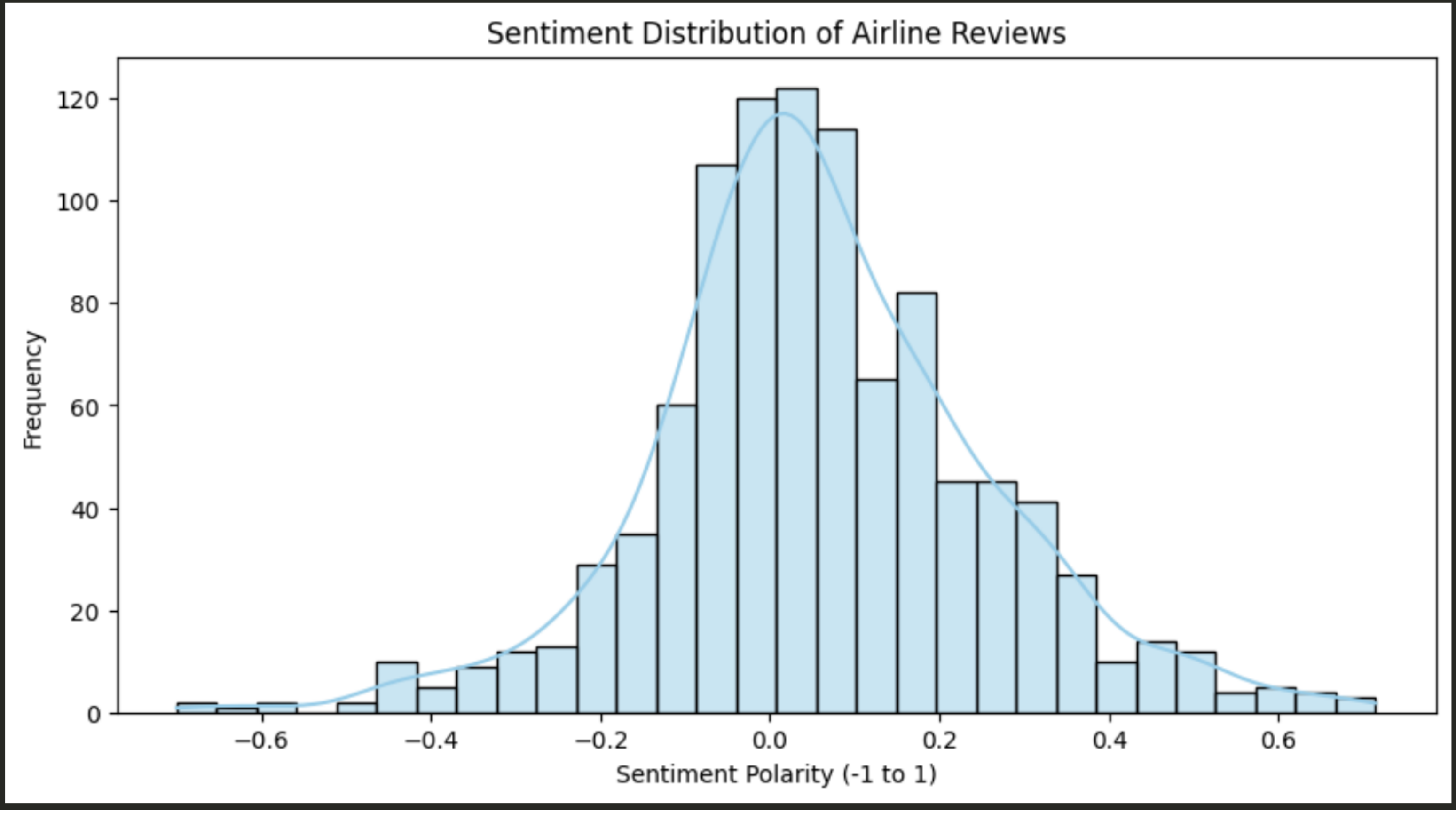


2. Sentiment Reviews Analyze:

- Negative reviews (polarity < 0) suggest dissatisfaction.
- Neutral reviews (around 0) indicate mixed feedback.
- Positive reviews (> 0) suggest customer satisfaction.

```python
plt.figure(figsize=(10, 5))
sns.histplot(df["sentiment"], bins=30, kde=True, color="skyblue")

plt.xlabel("Sentiment Polarity (-1 to 1)")
plt.ylabel("Frequency")
plt.title("Sentiment Distribution of Airline Reviews")
plt.show()
```

✓ 0.0s

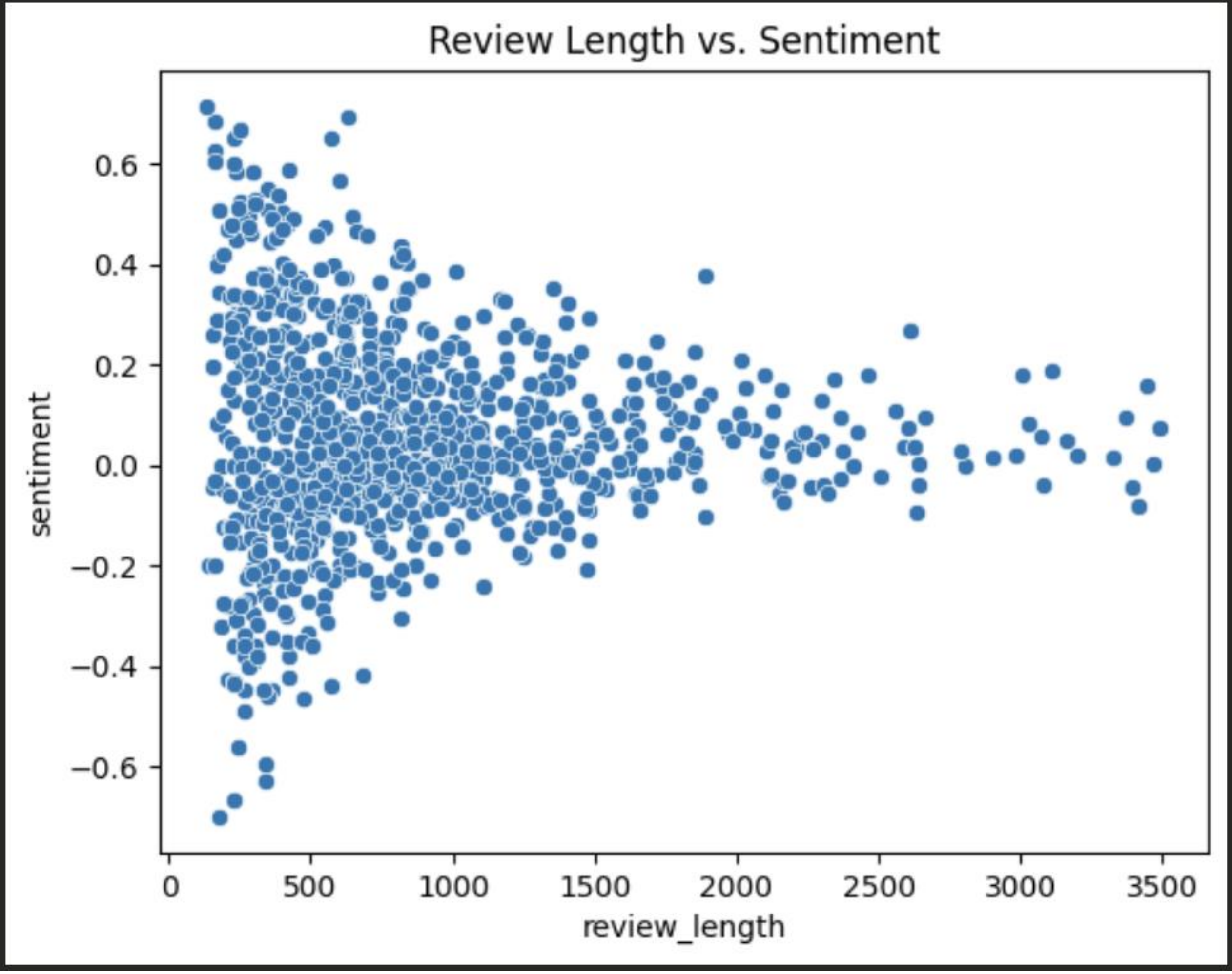


3. Do longer reviews tend to be more positive?

- It shows that longer reviews tend to be neutral and positive. While negative reviews tend to be shorter.

```python
df["review_length"] = df["reviews"].str.len()
sns.scatterplot(x=df["review_length"], y=df["sentiment"])
plt.title("Review Length vs. Sentiment")
plt.show()
```

✓ 0.0s



Thank you!
Q&A

Task 2: Building a model

Obtain a dataset:

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
```

✓ 0.3s

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

✓ 0.0s

	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
0	2	Internet	RoundTrip	262	19	7	Sat	AKLDEL	New Zealand	1	0	0	5.52	0
1	1	Internet	RoundTrip	112	20	3	Sat	AKLDEL	New Zealand	0	0	0	5.52	0
2	2	Internet	RoundTrip	243	22	17	Wed	AKLDEL	India	1	1	0	5.52	0
3	1	Internet	RoundTrip	96	31	4	Sat	AKLDEL	New Zealand	0	0	1	5.52	0
4	2	Internet	RoundTrip	68	22	15	Wed	AKLDEL	India	1	0	1	5.52	0

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

```python
df.info()
```

[3] ✓ 0.0s

```
<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

Some data’s info:

Cleaning data:

```
# 1. Clean data:
#check for missing value:
df.isnull().sum()
```

✓ 0.0s

```
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
```

Conclusion: There is no missing data, or null value -> OK to continue

Analyzing:

Findings:

- The number of customers who are solo travelers are larger than traveling in groups.
- The 7-9 am flight are the most often booked when comparing to others.

-> We can offer more flights depart between 7-9am. Or at noon.

- Usually, customers do not want preferred seat

-> We can offer promotion for people to increase chance of buying seat.

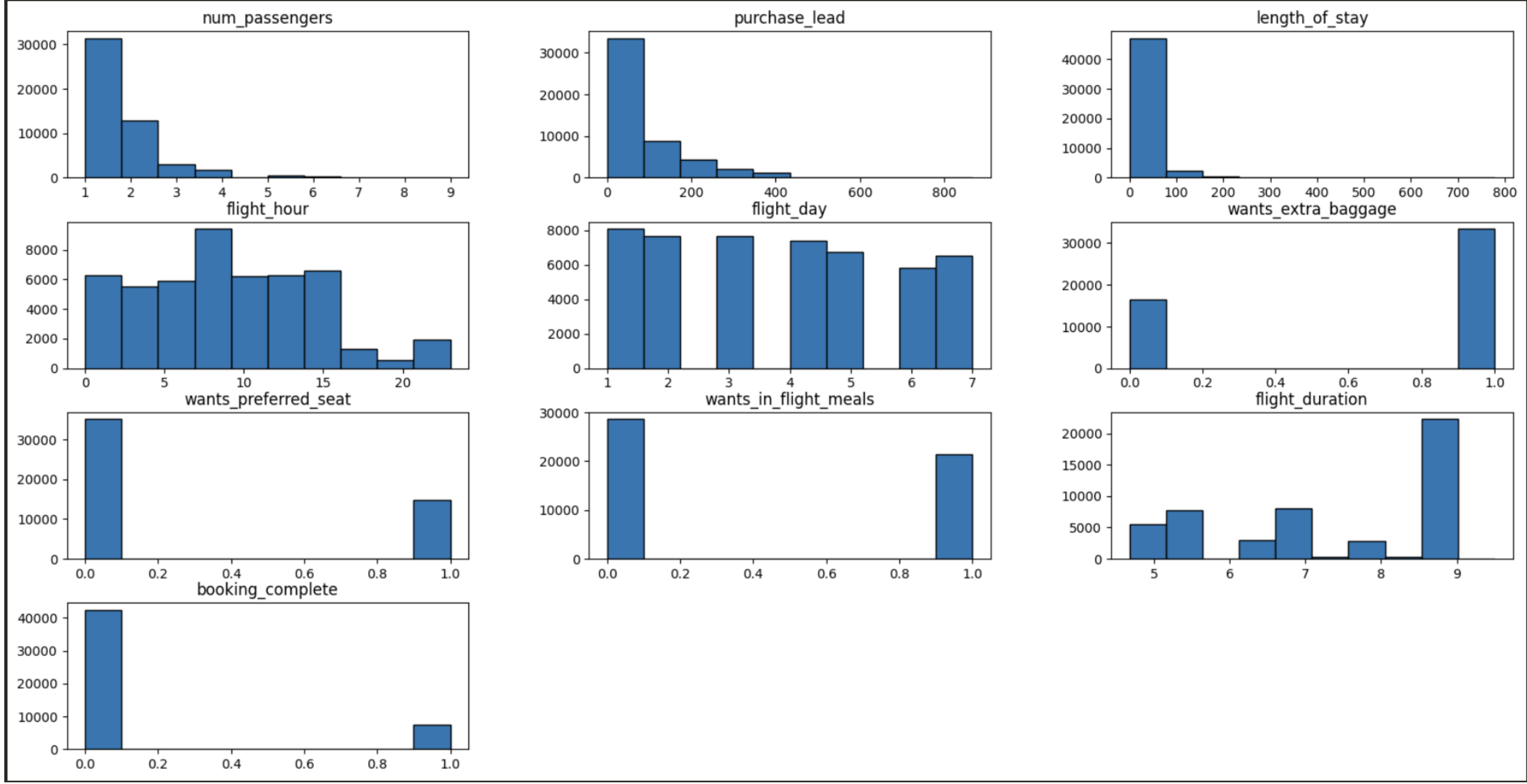
- Usually, customers booking is not completed. (0 means False, 1 means True)

-> Could be because of website feature, difficult to complete the booking? -> We can offer more user-friendly, easy to book feature.

```
#2. Visualizing:
#Plot histogram: Since I want to display the plot more beautifully

import matplotlib.pyplot as plt
fig = plt.figure(figsize=(20,10))
ax = fig.gca()

#create histogram using specified figure size
df.hist(grid=False, edgecolor = 'black', ax=ax)
```

✓ 0.6s

```
/var/folders/8l/3v0qcjcs3v53vjdw26w63wgr0000gn/T/ipykernel_10364/3740457870.py:9: UserWarning: To output multiple subplots, the figure containing the passed axes is being cleared.
  df.hist(grid=False, edgecolor = 'black', ax=ax)

array([[<Axes: title={'center': 'num_passengers'}>,
        <Axes: title={'center': 'purchase_lead'}>,
        <Axes: title={'center': 'length_of_stay'}>],
       [<Axes: title={'center': 'flight_hour'}>,
        <Axes: title={'center': 'flight_day'}>,
        <Axes: title={'center': 'wants_extra_baggage'}>],
       [<Axes: title={'center': 'wants_preferred_seat'}>,
        <Axes: title={'center': 'wants_in_flight_meals'}>,
        <Axes: title={'center': 'flight_duration'}>],
       [<Axes: title={'center': 'booking_complete'}>, <Axes: >, <Axes: >]],
      dtype=object)
```



Analyzing (Con't)

Common myth, booking on weekdays are cheaper than on weekends.

The time people complete a booking on weekdays are more than weekends.

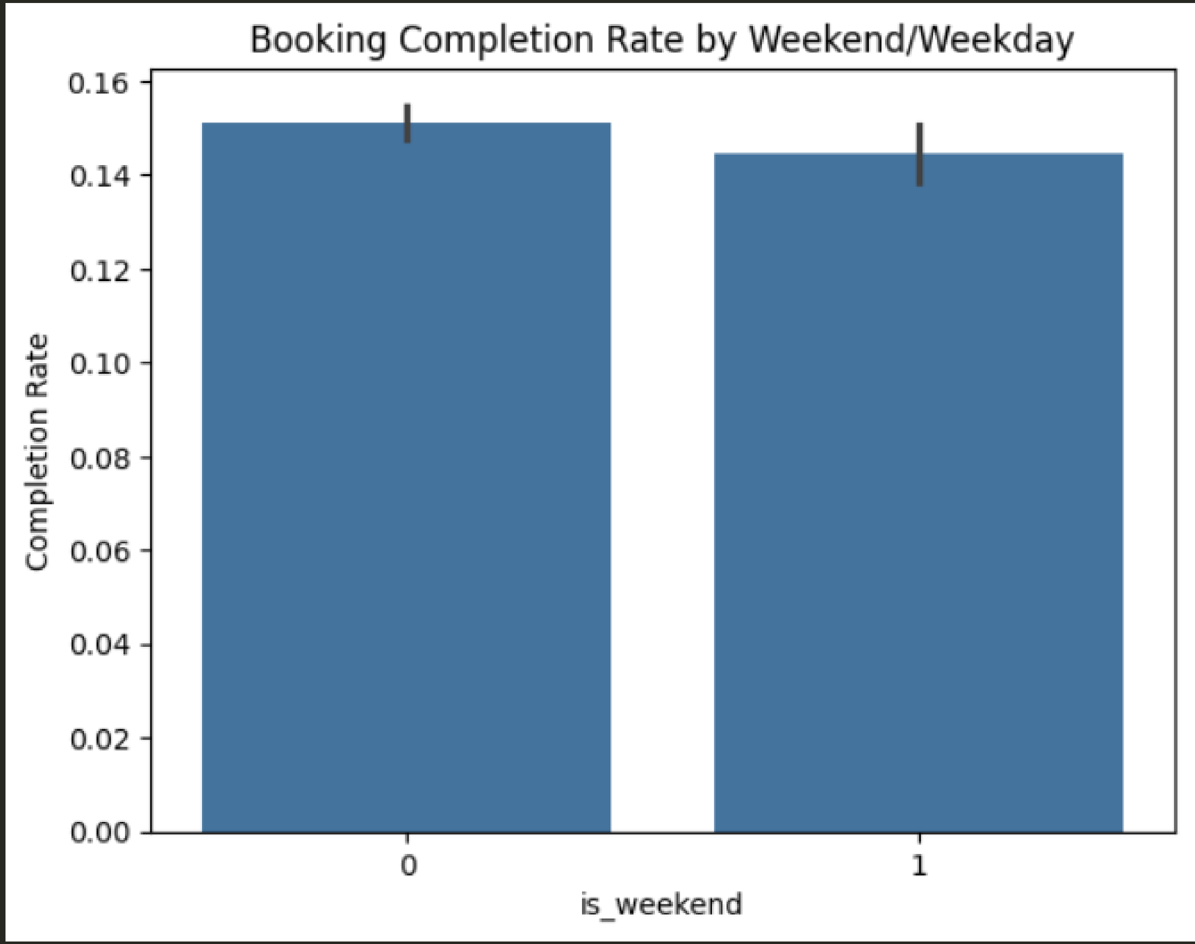
We can offer some special booking hours on weekends to attract people to complete their booking right away.

Suggestion:

- **Weekend Flash Deals:** Create time-sensitive offers like "Book between 5 PM - 8 PM for an extra 10% off."
- **Early Bird or Late-Night Discounts:** If weekday bookings tend to happen at specific times, mirror those patterns on weekends with incentives.
- **Weekend-Only Bundles:** Offer perks like free seat selection, extra baggage, or meal upgrades during slower booking hours.
- **Targeted Marketing:** Use ads and email campaigns to highlight weekend promotions, especially to customers who tend to book last-minutes

```python
import seaborn as sns
import matplotlib.pyplot as plt

chart = sns.barplot(x=df["is_weekend"], y=df["booking_complete"])
plt.title("Booking Completion Rate by Weekend/Weekday")
plt.xlabel("is_weekend")
plt.ylabel("Completion Rate")
plt.show()
```

✓ 0.3s



Analyzing (Con't)

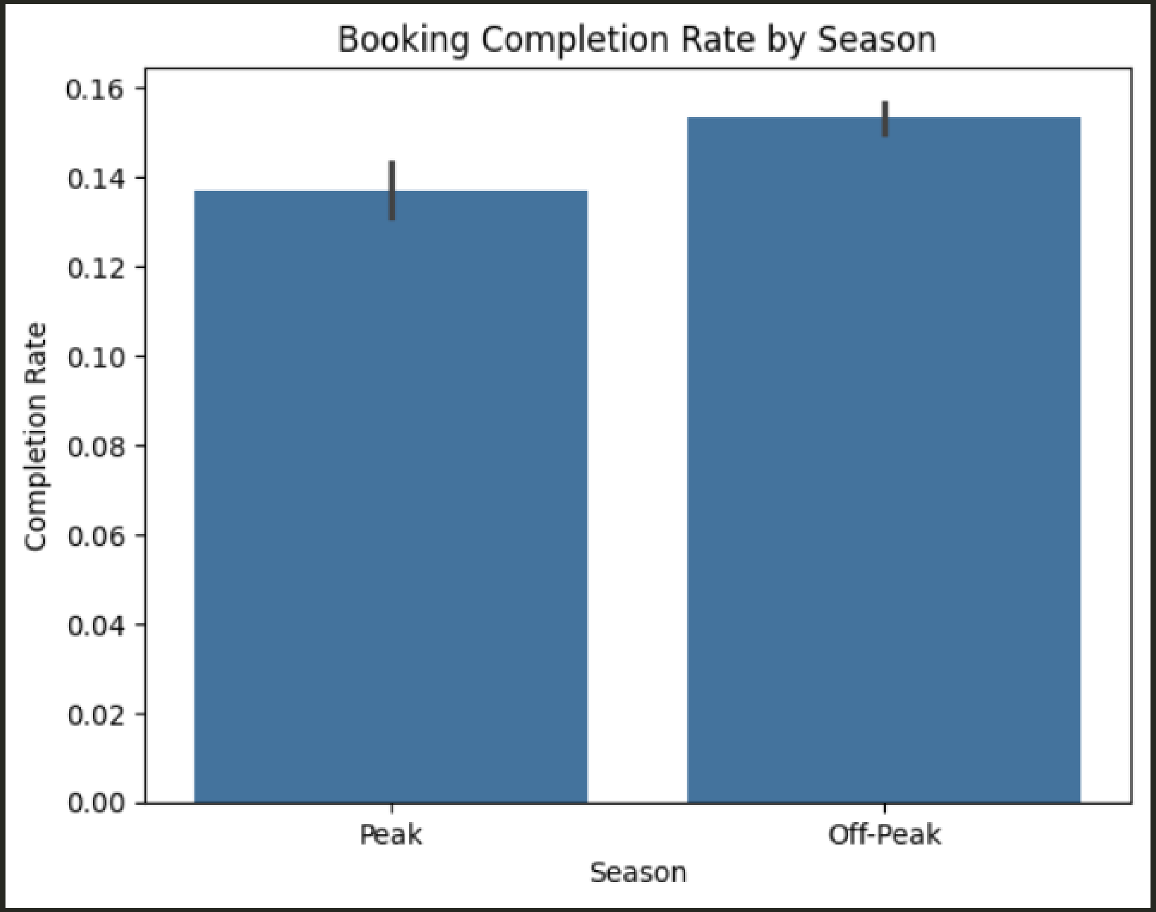
The time people complete a booking during off-peak season are more than peak seasons. => People tend to book in advance for better deal?

Suggestion:

- **Peak-season Flash Deals:** Create a good deal for people who are undecided during peak season. Such as: Spring break deal with affiliated hotels, or car rentals.
- **Targeted Marketing:** Use ads and email campaigns to highlight attractive programs during peak seasons, especially to customers who tend to book last-minutes

```python
import seaborn as sns
import matplotlib.pyplot as plt

sns.barplot(x=df["season"], y=df["booking_complete"])
plt.title("Booking Completion Rate by Season")
plt.xlabel("Season")
plt.ylabel("Completion Rate")
plt.show()
```

✓ 0.3s



Analyzing (Con’t)

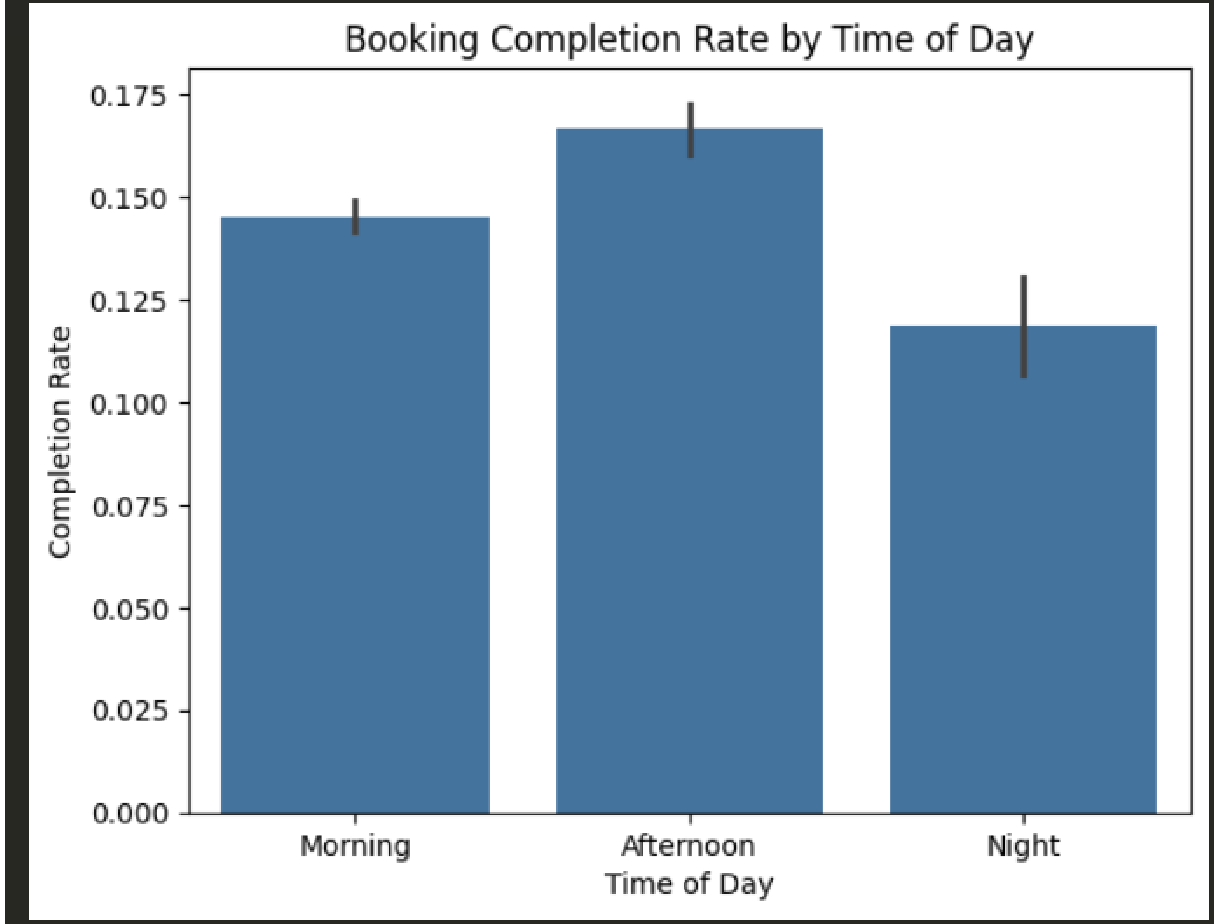
The time people complete a booking during in the afternoon are more than in the morning or night.

Suggestion:

- **Early Bird or Late-Night Discounts**
- **Sneak-peak deal in the morning or late night, such as: offer some free bonus points for customers who complete the booking between 12:00AM and 3:00AM or between 7:00AM to 9:00AM.**

```
import seaborn as sns
import matplotlib.pyplot as plt

sns.barplot(x=df["time_of_day"], y=df["booking_complete"])
plt.title("Booking Completion Rate by Time of Day")
plt.xlabel("Time of Day")
plt.ylabel("Completion Rate")
plt.show()
```

✓ 0.3s



Analyzing (Con’t)

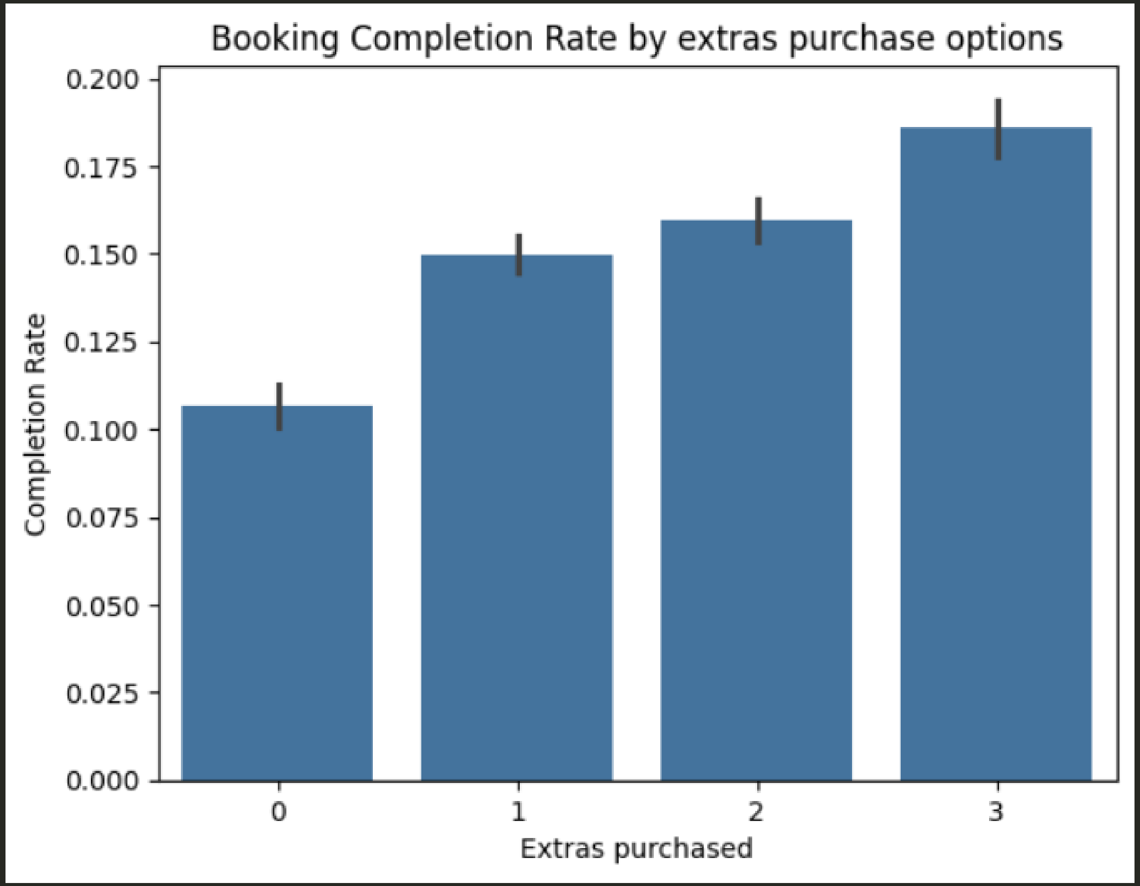
More bags purchased means higher chances of booking completeness.

Suggestion:

- **Seat secured when purchase a bag at the booking time.**
- **Preferred meals orders with a fee.**

```python
import seaborn as sns
import matplotlib.pyplot as plt

sns.barplot(x=df["extras_purchased"], y=df["booking_complete"])
plt.title("Booking Completion Rate by extras purchase options")
plt.xlabel("Extras purchased")
plt.ylabel("Completion Rate")
plt.show()
```

✓ 0.3s



Model training:

- Use the Random forest model since the booking_complete column is binary (0 and 1).
- Results:

```
Model Accuracy: 0.85
```

Ideas - Prepare data for Modeling:

1. Preprocessing Data:

- Ensuring the data is cleaned and ready for modeling. Steps: Handling missing values, encoding categorical variables (sales_channel, trip_type, route, and booking_origin, etc. into numerical representations)

```python
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer

# Identify categorical & numerical features
categorical_features = ['sales_channel', 'trip_type', 'flight_day', 'route', 'booking_origin']
numerical_features = ['num_passengers', 'purchase_lead', 'length_of_stay', 'flight_hour', 'flight_duration']

# Define preprocessing steps
preprocessor = ColumnTransformer([
    ('num', StandardScaler(), numerical_features),
    ('cat', OneHotEncoder(handle_unknown='ignore'), categorical_features)
])

# Split data
X = df.drop(columns=['booking_complete'])
y = df['booking_complete']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

✓ 2.8s

2. Choosing a model:

- Considering booking_complete is a binary target (Y/N) or (0/1):
  - Logistic Regression: Simple, interpretable, and effective.
  - Random forest: Greate for captureing complex rela.
  - XGBoost: Works well with structured data and handles missing values efficiently.

```python
from sklearn.ensemble import RandomForestClassifier
from sklearn.pipeline import Pipeline
from sklearn.metrics import accuracy_score

# Create pipeline
model = Pipeline([
    ('preprocessor', preprocessor),
    ('classifier', RandomForestClassifier(n_estimators=100, random_state=42))
])

# Train model
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Evaluate performance
accuracy = accuracy_score(y_test, y_pred)
print(f'Model Accuracy: {accuracy:.2f}')
```

✓ 13.6s

Thank you!

Q&A