PredictiveModelling_BA

October 23, 2024

0.1 Predictive modeling of customer bookings

0.1.1 Exploratory data analysis

```
[2]: import pandas as pd
```

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

```
[3]:    num_passengers sales_channel  trip_type  purchase_lead  length_of_stay  \
     0               2      Internet  RoundTrip            262              19
     1               1      Internet  RoundTrip            112              20
     2               2      Internet  RoundTrip            243              22
     3               1      Internet  RoundTrip             96              31
     4               2      Internet  RoundTrip             68              22

        flight_hour flight_day   route booking_origin  wants_extra_baggage  \
     0            7        Sat  AKLDEL    New Zealand                    1
     1            3        Sat  AKLDEL    New Zealand                    0
     2           17        Wed  AKLDEL          India                    1
     3            4        Sat  AKLDEL    New Zealand                    0
     4           15        Wed  AKLDEL          India                    1

        wants_preferred_seat  wants_in_flight_meals  flight_duration  \
     0                     0                      0             5.52
     1                     0                      0             5.52
     2                     1                      0             5.52
     3                     0                      1             5.52
     4                     0                      1             5.52

        booking_complete
     0                 0
     1                 0
     2                 0
     3                 0
     4                 0
```

```
[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
```

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

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

[6]:
```
array(['Sat', 'Wed', 'Thu', 'Mon', 'Sun', 'Tue', 'Fri'], dtype=object)
```

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

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

[8]:
```
df["flight_day"].unique()
```

[8]:
```
array([6, 3, 4, 1, 7, 2, 5], dtype=int64)
```

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

[9]:
```
       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
```

[10]:
```
# Check for missing values
print(df.isnull().sum())
```

```
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
```

[11]:

```
duplicates = df.duplicated().sum()
print(f'Duplicate rows: {duplicates}')
```

```
Duplicate rows: 719
```

[12]:

```
print(df['sales_channel'].unique())
print(df['trip_type'].unique())
```

```
['Internet' 'Mobile']
['RoundTrip' 'CircleTrip' 'OneWay']
```

[13]:

```
# Drop duplicates if any
if duplicates > 0:
    df = df.drop_duplicates()
    print("Duplicates removed.")
```

```
Duplicates removed.
```

[14]:

```
df = df.dropna(subset=['booking_complete'])
```

0.1.2 Feature Engineering

[16]:

```
# Create lead time categories
df['lead_time_category'] = pd.cut(df['purchase_lead'],
                                   bins=[-1, 30, 60, 90, 180, 360, 1000],
                                   labels=['Same-day', 'Short-term',␣
 ↪'Mid-term', 'Long-term', 'Very Long-term', 'Extreme Long-term'])
```

[17]:

```
# Create length of stay categories
df['length_of_stay_category'] = pd.cut(df['length_of_stay'],
                                        bins=[-1, 3, 7, 14, 30, 60, 365],
                                        labels=['Short', 'Medium', 'Long',␣
 ↪'Extended', 'Very Extended', 'Extreme'])
```

[18]:

```
# Convert flight hour to categories
df['time_of_day'] = pd.cut(df['flight_hour'],
                           bins=[-1, 6, 12, 18, 24],
                           labels=['Night', 'Morning', 'Afternoon', 'Evening'])
```

[19]:

```
# One-hot encoding for categorical features
df_encoded = pd.get_dummies(df, drop_first=True)
```

0.1.3 Preparing Data for Modelling

[21]:

```
from sklearn.model_selection import train_test_split

X = df_encoded.drop('booking_complete', axis=1)
y = df_encoded['booking_complete']

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

0.1.4 Training the Model

[23]:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix
```

[24]:

```
# Train the Random Forest model
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Evaluate the model
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[8234  144]
 [1332  147]]
              precision    recall  f1-score   support

           0       0.86      0.98      0.92      8378
           1       0.51      0.10      0.17      1479

    accuracy                           0.85      9857
   macro avg       0.68      0.54      0.54      9857
weighted avg       0.81      0.85      0.80      9857
```

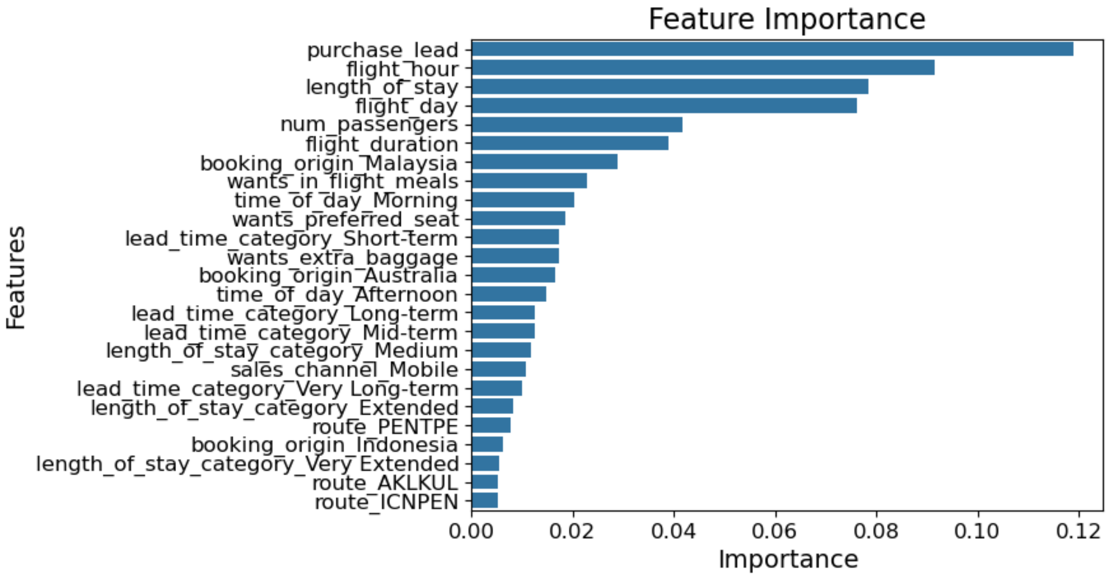
0.1.5 Display

[26]:

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

[58]:

```
sns.barplot(
    x=feature_importances.Importance[:top_n],
    y=feature_importances.index[:top_n]
)
plt.title('Feature Importance'.format(top_n), fontsize=16)
plt.xlabel('Importance', fontsize=14)
plt.ylabel('Features', fontsize=14)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.show()
```



[ ]: