Predictive modeling of customer bookings

Author: Rina RANARISON

www.rina-corp.xyz

www.github.com/ryurina

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 sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score,
confusion_matrix
import matplotlib.pyplot as plt

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

```
   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
```

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

```
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  \
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.

```
selected_columns = ['num_passengers', 'purchase_lead',
'length_of_stay', 'flight_hour',
                    'flight_day', 'wants_extra_baggage',
'wants_preferred_seat',
                    'wants_in_flight_meals', 'flight_duration',
'booking_complete']
df = df[selected_columns]
df.head()

   num_passengers  purchase_lead  length_of_stay  flight_hour
flight_day  \
0               2            262              19            7
6
1               1            112              20            3
6
2               2            243              22           17
3
3               1             96              31            4
6
4               2             68              22           15
3
```

```
   wants_extra_baggage  wants_preferred_seat  wants_in_flight_meals  \
0                    1                     0                      0
1                    0                     0                      0
2                    1                     1                      0
3                    0                     0                      1
4                    1                     0                      1

   flight_duration  booking_complete
0             5.52                 0
1             5.52                 0
2             5.52                 0
3             5.52                 0
4             5.52                 0
```

```
# Data Preparation
# Split the dataset into features (X) and the target (y)
X = df.drop(columns=['booking_complete'])
y = df['booking_complete']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Train a Random Forest Classifier
rf_classifier = RandomForestClassifier(n_estimators=100,
random_state=42)
rf_classifier.fit(X_train, y_train)
```

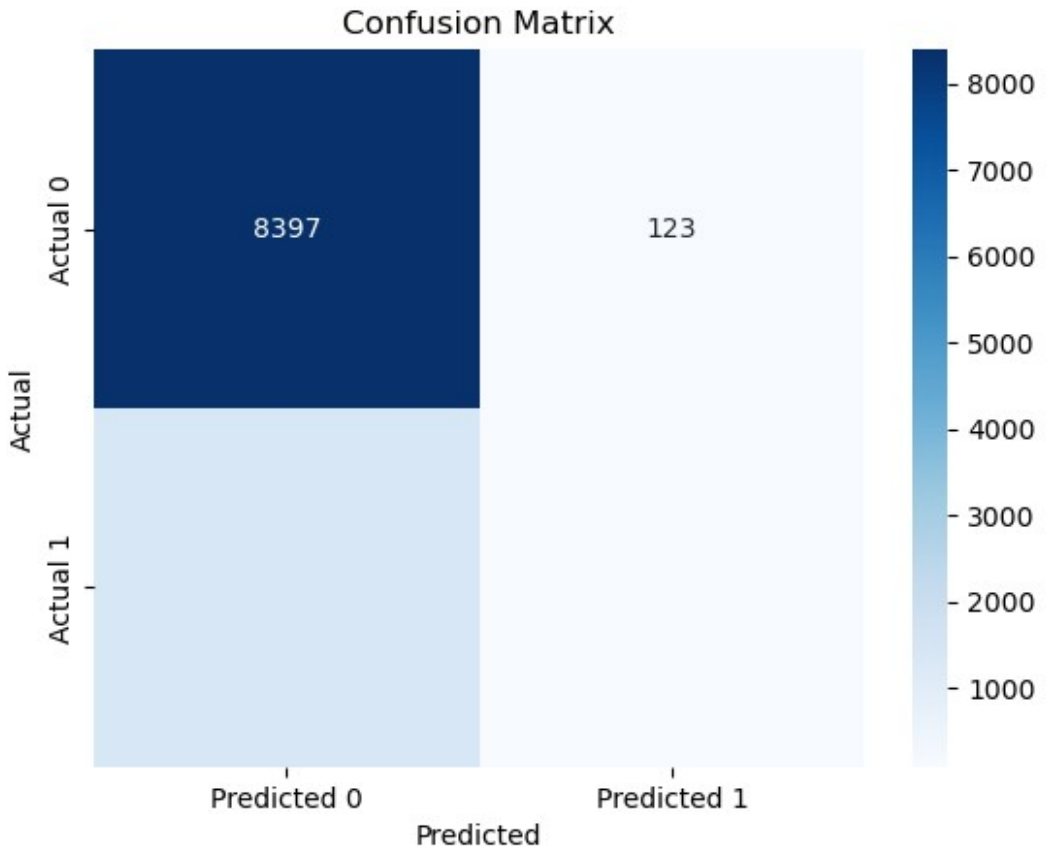
```
RandomForestClassifier(random_state=42)
```

```
# Make predictions on the test set
y_pred = rf_classifier.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)

# Print the evaluation metrics
print("Accuracy:", accuracy)
print("\nConfusion Matrix:\n", conf_matrix)
print("\nClassification Report:\n", classification_rep)
```

```
Accuracy: 0.8488

Confusion Matrix:
 [[8397  123]
 [1389   91]]

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

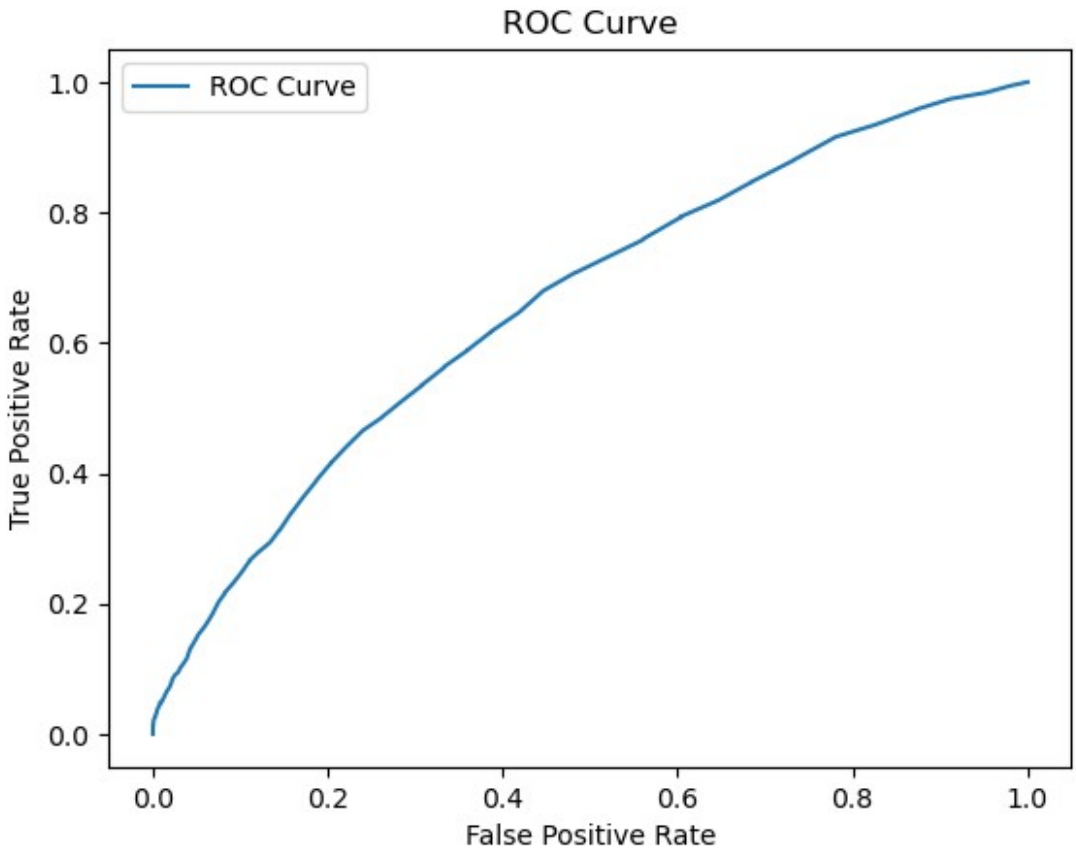
```
              0       0.86      0.99      0.92      8520
              1       0.43      0.06      0.11      1480

    accuracy                           0.85     10000
   macro avg       0.64      0.52      0.51     10000
weighted avg       0.79      0.85      0.80     10000
```

```
import seaborn as sns
sns.heatmap(conf_matrix, annot=True, cmap='Blues', fmt='d',
            xticklabels=['Predicted 0', 'Predicted 1'],
            yticklabels=['Actual 0', 'Actual 1'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

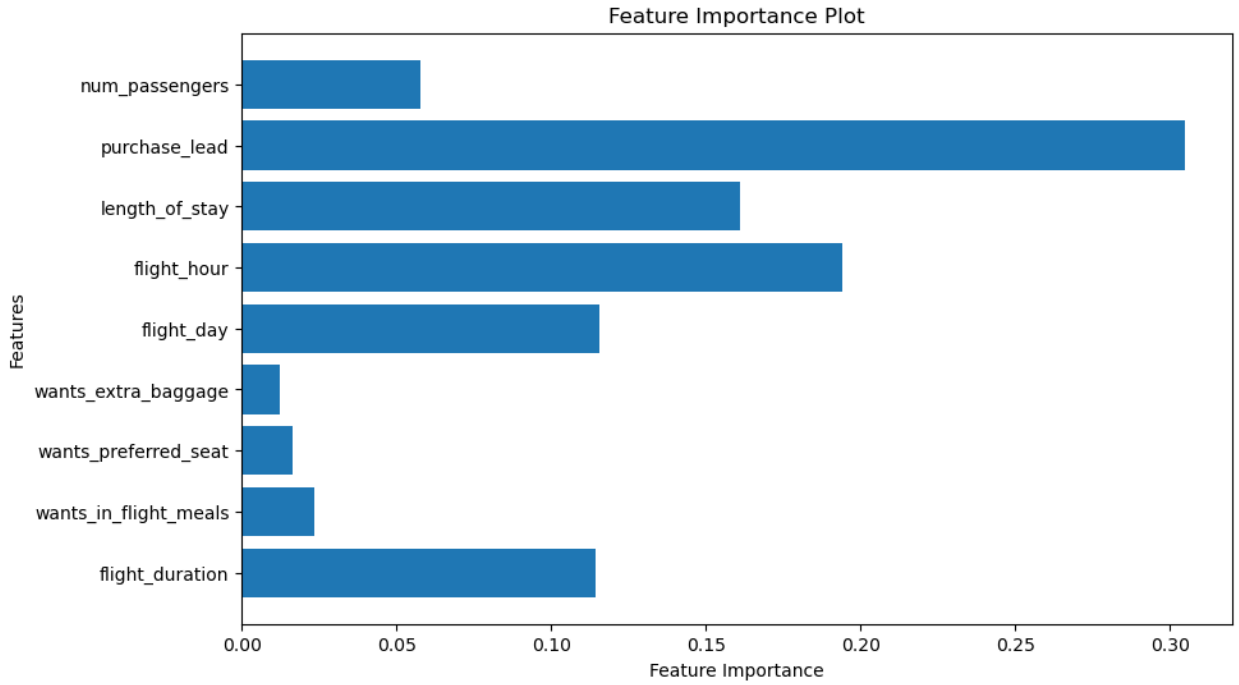



```
from sklearn.metrics import roc_curve, roc_auc_score
fpr, tpr, thresholds = roc_curve(y_test,
rf_classifier.predict_proba(X_test)[:,1])
plt.plot(fpr, tpr, label='ROC Curve')
plt.xlabel('False Positive Rate')
```

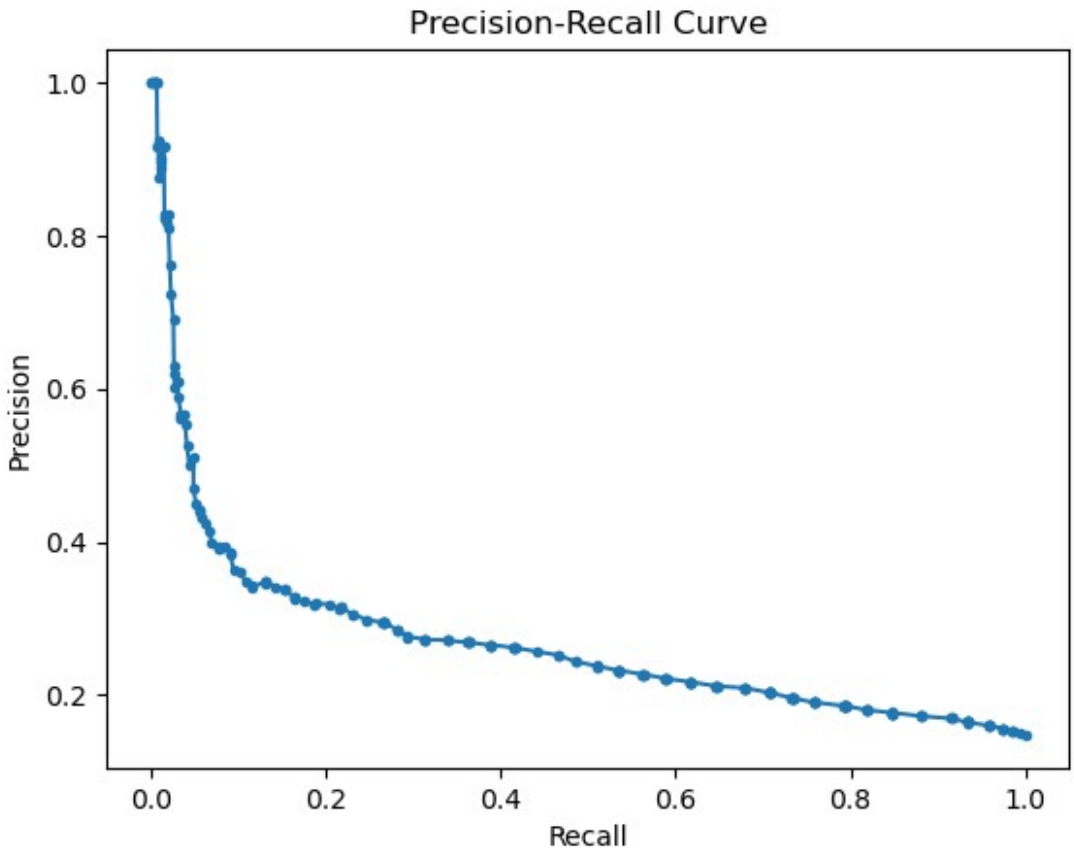
```
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend()
plt.show()
```




```
feature_importances = rf_classifier.feature_importances_
feature_names = X.columns
plt.figure(figsize=(10, 6))
plt.barh(range(len(feature_importances)), feature_importances,
align='center')
plt.yticks(range(len(feature_importances)), feature_names)
plt.xlabel('Feature Importance')
plt.ylabel('Features')
plt.title('Feature Importance Plot')
plt.gca().invert_yaxis()
plt.show()
```




```
from sklearn.metrics import precision_recall_curve
precision, recall, _ = precision_recall_curve(y_test,
rf_classifier.predict_proba(X_test)[:,1])
plt.plot(recall, precision, marker='.')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('Precision-Recall Curve')
plt.show()
```


Precision-Recall Curve
Precision
Recall
1.0
0.8
0.6
0.4
0.2
0.0
0.2
0.4
0.6
0.8
1.0