In [23]:

```
import numpy as np
import pandas as pd                                    # data processing
import matplotlib.pyplot as plt                        # visualization
import seaborn as sns
```

In [3]:

```
import warnings
warnings.filterwarnings('ignore')
```

Load the dataset

In [4]:

```
file_encoding = 'cp1252'         # set file_encoding to the file encoding (utf8, latin1, e
customer_data=pd.read_csv('D:/INTERNSHIP/customer_booking.csv', encoding=file_encoding)
customer_data
```

Out[4]:

	num_passengers	sales_channel	trip_type	purchase_lead	length_of_stay	flight_hour
0	2	Internet	RoundTrip	262	19	7
1	1	Internet	RoundTrip	112	20	3
2	2	Internet	RoundTrip	243	22	17
3	1	Internet	RoundTrip	96	31	4
4	2	Internet	RoundTrip	68	22	15
...	...	...	...	...	...	...
49995	2	Internet	RoundTrip	27	6	9
49996	1	Internet	RoundTrip	111	6	4
49997	1	Internet	RoundTrip	24	6	22
49998	1	Internet	RoundTrip	15	6	11
49999	1	Internet	RoundTrip	19	6	10

50000 rows × 14 columns

Data preprocessing

In [5]:

```
customer_data.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
```

According to the above information:

It seems that there are no missing values, which will be checked in more detail below. We have 14 features in which the target variable is booking complete. We have 5 categorical variables.

In [6]:

```
# Check missing value
customer_data.isnull().sum().to_frame('NaN value').T
```

Out[6]:

	num_passengers	sales_channel	trip_type	purchase_lead	length_of_stay	flight_hour
NaN value	0	0	0	0	0	0

Check each columns for detect syntax error or invalid values

In [7]:

```
# check count of unique values in each columns
for col in customer_data:
    print(f"{col}: {customer_data[col].nunique()}")
```

```
num_passengers: 9
sales_channel: 2
trip_type: 3
purchase_lead: 470
length_of_stay: 335
flight_hour: 24
flight_day: 7
route: 799
booking_origin: 104
wants_extra_baggage: 2
wants_preferred_seat: 2
wants_in_flight_meals: 2
flight_duration: 21
booking_complete: 2
```

In [8]:

```
customer_data.describe()
```

Out[8]:

	num_passengers	purchase_lead	length_of_stay	flight_hour	wants_extra_baggage	w
count	50000.000000	50000.000000	50000.00000	50000.00000	50000.000000	
mean	1.591240	84.940480	23.04456	9.06634	0.668780	
std	1.020165	90.451378	33.88767	5.41266	0.470657	
min	1.000000	0.000000	0.00000	0.00000	0.000000	
25%	1.000000	21.000000	5.00000	5.00000	0.000000	
50%	1.000000	51.000000	17.00000	9.00000	1.000000	
75%	2.000000	115.000000	28.00000	13.00000	1.000000	
max	9.000000	867.000000	778.00000	23.00000	1.000000	

In [9]:

```
customer_data['sales_channel'].value_counts()
```

Out[9]:

```
Internet    44382
Mobile       5618
Name: sales_channel, dtype: int64
```

In [10]:

```
# convert categoriacl feature to numerical:
customer_data['sales_channel'] = customer_data['sales_channel'].replace(['Internet', 'Mob
customer_data.head()
```

Out[10]:

	num_passengers	sales_channel	trip_type	purchase_lead	length_of_stay	flight_hour	flig
0	2	0	RoundTrip	262	19	7	
1	1	0	RoundTrip	112	20	3	
2	2	0	RoundTrip	243	22	17	
3	1	0	RoundTrip	96	31	4	
4	2	0	RoundTrip	68	22	15	

In [11]:

```
customer_data['trip_type'].value_counts()
```

Out[11]:

```
RoundTrip     49497
OneWay          387
CircleTrip      116
Name: trip_type, dtype: int64
```

In [12]:

```
customer_data['trip_type'] = customer_data['trip_type'].replace(['RoundTrip', 'OneWay', '
customer_data.head()
```

Out[12]:

	num_passengers	sales_channel	trip_type	purchase_lead	length_of_stay	flight_hour	fligh
0	2	0	0	262	19	7	
1	1	0	0	112	20	3	
2	2	0	0	243	22	17	
3	1	0	0	96	31	4	
4	2	0	0	68	22	15	

In [13]:

```
customer_data['flight_day'].value_counts()
```

Out[13]:

```
Mon    8102
Wed    7674
Tue    7673
Thu    7424
Fri    6761
Sun    6554
Sat    5812
Name: flight_day, dtype: int64
```

In [14]:

```
customer_data['flight_day'] = customer_data['flight_day'].replace(['Mon', 'Tue', 'Wed', '
                                                                   [0, 1, 2, 3, 4, 5, 6])
customer_data.head()
```

Out[14]:

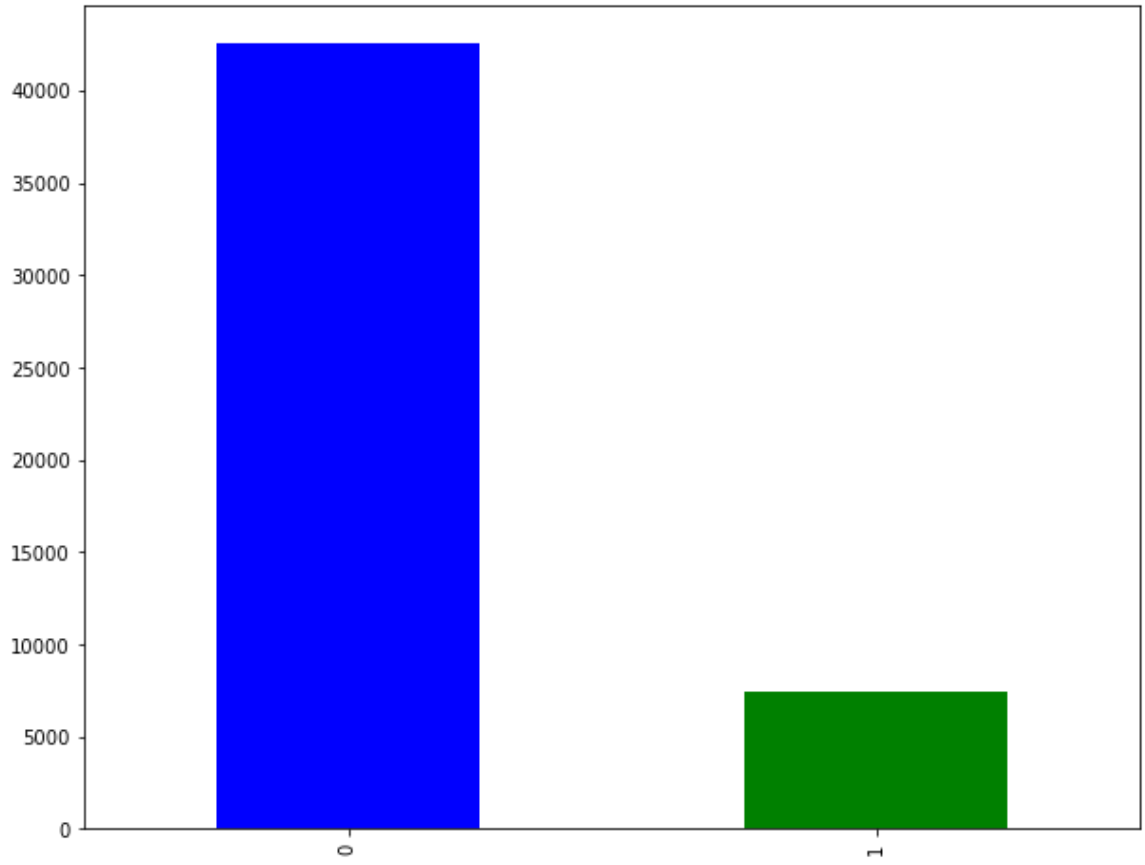
	num_passengers	sales_channel	trip_type	purchase_lead	length_of_stay	flight_hour	fligh
0	2	0	0	262	19	7	
1	1	0	0	112	20	3	
2	2	0	0	243	22	17	
3	1	0	0	96	31	4	
4	2	0	0	68	22	15	

In [15]:

```
bar = customer_data['booking_complete'].value_counts()
bar.plot(kind='bar', figsize=(10,8), color=['blue', 'green'])
```

Out[15]:

```
<AxesSubplot:>
```



In [16]:

```
customer_data.dtypes
```

Out[16]:

```
num_passengers             int64
sales_channel              int64
trip_type                  int64
purchase_lead              int64
length_of_stay             int64
flight_hour                int64
flight_day                 int64
route                     object
booking_origin            object
wants_extra_baggage        int64
wants_preferred_seat       int64
wants_in_flight_meals      int64
flight_duration          float64
booking_complete           int64
dtype: object
```

In [17]:

```
# splitting the dataset
X = customer_data.drop('booking_complete', axis=1)
Y = customer_data['booking_complete']
```

In [18]:

```
#changing object dtype to int dtype
for colname in X.select_dtypes("object"):
    X[colname], _ = X[colname].factorize()
```

In [19]:

```
X.dtypes
```

Out[19]:

```
num_passengers             int64
sales_channel              int64
trip_type                  int64
purchase_lead              int64
length_of_stay             int64
flight_hour                int64
flight_day                 int64
route                      int64
booking_origin             int64
wants_extra_baggage        int64
wants_preferred_seat       int64
wants_in_flight_meals      int64
flight_duration          float64
dtype: object
```

In [20]:

```
X.head()
```

Out[20]:

	num_passengers	sales_channel	trip_type	purchase_lead	length_of_stay	flight_hour	fligh
0	2	0	0	262	19	7	
1	1	0	0	112	20	3	
2	2	0	0	243	22	17	
3	1	0	0	96	31	4	
4	2	0	0	68	22	15	

Mutual information

In [21]:

```
from sklearn.feature_selection import mutual_info_classif
mi_score = mutual_info_classif(X,Y)
mi_score = pd.Series(mi_score, name="Mutual information Scores", index=X.columns)
mi_score = mi_score.sort_values(ascending=False)
mi_score
```

Out[21]:

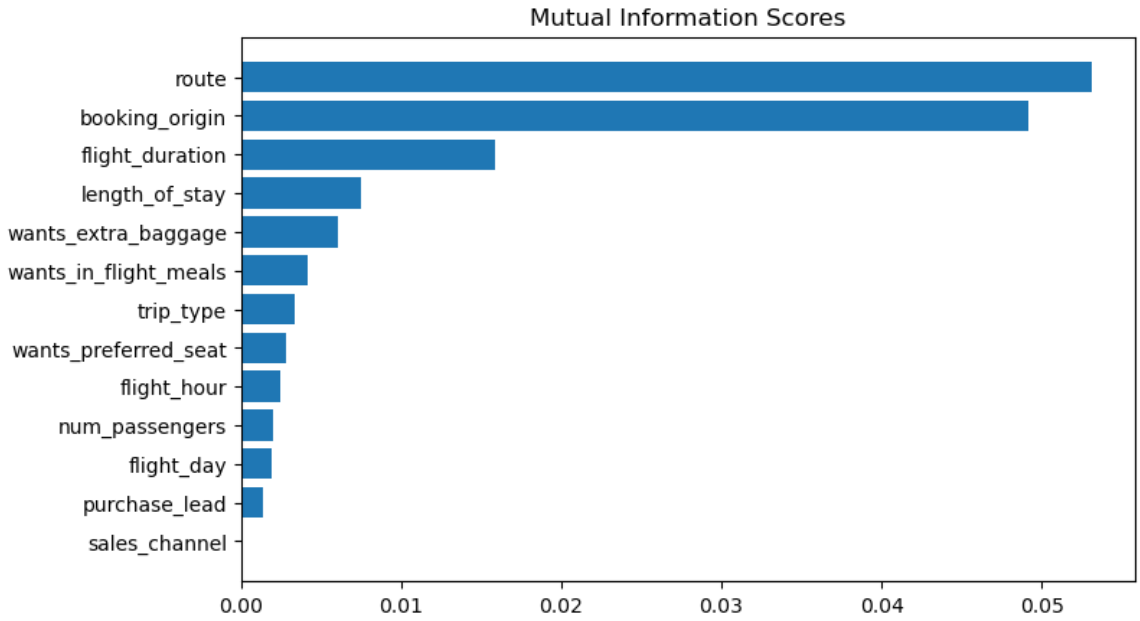
```
route                       0.053137
booking_origin              0.049195
flight_duration             0.015830
length_of_stay              0.007495
wants_extra_baggage         0.006076
wants_in_flight_meals       0.004127
trip_type                   0.003329
wants_preferred_seat        0.002811
flight_hour                 0.002495
num_passengers              0.002038
flight_day                  0.001893
purchase_lead               0.001412
sales_channel               0.000000
Name: Mutual information Scores, dtype: float64
```

Mutual information score is a measure of the strength of association between feature and the target. The MI score will fall in the range from 0 to ∞. The higher value, the closer connection between this feature and the target.

In [25]:

```
def plot_mi_scores(scores):
    scores = scores.sort_values(ascending=True)
    width = np.arange(len(scores))
    ticks = list(scores.index)
    plt.barh(width, scores)
    plt.yticks(width, ticks)
    plt.title("Mutual Information Scores")


plt.figure(dpi=100, figsize=(8, 5))
plot_mi_scores(mi_score)
```



We can see route, booking_origin, flight_duration, length_of_stay, wants_extra_baggage are the top 5 features which are dependant with booking_complete feature

In [26]:

```
# test train split
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.metrics import roc_auc_score
```

In [28]:

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3)
forest_model= RandomForestClassifier(random_state=1)
forest_model.fit(X_train, Y_train)
Y_pred= forest_model.predict(X_test)
```

In [29]:

```
print(classification_report(Y_test, Y_pred))
print(confusion_matrix(Y_test, Y_pred))
```

```
              precision    recall  f1-score   support

           0       0.87      0.98      0.92     12807
           1       0.54      0.11      0.18      2193

    accuracy                           0.86     15000
   macro avg       0.70      0.55      0.55     15000
weighted avg       0.82      0.86      0.81     15000

[[12596   211]
 [ 1948   245]]
```

In [30]:

```
print('AUC score: ',roc_auc_score(Y_test,Y_pred))
```

```
AUC score:  0.5476218706062017
```

Here we have an accuracy of 84%.