0.0.1 PREDICTING CUSTOMER BUYING BEHAVIOUR

My task is to build a predictive model to understand which factors influence customer booking.

[1]:
```
#import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

[2]:
```
#loading the British Airways dataset

data=pd.read_csv("E:/customer.csv", encoding='ISO-8859-1')
data.head()
```

[2]:
```
   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
```

Flight hours refers takeoff to landing.

Flight duration refers to the time spent on the ground for boarding.

0.0.2 Exploratory Data Analysis

[3]:
```python
data.shape
```

[3]:
```
(50000, 14)
```

[4]:
```python
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
```

From the above information, we can ensure that there is no null values present in our dataset.

[5]:
```python
#measure of dispersion
data.describe()
```

[5]:
```
       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

       wants_extra_baggage  wants_preferred_seat  wants_in_flight_meals  \
count         50000.000000          50000.000000           50000.000000
mean              0.668780              0.296960               0.427140
std               0.470657              0.456923               0.494668
min               0.000000              0.000000               0.000000
25%               0.000000              0.000000               0.000000
50%               1.000000              0.000000               0.000000
75%               1.000000              1.000000               1.000000
max               1.000000              1.000000               1.000000

       flight_duration  booking_complete
count     50000.000000      50000.000000
mean          7.277561          0.149560
std           1.496863          0.356643
min           4.670000          0.000000
25%           5.620000          0.000000
50%           7.570000          0.000000
75%           8.830000          0.000000
max           9.500000          1.000000
```

[6]:
```python
#view the categorical variable
print(data['sales_channel'].unique())
print(data['trip_type'].unique())
```

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

Roundtrip–>A ticket that allows a person to travel to one place and then return back to the place he or she left.

Onewaytrip–>the journey back from a destination.

Circletrip–> A circle trip is a return trip that usually includes multiple stops along the route of travel before returning to the point of origin.

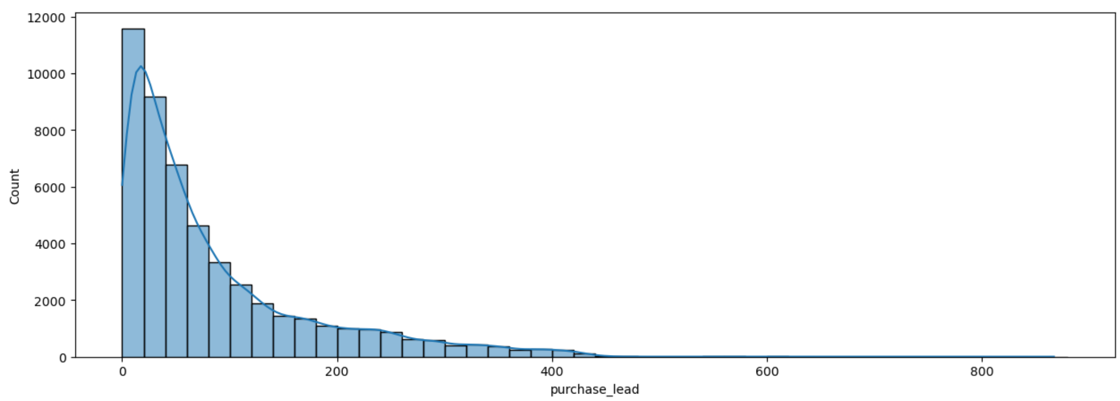
[7]:
```python
print(data['sales_channel'].value_counts())
print(data['trip_type'].value_counts())
```

```
Internet    44382
Mobile       5618
```

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

[8]:
```
plt.figure(figsize=(15,5))
sns.histplot(data,x='purchase_lead',binwidth=20,kde=True)
```

[8]:
```
<AxesSubplot:xlabel='purchase_lead', ylabel='Count'>
```



The graph shows more number of people showing interest to book ticket before a month of journey.

[9]:
```
df_final = data
```

[10]:
```
#applying OneHotEncoder for the desirable variable

from sklearn.preprocessing import OneHotEncoder

#create instance of one hot encoder
encoder =OneHotEncoder(handle_unknown = 'ignore')

#one hot encode for  sales channel
encoder_df = pd.DataFrame(encoder.fit_transform(data[['sales_channel']]).
 ↪toarray())
encoder_df = encoder_df.rename(columns={0:'Internet', 1:'Mobile'})
df_final = df_final.join(encoder_df)


#one hot encode for trip type
encoder_df = pd.DataFrame(encoder.fit_transform(data[['trip_type']]).toarray())
```

```
encoder_df = encoder_df.rename(columns={0:'RoundTrip', 1:'OneWayTrip',2:
↪'CircleTrip'})
df_final = df_final.join(encoder_df)
```

[11]:
```
#drop categorical values
df_final.
↪drop(['sales_channel','trip_type','flight_day','booking_origin','route'],␣
↪axis=1, inplace=True)
```

[12]:
```
#target variable
label =data['booking_complete']
```

[13]:
```
df_final = df_final.drop('booking_complete',axis=1)
```

[14]:
```
df_final.head()
```

[14]:
```
   num_passengers  purchase_lead  length_of_stay  flight_hour  \
0               2            262              19            7
1               1            112              20            3
2               2            243              22           17
3               1             96              31            4
4               2             68              22           15

   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  Internet  Mobile  RoundTrip  OneWayTrip  CircleTrip
0             5.52       1.0     0.0        0.0         0.0         1.0
1             5.52       1.0     0.0        0.0         0.0         1.0
2             5.52       1.0     0.0        0.0         0.0         1.0
3             5.52       1.0     0.0        0.0         0.0         1.0
4             5.52       1.0     0.0        0.0         0.0         1.0
```

[15]:
```
from sklearn.preprocessing import StandardScaler

#create a standard scaler object
scaler = StandardScaler()

#fit and transform the data
scaled_df = scaler.fit_transform(df_final)
```

[16]:
```
#create a dataframe of scaled data
scaled_df = pd.DataFrame(scaled_df, columns = df_final.columns)
```

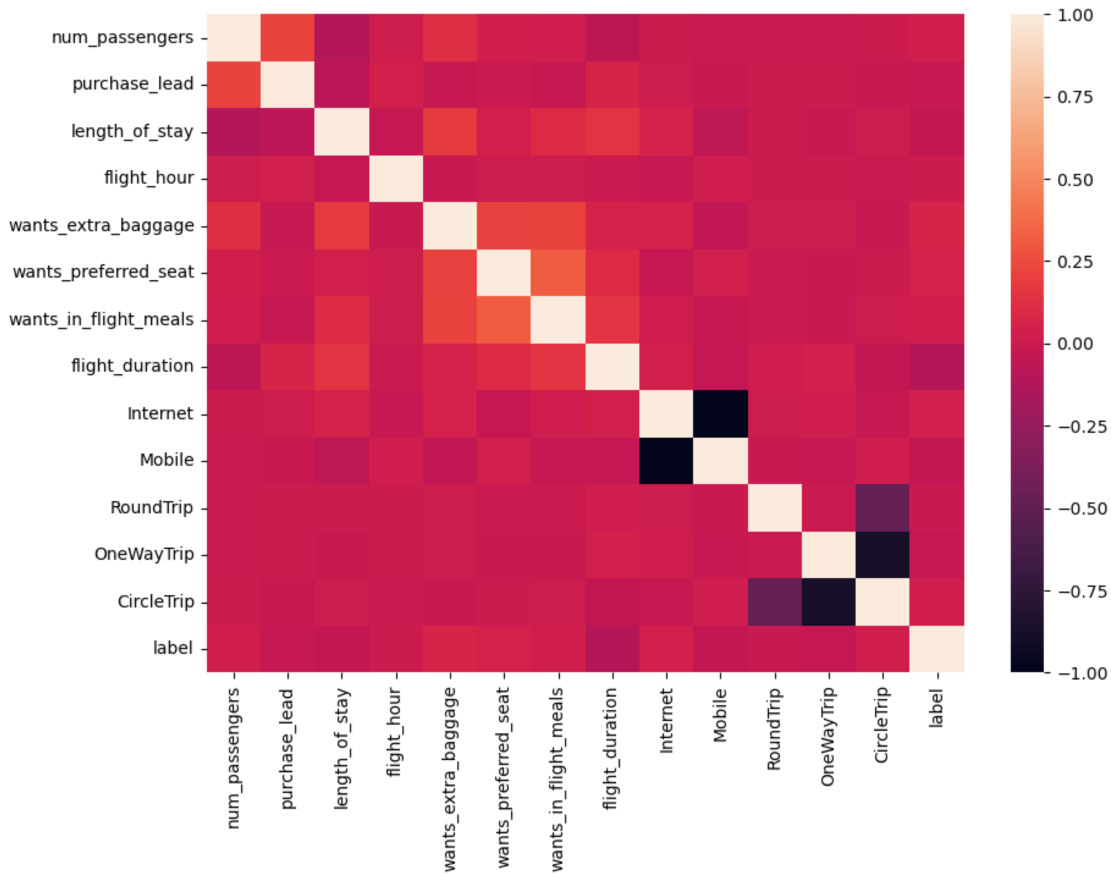
[17]:
```
#add the labels back to the dataframe
scaled_df['label']= label
```

[18]:
```
scaled_df.head()
```

[18]:
```
   num_passengers  purchase_lead  length_of_stay  flight_hour  \
0        0.400684       1.957530       -0.119353    -0.381764
1       -0.579559       0.299164       -0.089844    -1.120780
2        0.400684       1.747470       -0.030824     1.465775
3       -0.579559       0.122272        0.234761    -0.936026
4        0.400684      -0.187290       -0.030824     1.096267

   wants_extra_baggage  wants_preferred_seat  wants_in_flight_meals  \
0             0.703747             -0.649919              -0.863497
1            -1.420965             -0.649919              -0.863497
2             0.703747              1.538654              -0.863497
3            -1.420965             -0.649919               1.158082
4             0.703747             -0.649919               1.158082

   flight_duration  Internet    Mobile  RoundTrip  OneWayTrip  CircleTrip  \
0        -1.174175  0.355785 -0.355785  -0.048222    -0.08832    0.100808
1        -1.174175  0.355785 -0.355785  -0.048222    -0.08832    0.100808
2        -1.174175  0.355785 -0.355785  -0.048222    -0.08832    0.100808
3        -1.174175  0.355785 -0.355785  -0.048222    -0.08832    0.100808
4        -1.174175  0.355785 -0.355785  -0.048222    -0.08832    0.100808

   label
0      0
1      0
2      0
3      0
4      0
```

[19]:
```
corr = scaled_df.corr()

plt.figure(figsize=(10,7))
#plot the heatmap
sns.heatmap(corr)
```

[19]: <AxesSubplot:>



The person asking preferred seat is more likely wants for flight meals

0.1 Train Test Split

[20]:
```
from sklearn.model_selection import train_test_split

x = scaled_df.iloc[:,:-1]
y = scaled_df['label']

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.20,␣
↪random_state=0)
```

[21]:
```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()
rfc.fit(x_train,y_train)
result=rfc.score(x_test, y_test)
print(result)
```

```
0.8396
```

[22]:
```
y_pred_rfc = rfc.predict(x)
y_pred_rfc[2000:2010]
```

[22]: array([0, 0, 0, 0, 0, 0, 1, 0, 0, 0], dtype=int64)

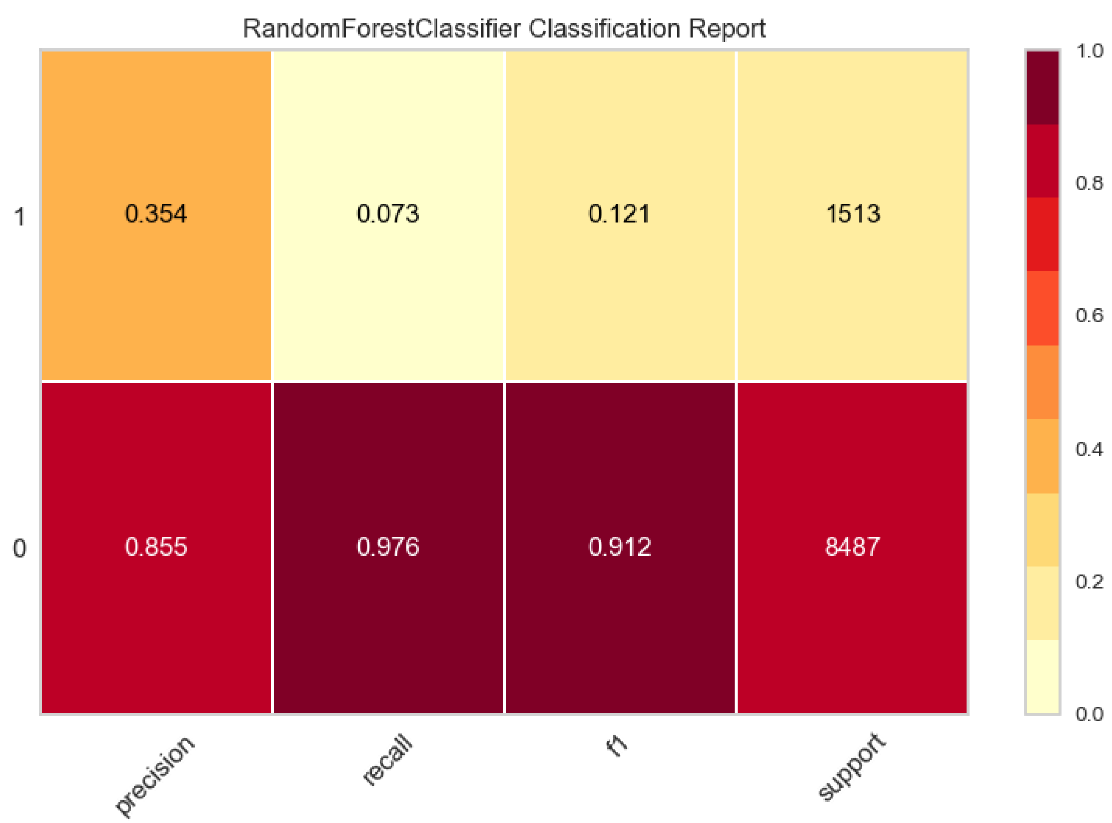
[23]:
```
#accuracy of our classification

from sklearn.metrics import accuracy_score
score = accuracy_score(y,y_pred_rfc)
```

[24]:
```
print(score)
```

```
0.9669
```

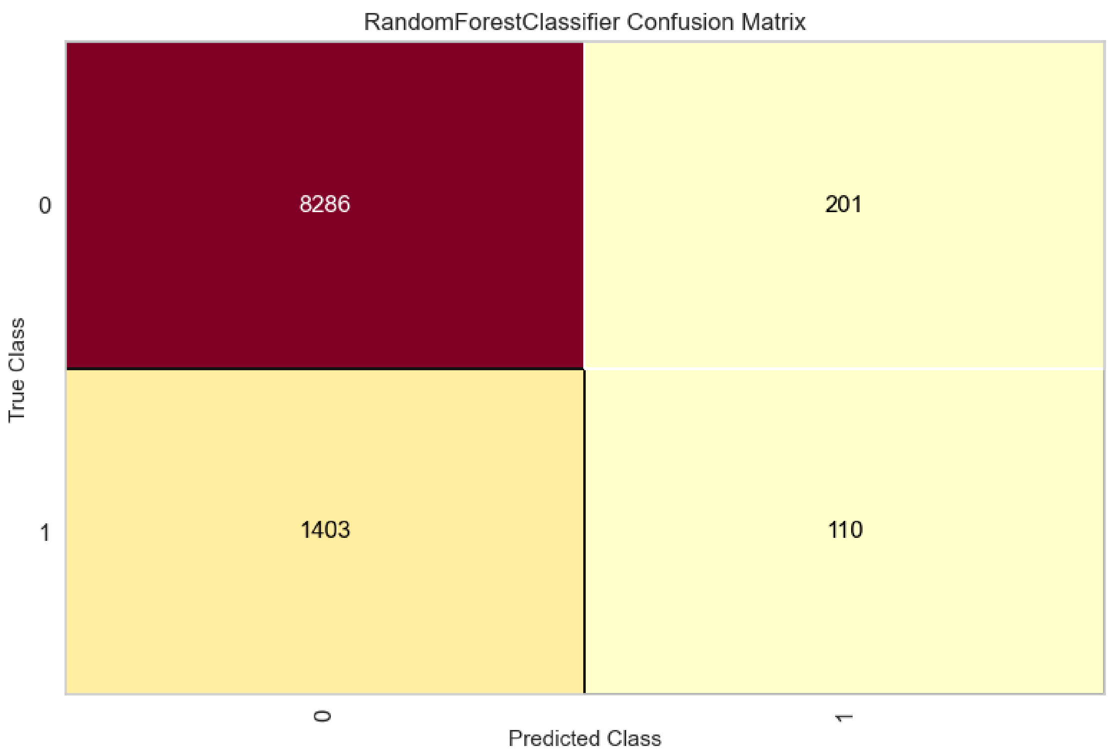
[25]:
```
from yellowbrick.classifier import ClassificationReport
vizualizer = ClassificationReport(rfc, classes=[0,1], support=True)
vizualizer.fit(x_train, y_train)
vizualizer.score(x_test, y_test)
vizualizer.show();
```

```
C:\Users\yukym\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but RandomForestClassifier was fitted with
feature names
  warnings.warn(
```



[26]:
```
from yellowbrick.classifier import ConfusionMatrix
cm = ConfusionMatrix(
    rfc, classes=[0,1],
    percent=False)
cm.fit(x_train, y_train)
cm.score(x_test, y_test)
cm.show();
```

```
C:\Users\yukym\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but RandomForestClassifier was fitted with
feature names
  warnings.warn(
```



From the Confusion Matrix we can say that our model is not predicting well. Training data is overfitted

0.2 Imbalanced dataset

[27]:
```
scaled_df.label.value_counts()
```

[27]:
```
0    42522
1     7478
Name: label, dtype: int64
```

[28]:
```
#create dataframe having all labels 0 with 10000 samples
scaled_df_0 = scaled_df[scaled_df.label == 0].sample(n=8000)
```

[29]:
```
#concatenate the two dataframe, one having all labels 0 other having all labels␣
↪as 1
scaled_df_new = pd.concat([scaled_df[scaled_df.label==1], scaled_df_0],␣
↪ignore_index=True)
```

[30]:
```
#shuffle the dataframe rows
scaled_df_new = scaled_df_new.sample(frac=1).reset_index(drop=True)
```

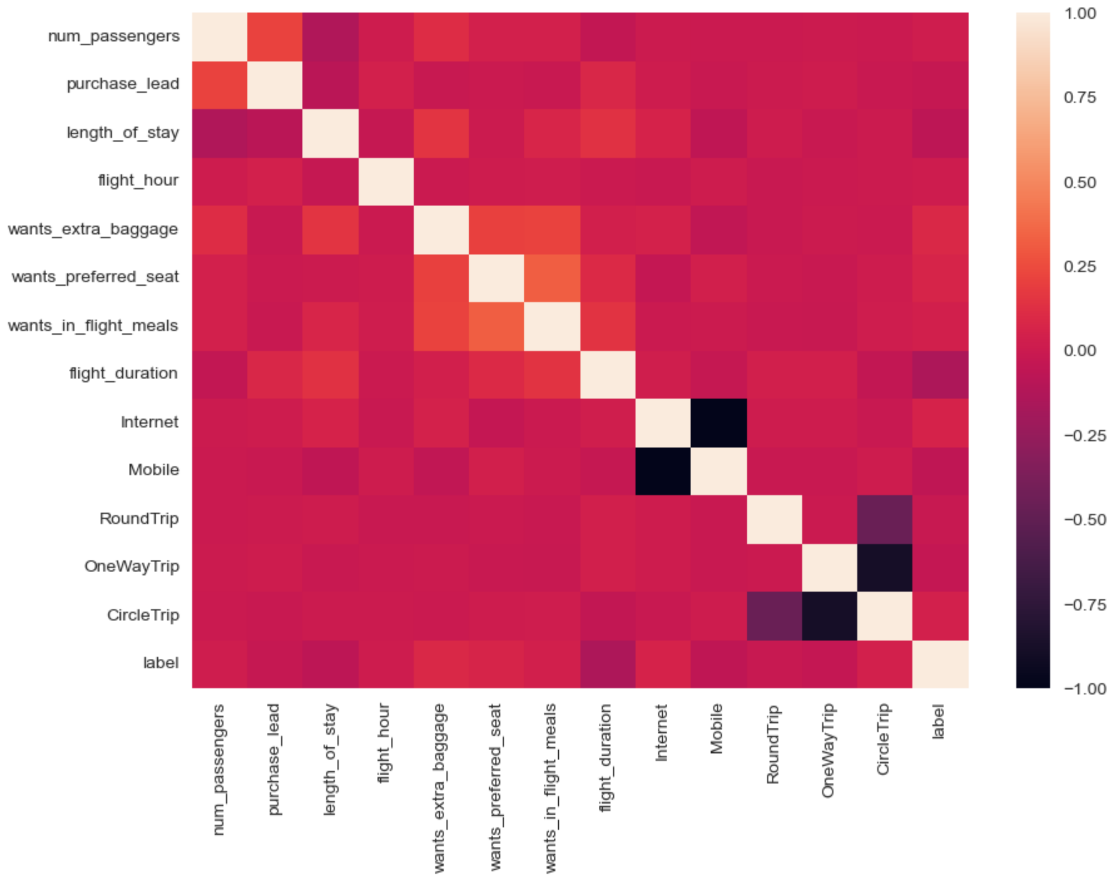
[31]:
```
scaled_df_new
```

[31]:
```
       num_passengers  purchase_lead  length_of_stay  flight_hour  \
0           -0.579559       0.365499       -0.502977     1.281021
1            1.380928       1.548466        0.264271     0.172497
2            1.380928      -0.264681       -0.532487     1.096267
3           -0.579559       0.387610       -0.001315    -0.751272
4           -0.579559       0.066993       -0.178372    -1.120780
…                 …              …               …           …
15473       -0.579559      -0.894860       10.090952    -0.566518
15474       -0.579559       1.028845       -0.502977    -1.675042
15475       -0.579559      -0.717967        0.470838    -1.305534
15476        2.361172      -0.717967       -0.502977    -0.381764
15477        0.400684       1.625857        0.382309    -0.936026

       wants_extra_baggage  wants_preferred_seat  wants_in_flight_meals  \
0                 0.703747             -0.649919               1.158082
1                 0.703747             -0.649919               1.158082
2                 0.703747              1.538654               1.158082
3                 0.703747              1.538654              -0.863497
4                 0.703747             -0.649919              -0.863497
…                      …                     …                      …
15473             0.703747             -0.649919               1.158082
15474             0.703747             -0.649919               1.158082
15475             0.703747             -0.649919              -0.863497
15476             0.703747             -0.649919              -0.863497
15477             0.703747              1.538654               1.158082

       flight_duration  Internet    Mobile  RoundTrip  OneWayTrip  CircleTrip  \
0             1.037139  0.355785 -0.355785  -0.048222    -0.08832    0.100808
1             1.037139  0.355785 -0.355785  -0.048222    -0.08832    0.100808
2            -1.107368  0.355785 -0.355785  -0.048222    -0.08832    0.100808
```

```
3             -1.174175  0.355785 -0.355785  -0.048222    -0.08832    0.100808
4             -1.688589  0.355785 -0.355785  -0.048222    -0.08832    0.100808
…                  …        …        …         …           …           …
15473          1.037139  0.355785 -0.355785  -0.048222    -0.08832    0.100808
15474          0.930248 -2.810688  2.810688  -0.048222    -0.08832    0.100808
15475         -1.688589  0.355785 -0.355785  -0.048222    -0.08832    0.100808
15476         -0.572911 -2.810688  2.810688  -0.048222    -0.08832    0.100808
15477          1.037139  0.355785 -0.355785  -0.048222    -0.08832    0.100808

       label
0          1
1          0
2          1
3          1
4          1
…      …
15473      0
15474      0
15475      1
15476      1
15477      0

[15478 rows x 14 columns]
```

[32]:
```
corr = scaled_df_new.corr()

plt.figure(figsize=(10,7))
#plot the heatmap
sns.heatmap(corr)
```

[32]: <AxesSubplot:>



0.3 Train_test_split

[33]:
```
from sklearn.model_selection import train_test_split

x = scaled_df_new.iloc[:,:-1]
y = scaled_df_new['label']

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.20,␣
↪random_state=0)
```

[34]:
```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()
rfc.fit(x_train,y_train)
result=rfc.score(x_test, y_test)
print(result)
```

```
0.6321059431524548
```

[35]:
```
y_pred_rfc = rfc.predict(x)
y_pred_rfc[2000:2010]
```

[35]:
```
array([0, 0, 0, 1, 0, 0, 1, 0, 1, 1], dtype=int64)
```

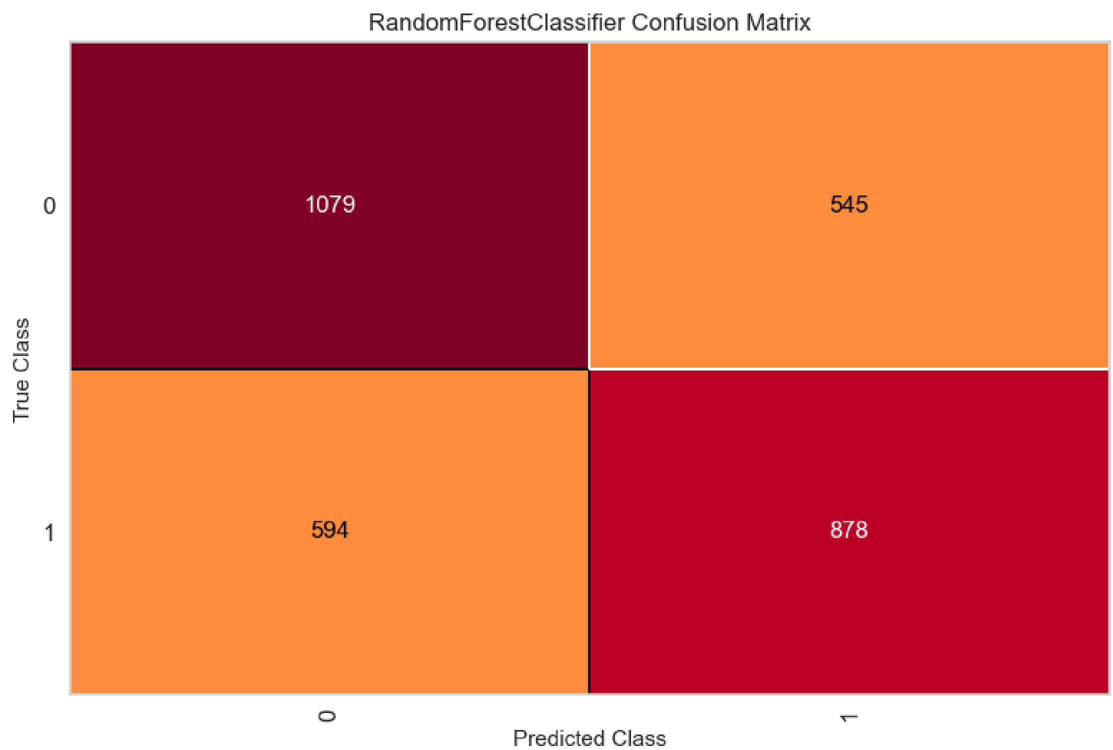
[36]:
```
#accuracy of our classification

from sklearn.metrics import accuracy_score
score = accuracy_score(y,y_pred_rfc)
```

[37]:
```
print(score)
```

```
0.9257009949605892
```

[38]:
```
from yellowbrick.classifier import ConfusionMatrix
cm = ConfusionMatrix(
    rfc, classes=[0,1],
    percent=False)
cm.fit(x_train, y_train)
cm.score(x_test, y_test)
cm.show();
```

```
C:\Users\yukym\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but RandomForestClassifier was fitted with
feature names
  warnings.warn(
```



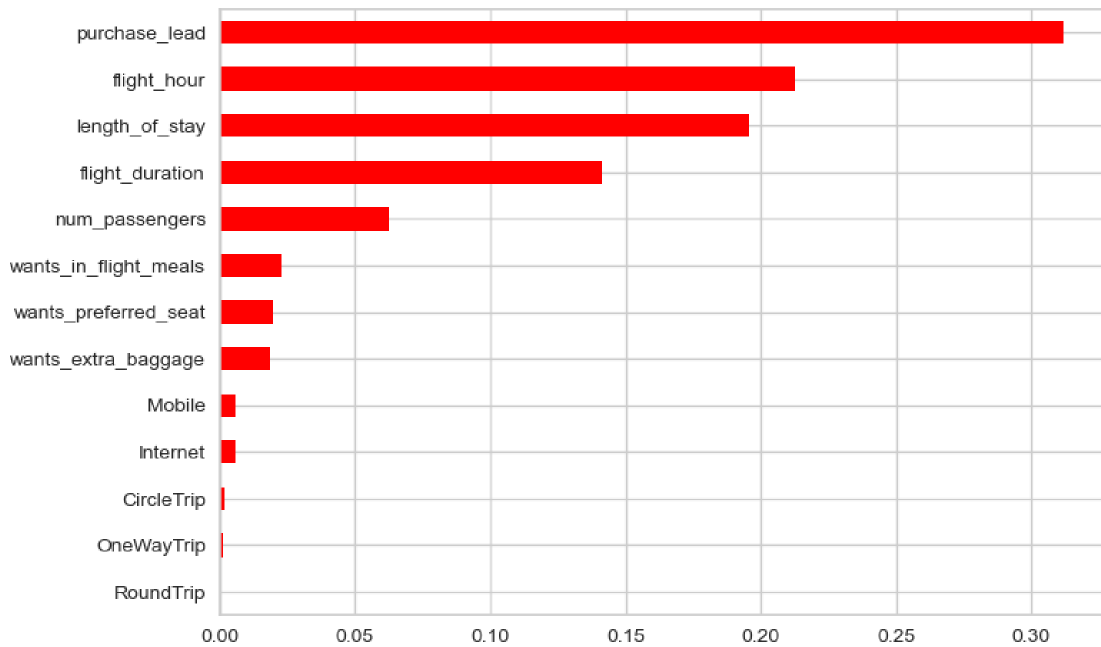
Compare to previous model this model done a better prediction.

[39]:
```
#finding which variable have more impact to the target variable
importance = rfc.feature_importances_
columns = x_train.columns#finding which variable have more impact to the target␣
↪variable
importance = rfc.feature_importances_
columns = x_train.columns
```

[40]:
```
rfc_cof = pd.Series(importance, columns)
rfc_cof
```

[40]:
```
num_passengers           0.062405
purchase_lead            0.312102
length_of_stay           0.195482
flight_hour              0.212663
wants_extra_baggage      0.018826
wants_preferred_seat     0.019688
wants_in_flight_meals    0.022649
flight_duration          0.141068
Internet                 0.005828
Mobile                   0.005887
RoundTrip                0.000332
OneWayTrip               0.001358
CircleTrip               0.001713
dtype: float64
```

[41]:
```
%matplotlib inline

rfc_cof.sort_values().plot.barh(color='red');
```



We can conclude that purchase lead more contribute towards customer booking.

```
[ ]:
```