Predictive Modelling

September 28, 2024

[3]:
```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

[4]:
```python
df = pd.read_csv("C:/Users/bendh/Desktop/data science/JN/
↪filtered_customer_booking.csv", encoding="ISO-8859-1")
df = df.drop(columns=['Unnamed: 0'])
df.head()
```

[4]:
```
   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            6  AKLDEL    New Zealand                    1
1            3            6  AKLDEL    New Zealand                    0
2           17            3  AKLDEL          India                    1
3            4            6  AKLDEL    New Zealand                    0
4           15            3  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
```

0.0.1 One Hot Encode

[5]:
```
#one hot encode categorical values
from sklearn.preprocessing import OneHotEncoder

df2 = df

#create instanc"e of one hot encoder
encoder = OneHotEncoder(handle_unknown='ignore')

#one hot encode Sales Channel
encoder_df = pd.DataFrame(encoder.fit_transform(df[["sales_channel"]]).
 ↪toarray())
encoder_df = encoder_df.rename(columns={0:'Internet', 1:'Phone'})
df2 = df2.join(encoder_df)

#one hot encode trip type
encoder_df = pd.DataFrame(encoder.fit_transform(df[["trip_type"]]).toarray())
encoder_df = encoder_df.rename(columns={0:'RoundTrip', 1:'OneWayTrip',2:
 ↪'CircleTrip'})
df2 = df2.join(encoder_df)
```

[6]:
```
#drop categorical columns now
df2.drop(['sales_channel', 'trip_type', 'booking_origin', 'route'], axis=1,␣
 ↪inplace = True)
```

[7]:
```
#store the label for supervised learning
label = df["booking_complete"]
```

[8]:
```
df2 = df2.drop("booking_complete", axis=1)
```

[9]:
```
df2
```

[9]:
```
       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
…                 …            …             …          …          …
49977               2             27               6            9           6
49978               1            111               6            4           7
49979               1             24               6           22           6
49980               1             15               6           11           1
49981               1             19               6           10           4

       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
…                   …                   …                   …
49977                 1                     0                     1
49978                 0                     0                     0
49979                 0                     0                     1
49980                 1                     0                     1
49981                 0                     1                     0

       flight_duration  Internet  Phone  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
…                  …       …    …        …         …         …
49977             5.62       1.0    0.0        0.0         0.0         1.0
49978             5.62       1.0    0.0        0.0         0.0         1.0
49979             5.62       1.0    0.0        0.0         0.0         1.0
49980             5.62       1.0    0.0        0.0         0.0         1.0
49981             5.62       1.0    0.0        0.0         0.0         1.0

[49982 rows x 14 columns]
```

0.1 Normalizing values

[10]:
```python
from sklearn.preprocessing import StandardScaler

#create a standard scaler object
scaler = StandardScaler()

#fit and transform the data
scaled_df = scaler.fit_transform(df2)
```

[11]:
```python
#create a dataframe of scaled data
scaled_df = pd.DataFrame(scaled_df, columns = df2.columns)
```

[12]:
```python
#add the labels back to the dataframe
scaled_df['label'] = label
```

[13]:
```python
scaled_df
```

[13]:
```
       num_passengers  purchase_lead  length_of_stay  flight_hour  flight_day  \
0            0.400769       1.971093       -0.119401    -0.381588    1.096876
1           -0.579424       0.302987       -0.089895    -1.120618    1.096876
```

2	0.400769	1.759799	-0.030885	1.465988	-0.408618
3	-0.579424	0.125056	0.234662	-0.935861	1.096876
4	0.400769	-0.186323	-0.030885	1.096473	-0.408618
…	…	…	…	…	…
49977	0.400769	-0.642272	-0.502969	-0.012073	1.096876
49978	-0.579424	0.291867	-0.502969	-0.935861	1.598707
49979	-0.579424	-0.675634	-0.502969	2.389776	1.096876
49980	-0.579424	-0.775721	-0.502969	0.357443	-1.412280
49981	-0.579424	-0.731238	-0.502969	0.172685	0.093214

	wants_extra_baggage	wants_preferred_seat	wants_in_flight_meals	\
0	0.703587	-0.650054	-0.863557	
1	-1.421288	-0.650054	-0.863557	
2	0.703587	1.538334	-0.863557	
3	-1.421288	-0.650054	1.158002	
4	0.703587	-0.650054	1.158002	
…	…	…	…	
49977	0.703587	-0.650054	1.158002	
49978	-1.421288	-0.650054	-0.863557	
49979	-1.421288	-0.650054	1.158002	
49980	0.703587	-0.650054	1.158002	
49981	-1.421288	1.538334	-0.863557	

	flight_duration	Internet	Phone	RoundTrip	OneWayTrip	CircleTrip	\
0	-1.174049	0.355786	-0.355786	-0.048231	-0.088336	0.100826	
1	-1.174049	0.355786	-0.355786	-0.048231	-0.088336	0.100826	
2	-1.174049	0.355786	-0.355786	-0.048231	-0.088336	0.100826	
3	-1.174049	0.355786	-0.355786	-0.048231	-0.088336	0.100826	
4	-1.174049	0.355786	-0.355786	-0.048231	-0.088336	0.100826	
…	…	…	…	…	…	…	
49977	-1.107240	0.355786	-0.355786	-0.048231	-0.088336	0.100826	
49978	-1.107240	0.355786	-0.355786	-0.048231	-0.088336	0.100826	
49979	-1.107240	0.355786	-0.355786	-0.048231	-0.088336	0.100826	
49980	-1.107240	0.355786	-0.355786	-0.048231	-0.088336	0.100826	
49981	-1.107240	0.355786	-0.355786	-0.048231	-0.088336	0.100826	

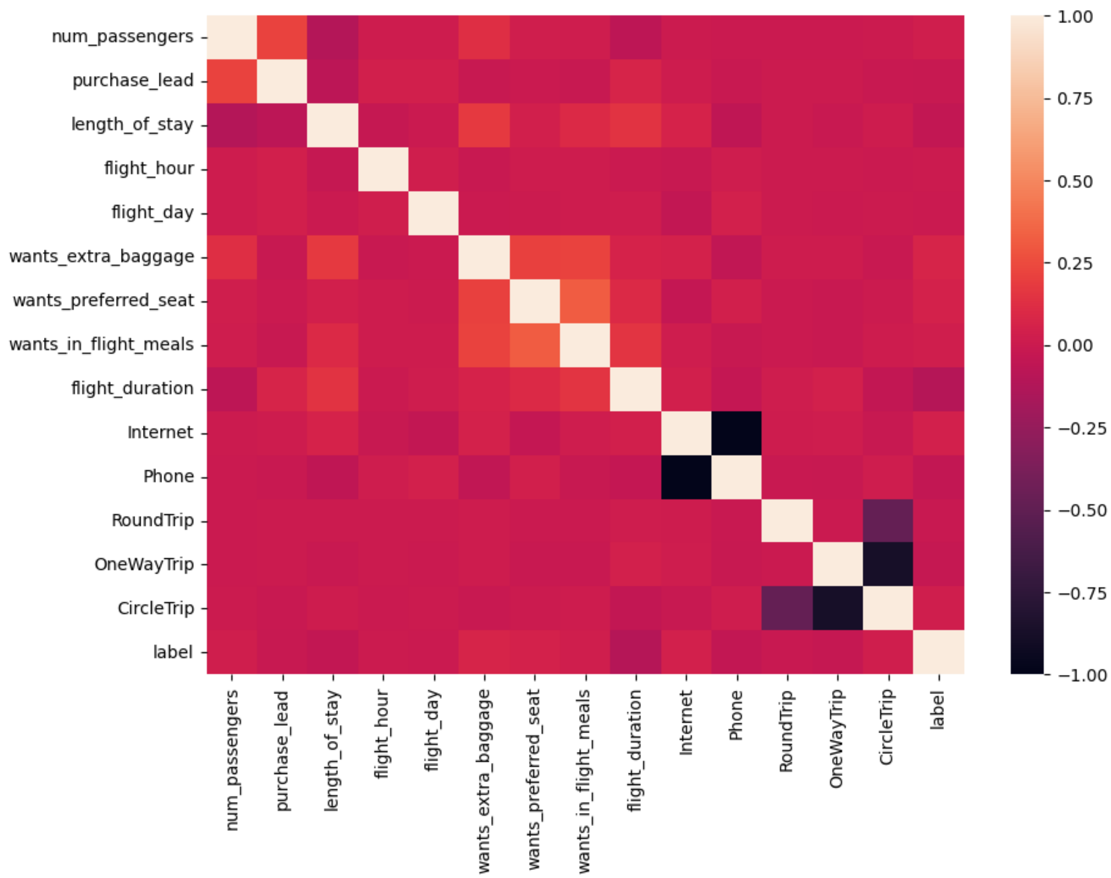
	label
0	0
1	0
2	0
3	0
4	0
…	…
49977	0
49978	0
49979	0
49980	0

```
49981      0

[49982 rows x 15 columns]
```

0.2 Correlation matrix

[14]:
```python
corr = scaled_df.corr()

plt.figure(figsize=(10,7))

#plot the heatmap
sns.heatmap(corr)
```

[14]:
```
<Axes: >
```



0.2.1 Splitting Train and Test Data

[15]:
```
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.to_numpy(), y.to_numpy(),␣
↪test_size=0.20, random_state = 42)
```

[16]:
```
!pip install yellowbrick

from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.metrics import f1_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.inspection import permutation_importance

from yellowbrick.classifier import ConfusionMatrix
from sklearn.model_selection import GridSearchCV, RepeatedStratifiedKFold
```

```
Requirement already satisfied: yellowbrick in
c:\users\bendh\appdata\local\programs\python\python39\lib\site-packages (1.5)
Requirement already satisfied: matplotlib!=3.0.0,>=2.0.2 in
c:\users\bendh\appdata\local\programs\python\python39\lib\site-packages (from
yellowbrick) (3.9.1.post1)
Requirement already satisfied: scipy>=1.0.0 in
c:\users\bendh\appdata\local\programs\python\python39\lib\site-packages (from
yellowbrick) (1.13.1)
Requirement already satisfied: scikit-learn>=1.0.0 in
c:\users\bendh\appdata\local\programs\python\python39\lib\site-packages (from
yellowbrick) (1.5.1)
Requirement already satisfied: numpy>=1.16.0 in
c:\users\bendh\appdata\local\programs\python\python39\lib\site-packages (from
yellowbrick) (1.26.4)
Requirement already satisfied: cycler>=0.10.0 in
c:\users\bendh\appdata\local\programs\python\python39\lib\site-packages (from
yellowbrick) (0.12.1)
Requirement already satisfied: contourpy>=1.0.1 in
c:\users\bendh\appdata\local\programs\python\python39\lib\site-packages (from
matplotlib!=3.0.0,>=2.0.2->yellowbrick) (1.2.1)
Requirement already satisfied: fonttools>=4.22.0 in
c:\users\bendh\appdata\local\programs\python\python39\lib\site-packages (from
matplotlib!=3.0.0,>=2.0.2->yellowbrick) (4.53.1)
Requirement already satisfied: kiwisolver>=1.3.1 in
c:\users\bendh\appdata\local\programs\python\python39\lib\site-packages (from
```

```
matplotlib!=3.0.0,>=2.0.2->yellowbrick) (1.4.5)
Requirement already satisfied: packaging>=20.0 in
c:\users\bendh\appdata\local\programs\python\python39\lib\site-packages (from
matplotlib!=3.0.0,>=2.0.2->yellowbrick) (24.1)
Requirement already satisfied: pillow>=8 in
c:\users\bendh\appdata\local\programs\python\python39\lib\site-packages (from
matplotlib!=3.0.0,>=2.0.2->yellowbrick) (10.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in
c:\users\bendh\appdata\local\programs\python\python39\lib\site-packages (from
matplotlib!=3.0.0,>=2.0.2->yellowbrick) (3.1.2)
Requirement already satisfied: python-dateutil>=2.7 in
c:\users\bendh\appdata\local\programs\python\python39\lib\site-packages (from
matplotlib!=3.0.0,>=2.0.2->yellowbrick) (2.9.0.post0)
Requirement already satisfied: importlib-resources>=3.2.0 in
c:\users\bendh\appdata\local\programs\python\python39\lib\site-packages (from
matplotlib!=3.0.0,>=2.0.2->yellowbrick) (6.4.0)
Requirement already satisfied: joblib>=1.2.0 in
c:\users\bendh\appdata\local\programs\python\python39\lib\site-packages (from
scikit-learn>=1.0.0->yellowbrick) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in
c:\users\bendh\appdata\local\programs\python\python39\lib\site-packages (from
scikit-learn>=1.0.0->yellowbrick) (3.5.0)
Requirement already satisfied: zipp>=3.1.0 in
c:\users\bendh\appdata\local\programs\python\python39\lib\site-packages (from
importlib-resources>=3.2.0->matplotlib!=3.0.0,>=2.0.2->yellowbrick) (3.19.2)
Requirement already satisfied: six>=1.5 in
c:\users\bendh\appdata\local\programs\python\python39\lib\site-packages (from
python-dateutil>=2.7->matplotlib!=3.0.0,>=2.0.2->yellowbrick) (1.16.0)
```

[17]:
```python
#create functions to fit and predict the values of wether customer would␣
↪complete the booking or not
#create functions with metrics to evaluate the model prediction

#check how well the model is performing on known data
def model_fit_predict(model, X, y, X_predict):
    model.fit(X,y)
    return model.predict(X_predict)

def acc_score(y_true, y_pred):
    return accuracy_score(y_true, y_pred)

def pre_score(y_true, y_pred):
    return precision_score(y_true, y_pred)

def f_score(y_true, y_pred):
    return f1_score(y_true, y_pred)
```

1 Random Forest Classifier

[18]:
```
#create an isntance of the classifier and fit the training data
clf_rf = RandomForestClassifier(max_depth = 50, min_samples_split= 5,␣
 ↪random_state= 0)
```

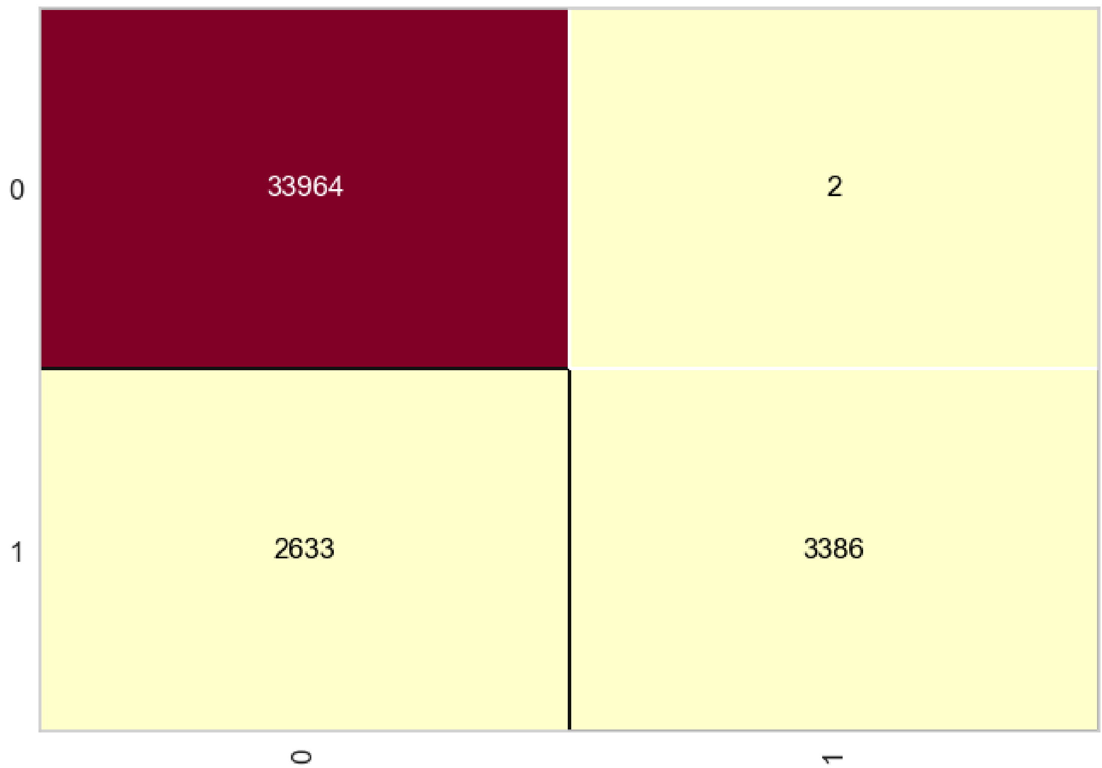
1.0.1 Checking Training Accuracy

[19]:
```
y_pred_train = model_fit_predict(clf_rf, X_train, y_train, X_train)
set(y_pred_train)

#f1 score for training data : It balances both false positives and false␣
 ↪negatives
f1 = round(f1_score(y_train, y_pred_train),2)

#accuracy score for training data : the ratio of correctly predicted instances␣
 ↪to total instances.
acc = round(accuracy_score(y_train, y_pred_train),2)

#precision score for training data : how many of the predicted positive cases␣
 ↪were actually correct.
pre = round(precision_score(y_train, y_pred_train),2)

print(f"Accuracy, precision and f1-score for training data are {acc}, {pre} and␣
 ↪{f1} respectively")
```

```
Accuracy, precision and f1-score for training data are 0.93, 1.0 and 0.72
respectively
```

[20]:
```
# confusion matrix shows the number of correct and incorrect predictions for␣
 ↪each class.
cm = ConfusionMatrix(clf_rf, classes=[0,1])
cm.fit(X_train, y_train)

cm.score(X_train, y_train)
```

[20]:
```
0.9341002876078529
```



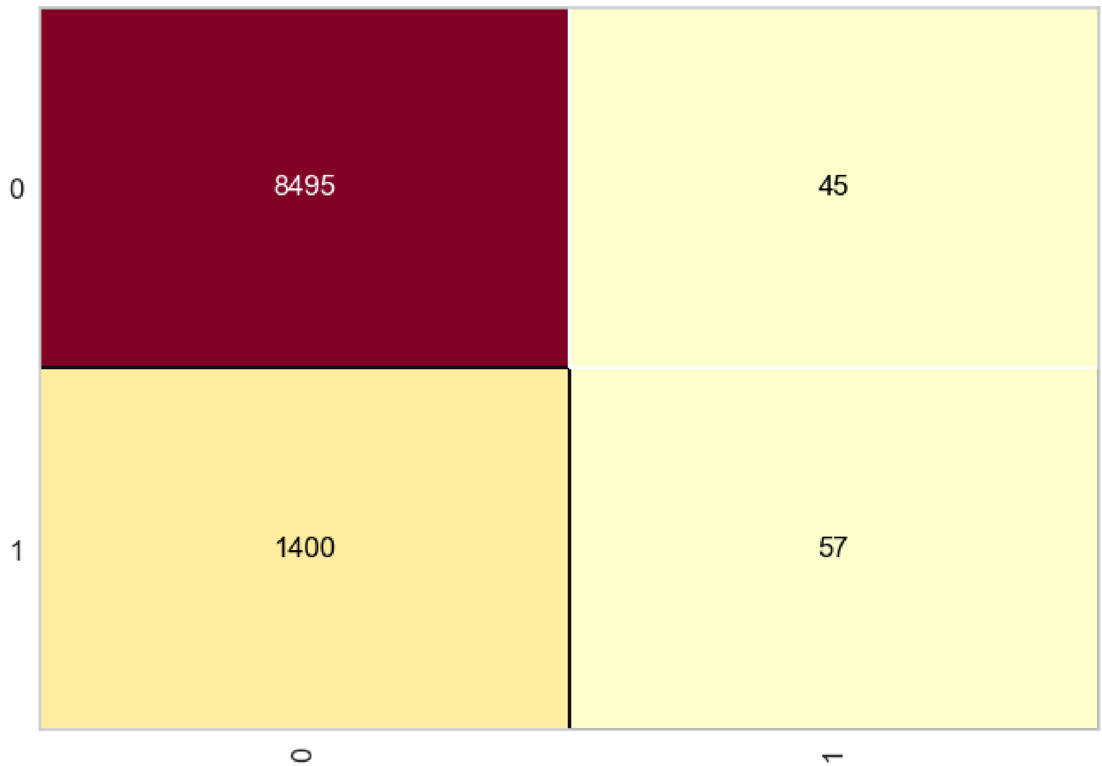
1.0.2 Checking Testing accuracy

```
[21]: #create an instance of the classifier and fit the training data
clf_rf = RandomForestClassifier(max_depth =50 ,␣
 ↪min_samples_split=5,random_state=0)

y_pred_test = model_fit_predict(clf_rf, X_train, y_train, X_test)

#f1 score for training data
f1 = round(f1_score(y_test, y_pred_test),2)

#accuracy score for training data
acc = round(accuracy_score(y_test, y_pred_test),2)

#precision score for training data
pre = round(precision_score(y_test, y_pred_test),2)

print(f"Accuracy, precision and f1-score for training data are {acc}, {pre} and␣
 ↪{f1} respectively")
```

```
Accuracy, precision and f1-score for training data are 0.86, 0.56 and 0.07
respectively
```

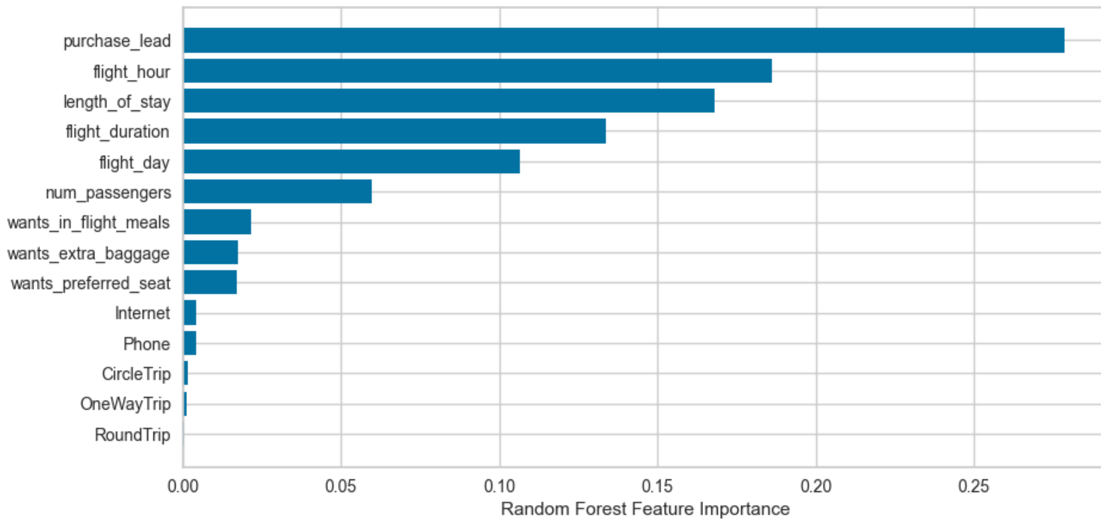
[22]:
```
cm = ConfusionMatrix(clf_rf, classes=[0,1])
cm.fit(X_train, y_train)

cm.score(X_test, y_test)
```

[22]: 0.8554566369910973



[23]:
```
plt.figure(figsize=(10,5))
sorted_idx = clf_rf.feature_importances_.argsort()
plt.barh(scaled_df.iloc[:,:-1].columns[sorted_idx], clf_rf.
  ↪feature_importances_[sorted_idx])
plt.xlabel("Random Forest Feature Importance")
```

[23]: Text(0.5, 0, 'Random Forest Feature Importance')



[24]:
```
# One major problem behind getting low F1 score is imbalanced dataset. We have␣
↪higher entries that are classified 0 than 1.
#We could reduce the number of entries that are classified 0 to be equal around␣
↪the number of entries that are classified as 1.
```

1.0.3 Balancing the dataset

[25]:
```
scaled_df.label.value_counts()
```

[25]:
```
label
0    42506
1     7476
Name: count, dtype: int64
```

[26]:
```
#create a dataframe having all labels 0 with 10000 samples
scaled_df_0 = scaled_df[scaled_df.label ==0].sample(n=8000)
```

[27]:
```
#concatenate the two dataframee, one havng all labels 0 and other having all␣
↪labels as 1
scaled_df_new = pd.concat([scaled_df[scaled_df.label==1], scaled_df_0],␣
↪ignore_index=True)
```

[28]:
```
#shuffle the dataframe rows
scaled_df_new = scaled_df_new.sample(frac = 1).reset_index(drop=True)
```

[29]:
```
scaled_df_new
```

[29]:
```
       num_passengers  purchase_lead  length_of_stay  flight_hour  flight_day  \
0            2.361155      -0.030634       -0.030885     0.726958    0.595045
```

```
1          0.400769      1.092557     -0.502969   -0.566346    0.595045
2          0.400769      2.971956      0.175652   -0.935861   -0.408618
3         -0.579424     -0.909169      0.028126   -0.012073    1.096876
4          0.400769      0.013849     -0.119401   -0.935861    0.595045
…               …            …             …           …           …
15471     -0.579424     -0.486582      1.119820    0.357443   -1.412280
15472     -0.579424     -0.920290     -0.650495   -0.935861    0.595045
15473      0.400769     -0.764600     -0.532474    1.281231    0.595045
15474      0.400769     -0.030634     -0.591484   -0.935861    0.595045
15475      1.380962      0.369712     -0.532474   -0.566346    0.093214

       wants_extra_baggage  wants_preferred_seat  wants_in_flight_meals  \
0                 0.703587              1.538334               1.158002
1                 0.703587             -0.650054              -0.863557
2                 0.703587              1.538334               1.158002
3                 0.703587             -0.650054              -0.863557
4                 0.703587              1.538334               1.158002
…                      …                     …                      …
15471             0.703587             -0.650054               1.158002
15472            -1.421288             -0.650054              -0.863557
15473             0.703587              1.538334               1.158002
15474            -1.421288             -0.650054              -0.863557
15475             0.703587             -0.650054               1.158002

       flight_duration  Internet     Phone  RoundTrip  OneWayTrip  CircleTrip  \
0            -1.107240 -2.810681  2.810681  -0.048231   -0.088336    0.100826
1            -0.185282  0.355786 -0.355786  -0.048231   -0.088336    0.100826
2             1.037314  0.355786 -0.355786  -0.048231   -0.088336    0.100826
3            -1.708517  0.355786 -0.355786  -0.048231   -0.088336    0.100826
4            -1.107240  0.355786 -0.355786  -0.048231   -0.088336    0.100826
…                  …         …        …          …           …           …
15471        -0.439155  0.355786 -0.355786  -0.048231   -0.088336    0.100826
15472        -1.741921  0.355786 -0.355786  -0.048231   -0.088336    0.100826
15473        -0.439155  0.355786 -0.355786  -0.048231   -0.088336    0.100826
15474        -0.439155  0.355786 -0.355786  -0.048231   -0.088336    0.100826
15475        -0.185282  0.355786 -0.355786  -0.048231   -0.088336    0.100826

       label
0          1
1          0
2          1
3          1
4          0
…      …
15471      0
15472      1
15473      1
```

```
15474      0
15475      1

[15476 rows x 15 columns]
```

[30]:
```
X = scaled_df_new.iloc[:,:-1]
y = scaled_df_new['label']

X_train, X_test, y_train, y_test = train_test_split(X.to_numpy(), y.to_numpy(),␣
 ↪test_size=0.20, random_state=42)
```

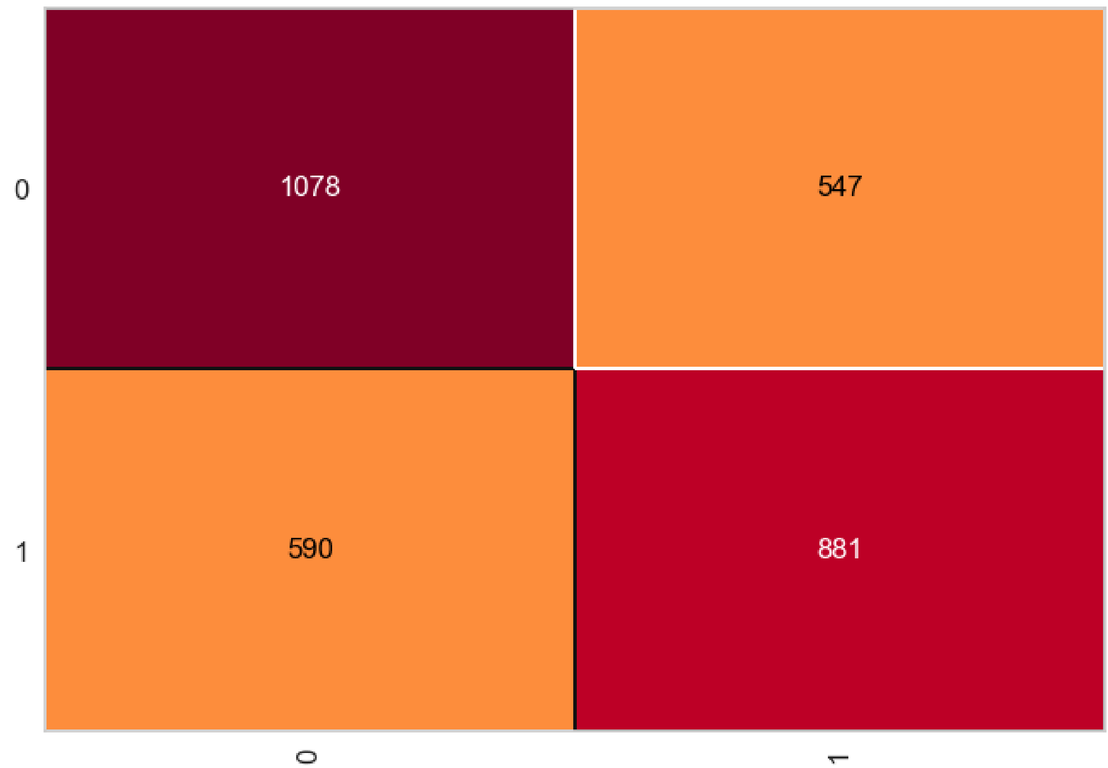
[31]:
```
#create an instance of the classifier and fit the training data
clf_rf = RandomForestClassifier(n_estimators=50,max_depth =50 ,␣
 ↪min_samples_split=5,random_state=0)
```

[32]:
```
y_pred_test = model_fit_predict(clf_rf, X_train, y_train, X_test)

#f1 score for training data
f1 = round(f1_score(y_test, y_pred_test),2)

#accuracy score for training data
acc = round(accuracy_score(y_test, y_pred_test),2)

#precision score for training data
pre = round(precision_score(y_test, y_pred_test),2)

#Measures how well the model identifies all the true positives (completed␣
 ↪bookings).
recall = round(recall_score(y_test, y_pred_test),2)

#Measures how well the model identifies the true negatives (non-completed␣
 ↪bookings).
specificity = round(recall_score(y_test, y_pred_test, pos_label=0),2)

print(f"Accuracy, precision, recall and f1-score for training data are {acc},␣
 ↪{pre}, {recall}, {specificity} and {f1} respectively")
```

```
Accuracy, precision, recall and f1-score for training data are 0.63, 0.62, 0.6,
0.66 and 0.61 respectively
```

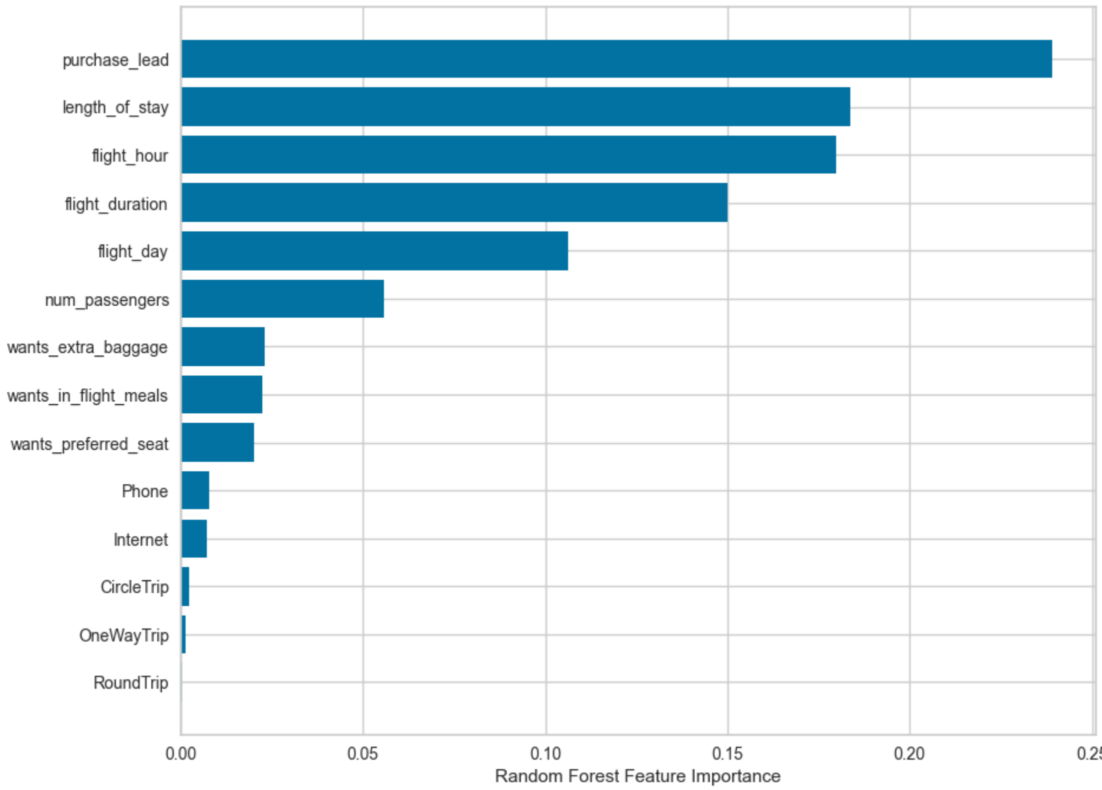
[33]:
```
cm = ConfusionMatrix(clf_rf, classes=[0,1])
cm.fit(X_train, y_train)

cm.score(X_test, y_test)
```

[33]: 0.6327519379844961



[34]:
```
plt.figure(figsize=(10,8))
sorted_idx = clf_rf.feature_importances_.argsort()
plt.barh(scaled_df.iloc[:,:-1].columns[sorted_idx], clf_rf.
  ↪feature_importances_[sorted_idx])
plt.xlabel("Random Forest Feature Importance")
```

[34]: Text(0.5, 0, 'Random Forest Feature Importance')



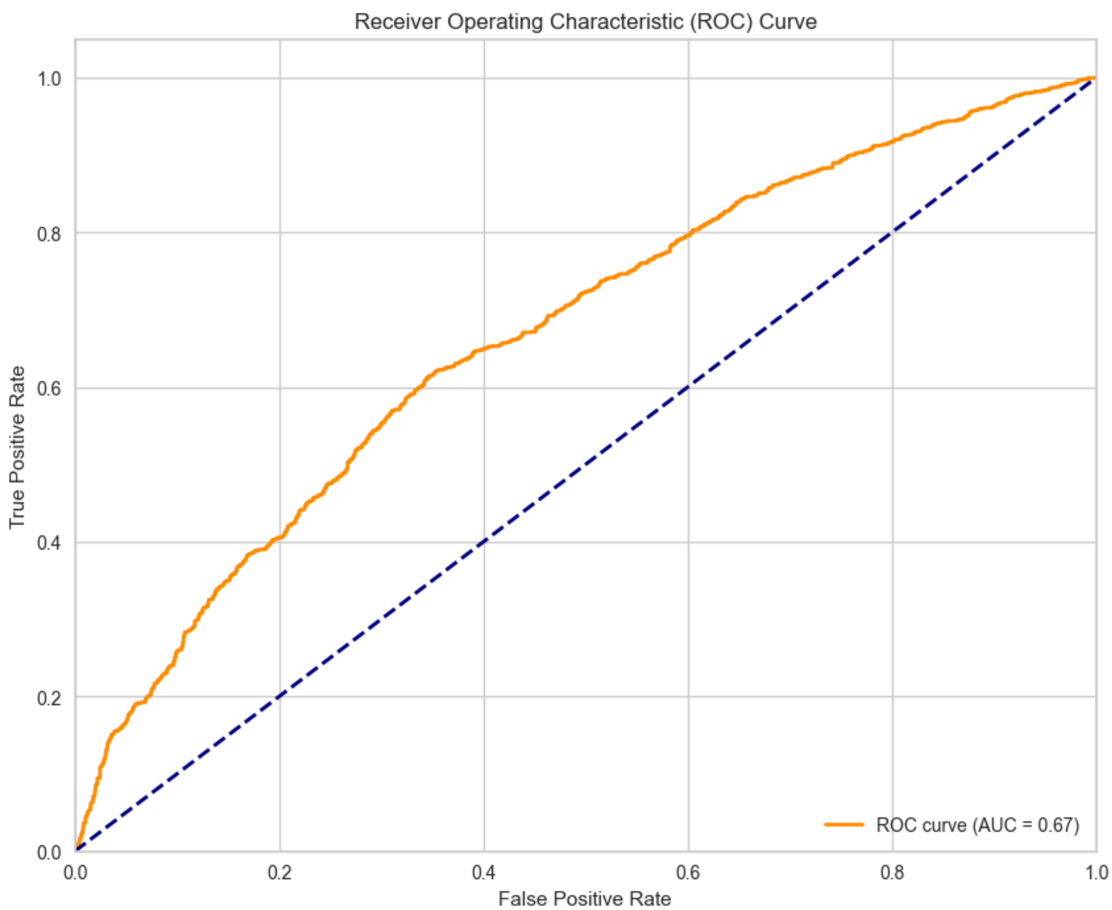
[35]:
```
from sklearn.metrics import roc_curve, auc
```

[36]:
```
y_pred_proba = clf_rf.predict_proba(X_test)[:, 1]
```

[37]:
```
fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba)
```

[38]:
```
roc_auc = auc(fpr, tpr)
```

[39]:
```
plt.figure(figsize=(10, 8))
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (AUC = {roc_auc:.
    ↪2f})')
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc="lower right")
plt.show()
```


Receiver Operating Characteristic (ROC) Curve
1.0
0.8
0.6
0.4
0.2
0.0
True Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate
ROC curve (AUC = 0.67)

[ ]: