**Predict Revenue of online\_shoppers using random forest algorithm**[**¶**](#gjdgxs)

This file consists of various Information related to customer behavior in online shopping websites.

In [1]:

**import** **numpy** **as** **np**   
**import** **pandas** **as** **pd**

In [2]:

df=pd.read\_csv("online\_shoppers\_intention.csv")  
df.head()

Out[2]:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Administrative** | **Administrative\_Duration** | **Informational** | **Informational\_Duration** | **ProductRelated** | **ProductRelated\_Duration** | **BounceRates** | **ExitRates** | **PageValues** | **SpecialDay** | **Month** | **OperatingSystems** | **Browser** | **Region** | **TrafficType** | **VisitorType** | **Weekend** | **Revenue** |
| **0** | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.000000 | 0.20 | 0.20 | 0.0 | 0.0 | Feb | 1 | 1 | 1 | 1 | Returning\_Visitor | False | False |
| **1** | 0.0 | 0.0 | 0.0 | 0.0 | 2.0 | 64.000000 | 0.00 | 0.10 | 0.0 | 0.0 | Feb | 2 | 2 | 1 | 2 | Returning\_Visitor | False | False |
| **2** | 0.0 | -1.0 | 0.0 | -1.0 | 1.0 | -1.000000 | 0.20 | 0.20 | 0.0 | 0.0 | Feb | 4 | 1 | 9 | 3 | Returning\_Visitor | False | False |
| **3** | 0.0 | 0.0 | 0.0 | 0.0 | 2.0 | 2.666667 | 0.05 | 0.14 | 0.0 | 0.0 | Feb | 3 | 2 | 2 | 4 | Returning\_Visitor | False | False |
| **4** | 0.0 | 0.0 | 0.0 | 0.0 | 10.0 | 627.500000 | 0.02 | 0.05 | 0.0 | 0.0 | Feb | 3 | 3 | 1 | 4 | Returning\_Visitor | True | False |

In [3]:

df.isna().sum()

Out[3]:

Administrative 14  
Administrative\_Duration 14  
Informational 14  
Informational\_Duration 14  
ProductRelated 14  
ProductRelated\_Duration 14  
BounceRates 14  
ExitRates 14  
PageValues 0  
SpecialDay 0  
Month 0  
OperatingSystems 0  
Browser 0  
Region 0  
TrafficType 0  
VisitorType 0  
Weekend 0  
Revenue 0  
dtype: int64

In [4]:

df['Administrative']=df['Administrative'].fillna(df['Administrative'].median())  
df['Administrative\_Duration']=df['Administrative\_Duration'].fillna(df['Administrative\_Duration'].median())  
df['Informational']=df['Informational'].fillna(df['Informational'].median())  
df['Informational\_Duration']=df['Informational\_Duration'].fillna(df['Informational\_Duration'].median())  
df['ProductRelated']=df['ProductRelated'].fillna(df['ProductRelated'].median())  
df['ProductRelated\_Duration']=df['ProductRelated\_Duration'].fillna(df['ProductRelated\_Duration'].median())  
df['BounceRates']=df['BounceRates'].fillna(df['BounceRates'].median())  
df['ExitRates']=df['ExitRates'].fillna(df['ExitRates'].median())

In [5]:

df.dtypes

Out[5]:

Administrative float64  
Administrative\_Duration float64  
Informational float64  
Informational\_Duration float64  
ProductRelated float64  
ProductRelated\_Duration float64  
BounceRates float64  
ExitRates float64  
PageValues float64  
SpecialDay float64  
Month object  
OperatingSystems int64  
Browser int64  
Region int64  
TrafficType int64  
VisitorType object  
Weekend bool  
Revenue bool  
dtype: object

In [6]:

**from** **sklearn.preprocessing** **import** LabelEncoder  
cat\_to\_num = df[['Month','VisitorType','Weekend','Revenue']]  
le = LabelEncoder()   
**for** i **in** cat\_to\_num:  
 df[i] = le.fit\_transform(cat\_to\_num[i])

In [7]:

df.head(3)

Out[7]:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Administrative** | **Administrative\_Duration** | **Informational** | **Informational\_Duration** | **ProductRelated** | **ProductRelated\_Duration** | **BounceRates** | **ExitRates** | **PageValues** | **SpecialDay** | **Month** | **OperatingSystems** | **Browser** | **Region** | **TrafficType** | **VisitorType** | **Weekend** | **Revenue** |
| **0** | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.2 | 0.2 | 0.0 | 0.0 | 2 | 1 | 1 | 1 | 1 | 2 | 0 | 0 |
| **1** | 0.0 | 0.0 | 0.0 | 0.0 | 2.0 | 64.0 | 0.0 | 0.1 | 0.0 | 0.0 | 2 | 2 | 2 | 1 | 2 | 2 | 0 | 0 |
| **2** | 0.0 | -1.0 | 0.0 | -1.0 | 1.0 | -1.0 | 0.2 | 0.2 | 0.0 | 0.0 | 2 | 4 | 1 | 9 | 3 | 2 | 0 | 0 |

In [8]:

df.describe()

Out[8]:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Administrative** | **Administrative\_Duration** | **Informational** | **Informational\_Duration** | **ProductRelated** | **ProductRelated\_Duration** | **BounceRates** | **ExitRates** | **PageValues** | **SpecialDay** | **Month** | **OperatingSystems** | **Browser** | **Region** | **TrafficType** | **VisitorType** | **Weekend** | **Revenue** |
| **count** | 12330.000000 | 12330.000000 | 12330.000000 | 12330.000000 | 12330.000000 | 12330.000000 | 12330.000000 | 12330.000000 | 12330.000000 | 12330.000000 | 12330.000000 | 12330.000000 | 12330.000000 | 12330.000000 | 12330.000000 | 12330.000000 | 12330.000000 | 12330.000000 |
| **mean** | 2.316302 | 80.823396 | 0.503406 | 34.467207 | 31.748256 | 1195.360026 | 0.022131 | 0.042982 | 5.889258 | 0.061427 | 5.163990 | 2.124006 | 2.357097 | 3.147364 | 4.069586 | 1.718329 | 0.232603 | 0.154745 |
| **std** | 3.321163 | 176.777041 | 1.270093 | 140.750298 | 44.467488 | 1913.390668 | 0.048404 | 0.048503 | 18.568437 | 0.198917 | 2.370199 | 0.911325 | 1.717277 | 2.401591 | 4.025169 | 0.690759 | 0.422509 | 0.361676 |
| **min** | 0.000000 | -1.000000 | 0.000000 | -1.000000 | 0.000000 | -1.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 0.000000 | 0.000000 | 0.000000 |
| **25%** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 7.000000 | 185.287500 | 0.000000 | 0.014286 | 0.000000 | 0.000000 | 5.000000 | 2.000000 | 2.000000 | 1.000000 | 2.000000 | 2.000000 | 0.000000 | 0.000000 |
| **50%** | 1.000000 | 8.000000 | 0.000000 | 0.000000 | 18.000000 | 599.766190 | 0.003119 | 0.025124 | 0.000000 | 0.000000 | 6.000000 | 2.000000 | 2.000000 | 3.000000 | 2.000000 | 2.000000 | 0.000000 | 0.000000 |
| **75%** | 4.000000 | 93.256250 | 0.000000 | 0.000000 | 38.000000 | 1464.157213 | 0.016667 | 0.050000 | 0.000000 | 0.000000 | 7.000000 | 3.000000 | 2.000000 | 4.000000 | 4.000000 | 2.000000 | 0.000000 | 0.000000 |
| **max** | 27.000000 | 3398.750000 | 24.000000 | 2549.375000 | 705.000000 | 63973.522230 | 0.200000 | 0.200000 | 361.763742 | 1.000000 | 9.000000 | 8.000000 | 13.000000 | 9.000000 | 20.000000 | 2.000000 | 1.000000 | 1.000000 |

In [9]:

X=df.drop('Revenue',axis=1)   
y=df['Revenue']

In [10]:

X.tail()

Out[10]:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Administrative** | **Administrative\_Duration** | **Informational** | **Informational\_Duration** | **ProductRelated** | **ProductRelated\_Duration** | **BounceRates** | **ExitRates** | **PageValues** | **SpecialDay** | **Month** | **OperatingSystems** | **Browser** | **Region** | **TrafficType** | **VisitorType** | **Weekend** |
| **12325** | 3.0 | 145.0 | 0.0 | 0.0 | 53.0 | 1783.791667 | 0.007143 | 0.029031 | 12.241717 | 0.0 | 1 | 4 | 6 | 1 | 1 | 2 | 1 |
| **12326** | 0.0 | 0.0 | 0.0 | 0.0 | 5.0 | 465.750000 | 0.000000 | 0.021333 | 0.000000 | 0.0 | 7 | 3 | 2 | 1 | 8 | 2 | 1 |
| **12327** | 0.0 | 0.0 | 0.0 | 0.0 | 6.0 | 184.250000 | 0.083333 | 0.086667 | 0.000000 | 0.0 | 7 | 3 | 2 | 1 | 13 | 2 | 1 |
| **12328** | 4.0 | 75.0 | 0.0 | 0.0 | 15.0 | 346.000000 | 0.000000 | 0.021053 | 0.000000 | 0.0 | 7 | 2 | 2 | 3 | 11 | 2 | 0 |
| **12329** | 0.0 | 0.0 | 0.0 | 0.0 | 3.0 | 21.250000 | 0.000000 | 0.066667 | 0.000000 | 0.0 | 7 | 3 | 2 | 1 | 2 | 0 | 1 |

In [11]:

**from** **sklearn.model\_selection** **import** train\_test\_split  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

In [12]:

**from** **sklearn.preprocessing** **import** StandardScaler  
sc = StandardScaler()  
  
X\_train = sc.fit\_transform(X\_train)  
X\_test = sc.transform(X\_test)

D:\anaconda3\lib\site-packages\sklearn\preprocessing\data.py:645: DataConversionWarning: Data with input dtype int32, int64, float64 were all converted to float64 by StandardScaler.  
 return self.partial\_fit(X, y)  
D:\anaconda3\lib\site-packages\sklearn\base.py:464: DataConversionWarning: Data with input dtype int32, int64, float64 were all converted to float64 by StandardScaler.  
 return self.fit(X, \*\*fit\_params).transform(X)  
D:\anaconda3\lib\site-packages\ipykernel\_launcher.py:5: DataConversionWarning: Data with input dtype int32, int64, float64 were all converted to float64 by StandardScaler.  
 """

In [13]:

**from** **sklearn.ensemble** **import** RandomForestClassifier  
my\_model = RandomForestClassifier(n\_estimators = 50, criterion = 'entropy', random\_state = 42)  
result=my\_model.fit(X\_train, y\_train)

In [14]:

predictions = result.predict(X\_test)

In [15]:

predictions

Out[15]:

array([0, 0, 1, ..., 0, 0, 0], dtype=int64)

In [16]:

**from** **sklearn** **import** metrics

In [17]:

print("Accuracy:",metrics.accuracy\_score(y\_test, predictions))

Accuracy: 0.889294403892944

In [24]:

**import** **seaborn** **as** **sn**  
**from** **sklearn.metrics** **import** confusion\_matrix  
conf\_matrix =confusion\_matrix(predictions,y\_test)  
confusion\_df = pd.DataFrame(conf\_matrix, index=['Actual 0','Actual 1'], columns=['Predicted 0','Predicted 1'])  
confusion\_df  
*#matrix=sn.heatmap(confusion\_df, cmap='coolwarm', annot=True)*

Out[24]:

|  |  |  |
| --- | --- | --- |
|  | **Predicted 0** | **Predicted 1** |
| **Actual 0** | 1979 | 197 |
| **Actual 1** | 76 | 214 |

In [20]:

pred\_new = result.predict([[0.0,0.0,0.0,0.0,0.1,0.0,0.5,0.2,0.0,0.0,2,1,1,1,1,2,0]])  
pred\_new

Out[20]:

array([0], dtype=int64)

In [21]:

pred\_new = result.predict([[4.0,75.0,0.0,0.0,15.0,346.000000,0.000000,0.021053,0.000000,0.0,7,2,2,3,11,2,0]])  
pred\_new

Out[21]:

array([0], dtype=int64)

In [ ]: