In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:

```
import io
%cd "G:\PGW23\python\online shopper"
```

```
G:\PGW23\python\online shopper
```

In [3]:

```
intent=pd.read_csv("online_shoppers_intention.csv")
```

In [4]:

```
intent.head(30)
```

Out[4]:

	Administrative	Administrative_Duration	Informational	Informational_Duration	ProductRelated	ProductRelated_Duration	BounceRates	ExitRates	Page\
0	0	0.0	0	0.0	1	0.000000	0.200000	0.200000	0.0
1	0	0.0	0	0.0	2	64.000000	0.000000	0.100000	0.0
2	0	0.0	0	0.0	1	0.000000	0.200000	0.200000	0.0
3	0	0.0	0	0.0	2	2.666667	0.050000	0.140000	0.0
4	0	0.0	0	0.0	10	627.500000	0.020000	0.050000	0.0
5	0	0.0	0	0.0	19	154.216667	0.015789	0.024561	0.0
6	0	0.0	0	0.0	1	0.000000	0.200000	0.200000	0.0
7	1	0.0	0	0.0	0	0.000000	0.200000	0.200000	0.0
8	0	0.0	0	0.0	2	37.000000	0.000000	0.100000	0.0
9	0	0.0	0	0.0	3	738.000000	0.000000	0.022222	0.0
10	0	0.0	0	0.0	3	395.000000	0.000000	0.066667	0.0
11	0	0.0	0	0.0	16	407.750000	0.018750	0.025833	0.0
12	0	0.0	0	0.0	7	280.500000	0.000000	0.028571	0.0
13	0	0.0	0	0.0	6	98.000000	0.000000	0.066667	0.0
14	0	0.0	0	0.0	2	68.000000	0.000000	0.100000	0.0
15	2	53.0	0	0.0	23	1668.285119	0.008333	0.016313	0.0
16	0	0.0	0	0.0	1	0.000000	0.200000	0.200000	0.0
17	0	0.0	0	0.0	13	334.966667	0.000000	0.007692	0.0
18	0	0.0	0	0.0	2	32.000000	0.000000	0.100000	0.0
19	0	0.0	0	0.0	20	2981.166667	0.000000	0.010000	0.0
20	0	0.0	0	0.0	8	136.166667	0.000000	0.008333	0.0
21	0	0.0	0	0.0	2	0.000000	0.200000	0.200000	0.0
22	0	0.0	0	0.0	3	105.000000	0.000000	0.033333	0.0
23	0	0.0	0	0.0	2	15.000000	0.000000	0.100000	0.0
24	0	0.0	0	0.0	1	0.000000	0.200000	0.200000	0.0
25	0	0.0	0	0.0	5	156.000000	0.000000	0.040000	0.0
26	4	64.6	0	0.0	32	1135.444444	0.002857	0.009524	0.0
27	0	0.0	0	0.0	4	76.000000	0.050000	0.100000	0.0
28	0	0.0	0	0.0	4	63.000000	0.000000	0.050000	0.0
29	1	6.0	1	0.0	45	1582.750000	0.043478	0.050821	54.1

In [5]:

```
intent.shape
```

Out[5]:

```
(12330, 18)
```

In [6]:

```
intent.dtypes
```

Out[6]:

```
Administrative               int64
Administrative_Duration    float64
Informational                int64
Informational_Duration     float64
ProductRelated               int64
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 [7]:

```
intent.all()
```

Out[7]:

```
Administrative             False
Administrative_Duration    False
Informational              False
Informational_Duration     False
ProductRelated             False
ProductRelated_Duration    False
BounceRates                False
ExitRates                  False
PageValues                 False
SpecialDay                 False
Month                       True
OperatingSystems            True
Browser                     True
Region                      True
TrafficType                 True
VisitorType                 True
Weekend                    False
Revenue                    False
dtype: bool
```

In [8]:

```
intent.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12330 entries, 0 to 12329
Data columns (total 18 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   Administrative           12330 non-null  int64  
 1   Administrative_Duration  12330 non-null  float64
 2   Informational            12330 non-null  int64  
 3   Informational_Duration   12330 non-null  float64
 4   ProductRelated           12330 non-null  int64  
 5   ProductRelated_Duration  12330 non-null  float64
 6   BounceRates              12330 non-null  float64
 7   ExitRates                12330 non-null  float64
 8   PageValues               12330 non-null  float64
 9   SpecialDay               12330 non-null  float64
 10  Month                    12330 non-null  object 
 11  OperatingSystems         12330 non-null  int64  
 12  Browser                  12330 non-null  int64  
 13  Region                   12330 non-null  int64  
 14  TrafficType              12330 non-null  int64  
 15  VisitorType              12330 non-null  object 
 16  Weekend                  12330 non-null  bool   
 17  Revenue                  12330 non-null  bool   
dtypes: bool(2), float64(7), int64(7), object(2)
memory usage: 1.5+ MB
```

In [9]:

```
intent.columns
```

Out[9]:

```
Index(['Administrative', 'Administrative_Duration', 'Informational',
       'Informational_Duration', 'ProductRelated', 'ProductRelated_Duration',
       'BounceRates', 'ExitRates', 'PageValues', 'SpecialDay', 'Month',
       'OperatingSystems', 'Browser', 'Region', 'TrafficType', 'VisitorType',
       'Weekend', 'Revenue'],
      dtype='object')
```

In [10]:

```
intent.Administrative.value_counts()
```

Out[10]:

```
0     5768
1     1354
2     1114
3      915
4      765
5      575
6      432
7      338
8      287
9      225
10     153
11     105
12      86
13      56
14      44
15      38
16      24
17      16
18      12
19       6
24       4
22       4
23       3
21       2
20       2
27       1
26       1
Name: Administrative, dtype: int64
```

In [11]:

```
intent.Informational.value_counts()
```

Out[11]:

```
0     9699
1     1041
2      728
3      380
4      222
5       99
6       78
7       36
9       15
8       14
10       7
12       5
14       2
16       1
11       1
24       1
13       1
Name: Informational, dtype: int64
```

In [12]:

```
intent.ProductRelated.value_counts()
```

Out[12]:

```
1      622
2      465
3      458
4      404
6      396
      ...
243      1
409      1
262      1
414      1
192      1
Name: ProductRelated, Length: 311, dtype: int64
```

In [13]:

```
intent.OperatingSystems.value_counts()
```

Out[13]:

```
2    6601
1    2585
3    2555
4     478
8      79
6      19
7       7
5       6
Name: OperatingSystems, dtype: int64
```

In [14]:

```
intent.Browser.value_counts()
```

Out[14]:

```
2     7961
1     2462
4      736
5      467
6      174
10     163
8      135
3      105
13      61
7       49
12      10
11       6
9        1
Name: Browser, dtype: int64
```

In [15]:

```
intent.Region.value_counts()
```

Out[15]:

```
1    4780
3    2403
4    1182
2    1136
6     805
7     761
9     511
8     434
5     318
Name: Region, dtype: int64
```

In [16]:

```
intent.TrafficType.value_counts()
```

Out[16]:

```
2     3913
1     2451
3     2052
4     1069
13     738
10     450
6      444
8      343
5      260
11     247
20     198
9       42
7       40
15      38
19      17
14      13
18      10
16       3
12       1
17       1
Name: TrafficType, dtype: int64
```

In [17]:

```
intent.Informational.value_counts()
```

Out[17]:

```
0     9699
1     1041
2      728
3      380
4      222
5       99
6       78
7       36
9       15
8       14
10       7
12       5
14       2
16       1
11       1
24       1
13       1
Name: Informational, dtype: int64
```

In [18]:

```
intent.BounceRates.groupby(intent.Revenue).mean()
```

Out[18]:

```
Revenue
False    0.025317
True     0.005117
Name: BounceRates, dtype: float64
```

In [19]:

```
intent.PageValues.groupby(intent.Revenue).mean()
```

Out[19]:

```
Revenue
False     1.975998
True     27.264518
Name: PageValues, dtype: float64
```

In [20]:

```
intent.PageValues.describe
```

Out[20]:

```
<bound method NDFrame.describe of 0         0.000000
1         0.000000
2         0.000000
3         0.000000
4         0.000000
            ...
12325    12.241717
12326     0.000000
12327     0.000000
12328     0.000000
12329     0.000000
Name: PageValues, Length: 12330, dtype: float64>
```

In [21]:

```
# test null the avg bounce rate of revenue False/true are eqaul
intent.BounceRates.groupby(intent.Revenue).var()
```

Out[21]:

```
Revenue
False    0.002691
True     0.000148
Name: BounceRates, dtype: float64
```

In [22]:

```
# null -no signification differences in avg bouncerates of the revenue True/False

#  alt - signification differences in avg bouncerates of the revenue True/False

revT=intent[intent.Revenue==True]
revF=intent[intent.Revenue==False]
```

In [23]:

```
revF.head()
```

Out[23]:

	Administrative	Administrative_Duration	Informational	Informational_Duration	ProductRelated	ProductRelated_Duration	BounceRates	ExitRates	PageVa
0	0	0.0	0	0.0	1	0.000000	0.20	0.20	
1	0	0.0	0	0.0	2	64.000000	0.00	0.10	
2	0	0.0	0	0.0	1	0.000000	0.20	0.20	
3	0	0.0	0	0.0	2	2.666667	0.05	0.14	
4	0	0.0	0	0.0	10	627.500000	0.02	0.05	

In [24]:

```
revT.head()
```

Out[24]:

	Administrative	Administrative_Duration	Informational	Informational_Duration	ProductRelated	ProductRelated_Duration	BounceRates	ExitRates	Page
65	3	87.833333	0	0.0	27	798.333333	0.000000	0.012644	22.
76	10	1005.666667	0	0.0	36	2111.341667	0.004348	0.014493	11.
101	4	61.000000	0	0.0	19	607.000000	0.000000	0.026984	17.
188	9	111.500000	1	48.5	49	1868.819697	0.000000	0.020709	1.
196	2	56.000000	1	144.0	67	2563.783333	0.000000	0.005797	19.

In [25]:

```
from scipy.stats import ttest_ind
```

In [26]:

```
ttest_ind(revT.BounceRates,revF.BounceRates,equal_var=False)
# REJECT NULL
```

Out[26]:

```
Ttest_indResult(statistic=-34.84635983271681, pvalue=2.587228296767619e-253)
```

In [27]:

```
# TEST NULL AVERAGE EXITRATES FOR REVENUE TRUE/FLASE EQAUL TO 1

ttest_ind(revT.ExitRates,revF.ExitRates,equal_var=False)
```

Out[27]:

```
Ttest_indResult(statistic=-44.33213022344043, pvalue=0.0)
```

In [28]:

```
# test Null avg pagevalue of different visitortype eaqual?

intent.PageValues.groupby(intent.VisitorType).mean()
```

Out[28]:

```
VisitorType
New_Visitor          10.772187
Other                18.191812
Returning_Visitor     5.006176
Name: PageValues, dtype: float64
```

In [29]:

```
# null -no signification differences in avg page value of the vistor type

# Alt - signification differences in avg page value of the vistor type


new=intent[intent.VisitorType=="New_Visitor"]
other=intent[intent.VisitorType=="Other"]
returning=intent[intent.VisitorType=="Returning_Visitor"]
```

In [30]:

```
new.head()
```

Out[30]:

	Administrative	Administrative_Duration	Informational	Informational_Duration	ProductRelated	ProductRelated_Duration	BounceRates	ExitRates	Page
93	0	0.0	0	0.0	13	649.250000	0.0	0.015385	0.
196	2	56.0	1	144.0	67	2563.783333	0.0	0.005797	19.
198	0	0.0	0	0.0	17	840.233333	0.0	0.001667	109.
199	3	94.0	2	125.0	55	1970.844805	0.0	0.001724	96.
202	5	218.0	0	0.0	13	284.500000	0.0	0.004167	0.

In [31]:

```
other.head()
```

Out[31]:

	Administrative	Administrative_Duration	Informational	Informational_Duration	ProductRelated	ProductRelated_Duration	BounceRates	ExitRates	Pag
5679	0	0.00	4	225.766667	222	9630.209524	0.053355	0.066159	
8006	5	446.25	0	0.000000	18	815.250000	0.000000	0.002500	
8105	0	0.00	0	0.000000	8	493.750000	0.000000	0.050000	
8115	0	0.00	0	0.000000	7	87.000000	0.000000	0.028571	
8187	0	0.00	0	0.000000	4	129.500000	0.000000	0.050000	

In [32]:

```
returning.head()
```

Out[32]:

	Administrative	Administrative_Duration	Informational	Informational_Duration	ProductRelated	ProductRelated_Duration	BounceRates	ExitRates	PageVa
0	0	0.0	0	0.0	1	0.000000	0.20	0.20	
1	0	0.0	0	0.0	2	64.000000	0.00	0.10	
2	0	0.0	0	0.0	1	0.000000	0.20	0.20	
3	0	0.0	0	0.0	2	2.666667	0.05	0.14	
4	0	0.0	0	0.0	10	627.500000	0.02	0.05	

In [33]:

```
from scipy.stats import f_oneway
```

In [34]:

```
f_oneway(new.PageValues,other.PageValues,returning.PageValues)
```

Out[34]:

```
F_onewayResult(statistic=90.45482263934825, pvalue=1.0033303968830675e-39)
```

In [35]:

```
f_oneway(new.ExitRates,other.ExitRates,returning.ExitRates)
```

Out[35]:

```
F_onewayResult(statistic=221.166709631569, pvalue=4.282460054622956e-95)
```

In [36]:

```
# test null no association b/w weekend and revenue

pd.crosstab(intent.Weekend,intent.Revenue)
```

Out[36]:

Revenue	**False**	**True**
Weekend		
False	8053	1409
True	2369	499

In [37]:

```
# null-no association b/w both variable
# alt-association b/w both variable

from scipy.stats import chi2_contingency
```

In [38]:

```
chi2_contingency(pd.crosstab(intent.Weekend,intent.Revenue))
```

Out[38]:

```
(10.390978319534856,
 0.0012663251061221968,
 1,
 array([[7997.80729927, 1464.19270073],
        [2424.19270073,  443.80729927]]))
```

In [39]:

```
# test null no association b/w month and reveue

pd.crosstab(intent.Month,intent.Revenue)
```

Out[39]:

Revenue	False	True
Month		
Aug	357	76
Dec	1511	216
Feb	181	3
Jul	366	66
June	259	29
Mar	1715	192
May	2999	365
Nov	2238	760
Oct	434	115
Sep	362	86

In [40]:

```
chi2_contingency(pd.crosstab(intent.Month,intent.Revenue))
```

Out[40]:

```
(384.93476153599426,
 2.2387855164805443e-77,
 9,
 array([[ 365.99562044,   67.00437956],
        [1459.75620438,  267.24379562],
        [ 155.5270073 ,   28.4729927 ],
        [ 365.15036496,   66.84963504],
        [ 243.43357664,   44.56642336],
        [1611.90218978,  295.09781022],
        [2843.43941606,  520.56058394],
        [2534.07591241,  463.92408759],
        [ 464.04525547,   84.95474453],
        [ 378.67445255,   69.32554745]]))
```

In [41]:

```
intent.columns
```

Out[41]:

```
Index(['Administrative', 'Administrative_Duration', 'Informational',
       'Informational_Duration', 'ProductRelated', 'ProductRelated_Duration',
       'BounceRates', 'ExitRates', 'PageValues', 'SpecialDay', 'Month',
       'OperatingSystems', 'Browser', 'Region', 'TrafficType', 'VisitorType',
       'Weekend', 'Revenue'],
      dtype='object')
```

In [42]:

```
objectcol=intent[['Administrative','Informational','ProductRelated','SpecialDay','Month',
       'OperatingSystems','Browser','Region','TrafficType','VisitorType',
       'Weekend','Revenue']]
```

In [43]:

```
numericol=intent[['Administrative_Duration','Informational_Duration','ProductRelated_Duration',
       'BounceRates', 'ExitRates', 'PageValues']]
```

In [44]:

```
objectcol.shape
```

Out[44]:

```
(12330, 12)
```

In [45]:

```
numericol.shape
```

Out[45]:

```
(12330, 6)
```

In [46]:

```
intent.shape
```

Out[46]:

```
(12330, 18)
```

In [47]:

```
from sklearn.preprocessing import LabelEncoder
```

In [48]:

```
lb=LabelEncoder()
```

In [49]:

```
objectcoldummy=objectcol.apply(lb.fit_transform)
```

In [50]:

```
objectcoldummy.head()
```

Out[50]:

	Administrative	Informational	ProductRelated	SpecialDay	Month	OperatingSystems	Browser	Region	TrafficType	VisitorType	Weekend	Revenue
0	0	0	1	0	2	0	0	0	0	2	0	0
1	0	0	2	0	2	1	1	0	1	2	0	0
2	0	0	1	0	2	3	0	8	2	2	0	0
3	0	0	2	0	2	2	1	1	3	2	0	0
4	0	0	10	0	2	2	2	0	3	2	1	0

In [51]:

```
intent_df=pd.concat([numericol,objectcoldummy],axis=1)
```

In [52]:

```
intent_df.shape
```

Out[52]:

```
(12330, 18)
```

In [53]:

```
y=intent_df.Revenue
X=intent_df.drop('Revenue',axis=1)
```

In [54]:

```
from sklearn.linear_model import LogisticRegression
```

In [55]:

```
logfit=LogisticRegression(max_iter=10000)
```

In [56]:

```
logitmodel=logfit.fit(X,y)
```

In [57]:

```
logitmodel.score(X,y)
```

Out[57]:

```
0.8832927818329278
```

In [58]:

```
logitpredict=logitmodel.predict(X)
```

In [59]:

```
from sklearn.metrics import classification_report,plot_roc_curve
```

In [60]:

```
pd.crosstab(y,logitpredict)
```

Out[60]:

col_0	0	1
Revenue		
0	10171	251
1	1188	720

In [61]:

```
print(classification_report(y,logitpredict))
```

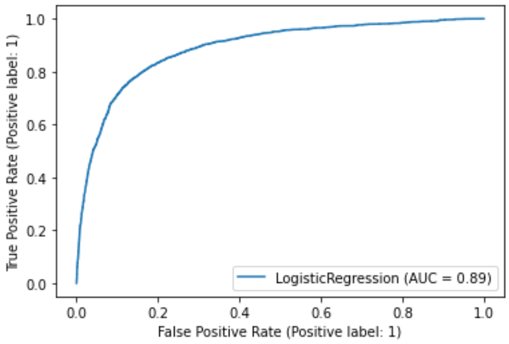
```
              precision    recall  f1-score   support

           0       0.90      0.98      0.93     10422
           1       0.74      0.38      0.50      1908

    accuracy                           0.88     12330
   macro avg       0.82      0.68      0.72     12330
weighted avg       0.87      0.88      0.87     12330
```

In [62]:

```
plot_roc_curve(logfit,X,y)
```

Out[62]:

```
<sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x179643ed8b0>
```



In [63]:

```
from sklearn.tree import DecisionTreeClassifier
```

In [64]:

```
tree=DecisionTreeClassifier(max_depth=12)
```

In [65]:

```
treemodel=tree.fit(X,y)
```

In [66]:

```
treemodel.score(X,y)
```

Out[66]:

```
0.9632603406326034
```

In [67]:

```
treepredicate=treemodel.predict(X)
```

In [68]:

```
pd.crosstab(y,treepredicate)
```

Out[68]:

col_0	0	1
Revenue		
0	10355	67
1	386	1522

In [69]:

```
print(classification_report(y,treepredicate))
```

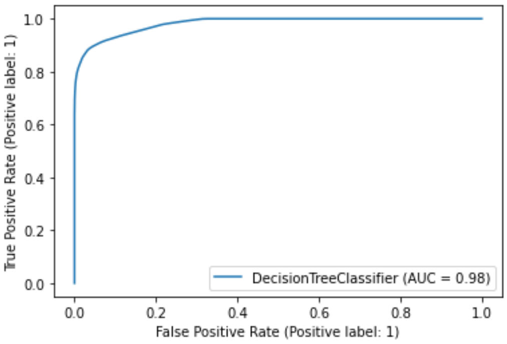
```
              precision    recall  f1-score   support

           0       0.96      0.99      0.98     10422
           1       0.96      0.80      0.87      1908

    accuracy                           0.96     12330
   macro avg       0.96      0.90      0.92     12330
weighted avg       0.96      0.96      0.96     12330
```

In [70]:

```
plot_roc_curve(tree,X,y)
```

Out[70]:

```
<sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x17966d01670>
```



In [71]:

```

from sklearn.naive_bayes import BernoulliNB
```

In [72]:

```
nbber=BernoulliNB()
```

In [73]:

```
nbbermodel=nbber.fit(X,y)
```

In [74]:

```
nbbermodel.score(X,y)
```

Out[74]:

```
0.8558799675587997
```

In [75]:

```
nbberpredict=nbbermodel.predict(X)
```

In [76]:

```
pd.crosstab(y,nbberpredict)
```

Out[76]:

col_0	0	1
Revenue		
0	9378	1044
1	733	1175

In [77]:

```
print(classification_report(y,nbberpredict))
```

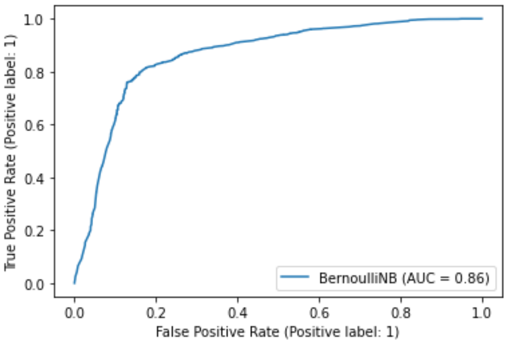
```
              precision    recall  f1-score   support

           0       0.93      0.90      0.91     10422
           1       0.53      0.62      0.57      1908

    accuracy                           0.86     12330
   macro avg       0.73      0.76      0.74     12330
weighted avg       0.87      0.86      0.86     12330
```

In [78]:

```
plot_roc_curve(nbber,X,y)
```

Out[78]:

```
<sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x17966d8d460>
```



In [79]:

```
from sklearn.ensemble import RandomForestClassifier
```

In [80]:

```
rf=RandomForestClassifier(n_estimators=5000,max_depth=20)
```

In [81]:

```
rfmodel=rf.fit(X,y)
```

In [82]:

```
rfmodel.score(X,y)
```

Out[82]:

```
0.999918896999189
```

In [83]:

```
rfpredicte=rfmodel.predict(X)
```

In [84]:

```
pd.crosstab(y,rfpredicte)
```

Out[84]:

col_0	0	1
Revenue		
0	10422	0
1	1	1907

In [85]:

```
print(classification_report(y,rfpredicte))
```

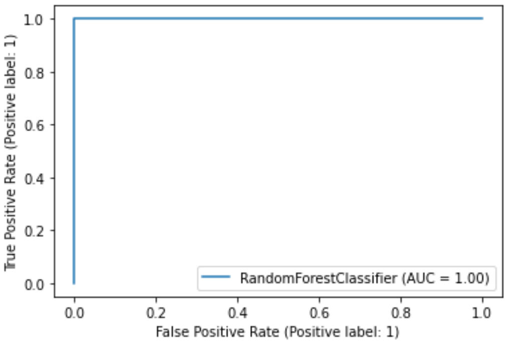
```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     10422
           1       1.00      1.00      1.00      1908

    accuracy                           1.00     12330
   macro avg       1.00      1.00      1.00     12330
weighted avg       1.00      1.00      1.00     12330
```

In [86]:

```
plot_roc_curve(rf,X,y)
```

Out[86]:

```
<sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x17966e688b0>
```



In [87]:

```
from sklearn.ensemble import GradientBoostingClassifier
```

In [88]:

```
gb=GradientBoostingClassifier(n_estimators=4000)
```

In [89]:

```
gbmodel=gb.fit(X,y)
```

In [90]:

```
gbmodel.score(X,y)
```

Out[90]:

```
0.9979724249797243
```

In [91]:

```
gbpredict=gbmodel.predict(X)
```

In [92]:

```
pd.crosstab(y,gbpredict)
```

Out[92]:

col_0	0	1
Revenue		
0	10422	0
1	25	1883

In [93]:

```
print(classification_report(y,gbpredict))
```

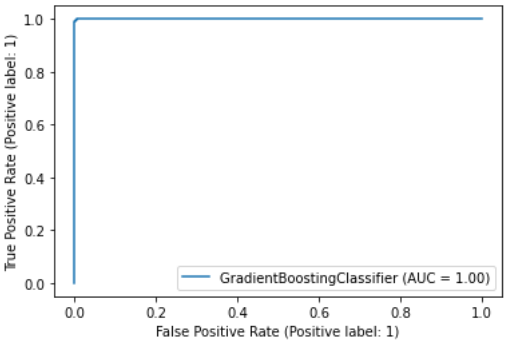
```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     10422
           1       1.00      0.99      0.99      1908

    accuracy                           1.00     12330
   macro avg       1.00      0.99      1.00     12330
weighted avg       1.00      1.00      1.00     12330
```

In [94]:

```
plot_roc_curve(gb,X,y)
```

Out[94]:

```
<sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x1797fe59c10>
```



In [95]:

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV
```

In [96]:

```
knn=KNeighborsClassifier()
```

In [99]:

```
k_range=list(range(1,20))
param_grid=dict(n_neighbors=k_range)
grid=GridSearchCV(knn,param_grid,cv=5)
```

In [100]:

```
grid_search=grid.fit(X,y)
```

In [101]:

```
grid_search.best_params_
```

Out[101]:

```
{'n_neighbors': 10}
```

In [102]:

```
grid_search.best_score_
```

Out[102]:

```
0.8659367396593673
```

In [ ]: