Task 1: Retrieving and Preparing the Data

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

df = pd.read_csv('file:///Users/amayiyer/Desktop/DatSci_Python/Ass2/Datasets/Online%20Shoppers%20Purchasing%20I
df.head(7)
```

Out[1]:

	Administrative	Administrative_Duration	Informational	Informational_Duration	ProductRelated	ProductRelated_Duration	BounceRates	ExitRates	PageValu
0	0	0.0	0	0.0	1	0.000000	0.200000	0.200000	
1	0	0.0	0	0.0	2	64.000000	0.000000	0.100000	
2	0	0.0	0	0.0	1	0.000000	0.200000	0.200000	
3	0	0.0	0	0.0	2	2.666667	0.050000	0.140000	
4	0	0.0	0	0.0	10	627.500000	0.020000	0.050000	
5	0	0.0	0	0.0	19	154.216667	0.015789	0.024561	
6	0	0.0	0	0.0	1	0.000000	0.200000	0.200000	

1.1: Checking for Missing Values (I have also conducted more data cleaning further in the notebook when the occasion arises)

In [2]:
```
# Checking the data types of all columns
print(df.dtypes)
print(df.isnull().sum())
```

```
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
Administrative             0
Administrative_Duration    0
Informational              0
Informational_Duration     0
ProductRelated             0
ProductRelated_Duration    0
BounceRates                0
ExitRates                  0
PageValues                 0
SpecialDay                 0
Month                      0
OperatingSystems           0
Browser                    0
Region                     0
TrafficType                0
VisitorType                0
Weekend                    0
Revenue                    0
dtype: int64
```

In [3]:
```
duplicate_rows = df.duplicated()
print("Number of duplicate rows: ", duplicate_rows.sum())
```

```
Number of duplicate rows:  125
```

Task 2.1 Description of the Statistics

In [4]:

```
#Descriptive and Summarizing Statistics will help me get a numerical understanding of the data
print(df.describe())

# This is for all of the categorical columns
for column in ['Month', 'OperatingSystems', 'Browser', 'Region', 'TrafficType', 'VisitorType', 'Weekend', 'Rever
    print(df[column].value_counts())
```

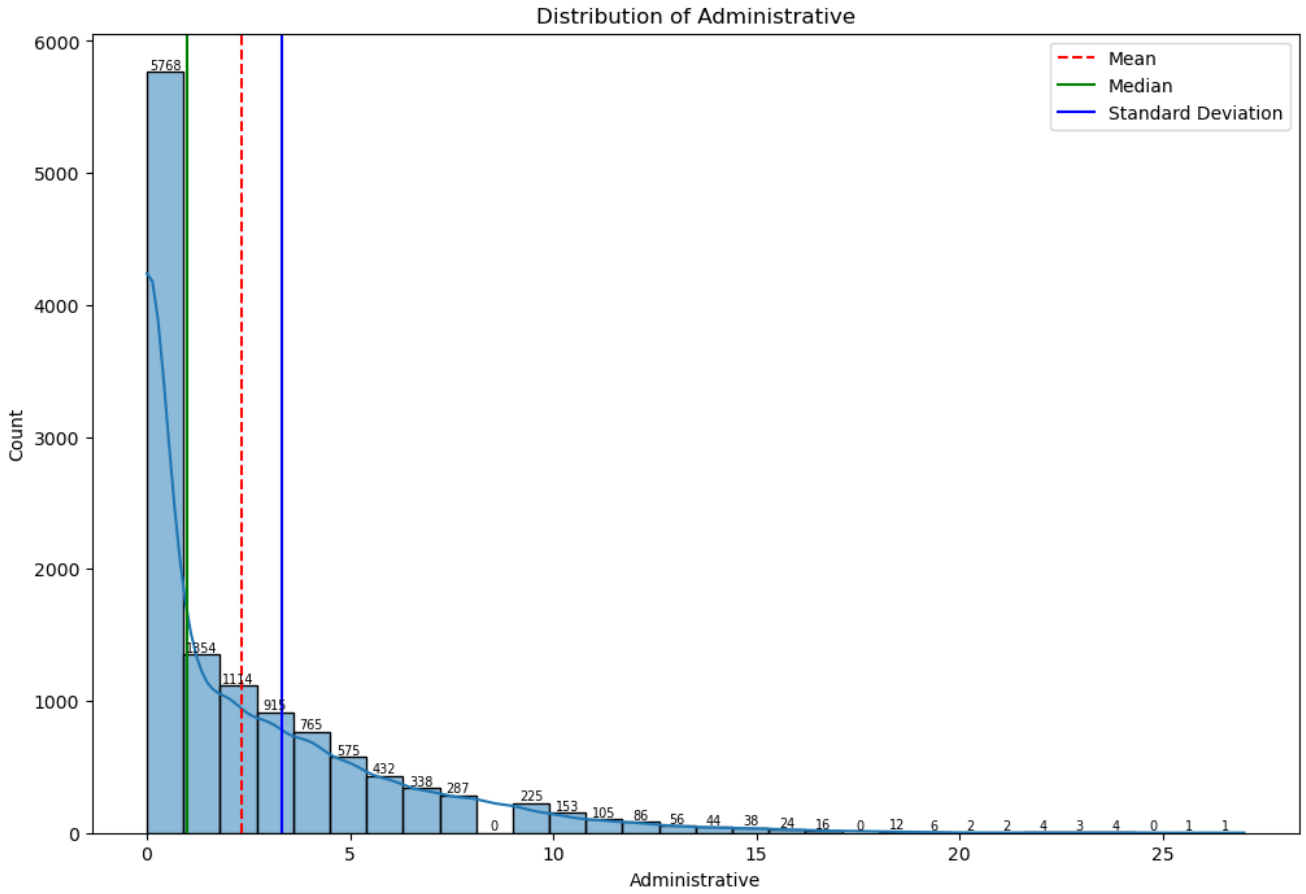
```
       Administrative  Administrative_Duration  Informational  \
count    12330.000000             12330.000000   12330.000000
mean         2.315166                80.818611       0.503569
std          3.321784               176.779107       1.270156
min          0.000000                 0.000000       0.000000
25%          0.000000                 0.000000       0.000000
50%          1.000000                 7.500000       0.000000
75%          4.000000                93.256250       0.000000
max         27.000000              3398.750000      24.000000

       Informational_Duration  ProductRelated  ProductRelated_Duration  \
count            12330.000000    12330.000000             12330.000000
mean                34.472398       31.731468              1194.746220
std                140.749294       44.475503              1913.669288
min                  0.000000        0.000000                 0.000000
25%                  0.000000        7.000000               184.137500
50%                  0.000000       18.000000               598.936905
75%                  0.000000       38.000000              1464.157214
max               2549.375000      705.000000             63973.522230

        BounceRates     ExitRates    PageValues    SpecialDay  \
count  12330.000000  12330.000000  12330.000000  12330.000000
mean       0.022191      0.043073      5.889258      0.061427
std        0.048488      0.048597     18.568437      0.198917
min        0.000000      0.000000      0.000000      0.000000
25%        0.000000      0.014286      0.000000      0.000000
50%        0.003112      0.025156      0.000000      0.000000
75%        0.016813      0.050000      0.000000      0.000000
max        0.200000      0.200000    361.763742      1.000000

       OperatingSystems       Browser        Region   TrafficType
count      12330.000000  12330.000000  12330.000000  12330.000000
mean           2.124006      2.357097      3.147364      4.069586
std            0.911325      1.717277      2.401591      4.025169
min            1.000000      1.000000      1.000000      1.000000
25%            2.000000      2.000000      1.000000      2.000000
50%            2.000000      2.000000      3.000000      2.000000
75%            3.000000      2.000000      4.000000      4.000000
max            8.000000     13.000000      9.000000     20.000000
May     3364
Nov     2998
Mar     1907
Dec     1727
Oct      549
Sep      448
Aug      433
Jul      432
June     288
Feb      184
Name: Month, dtype: int64
2    6601
1    2585
3    2555
4     478
8      79
6      19
7       7
5       6
Name: OperatingSystems, dtype: int64
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
1    4780
3    2403
4    1182
2    1136
6     805
7     761
9     511
8     434
5     318
Name: Region, dtype: int64
```

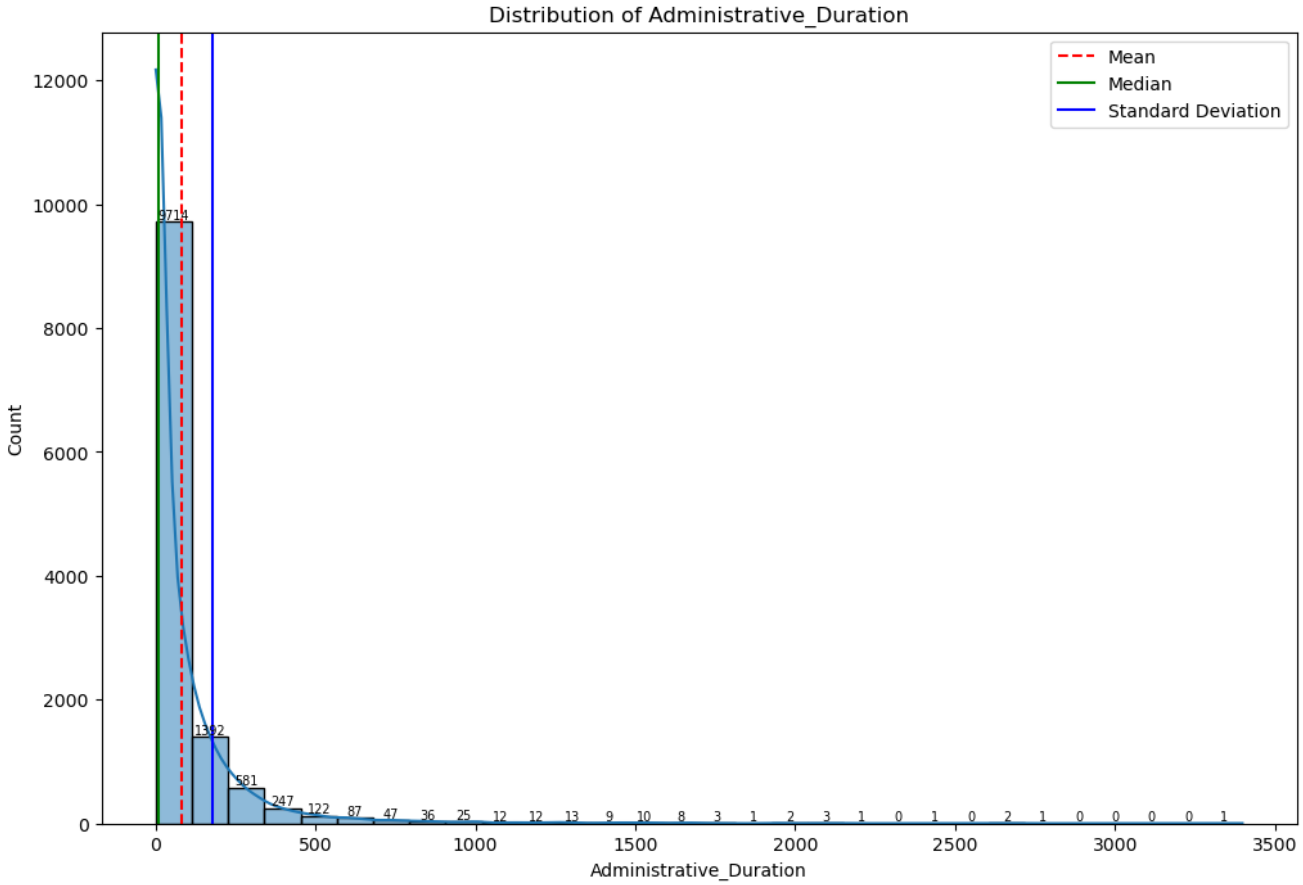
```
2     3913
1     2451
3     2052
4     1069
13     738
10     450
6      444
8      343
5      260
11     247
20     198
9       42
7       40
15      38
19      17
14      13
18      10
16       3
12       1
17       1
Name: TrafficType, dtype: int64
Returning_Visitor    10551
New_Visitor           1694
Other                   85
Name: VisitorType, dtype: int64
False    9462
True     2868
Name: Weekend, dtype: int64
False    10422
True      1908
Name: Revenue, dtype: int64
```

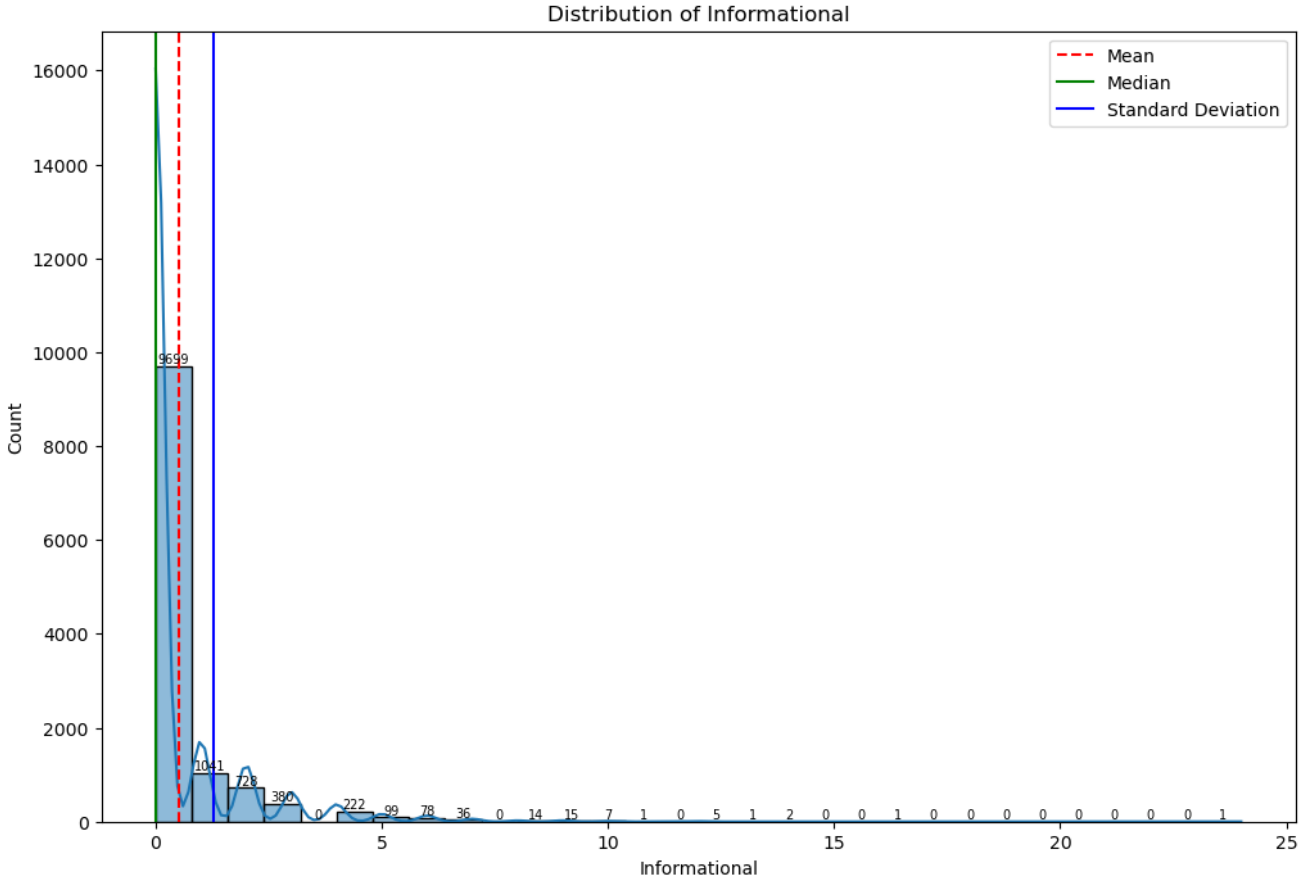
Task 1.1: Removing all values for which the SpecialDay values are not 0 or 1

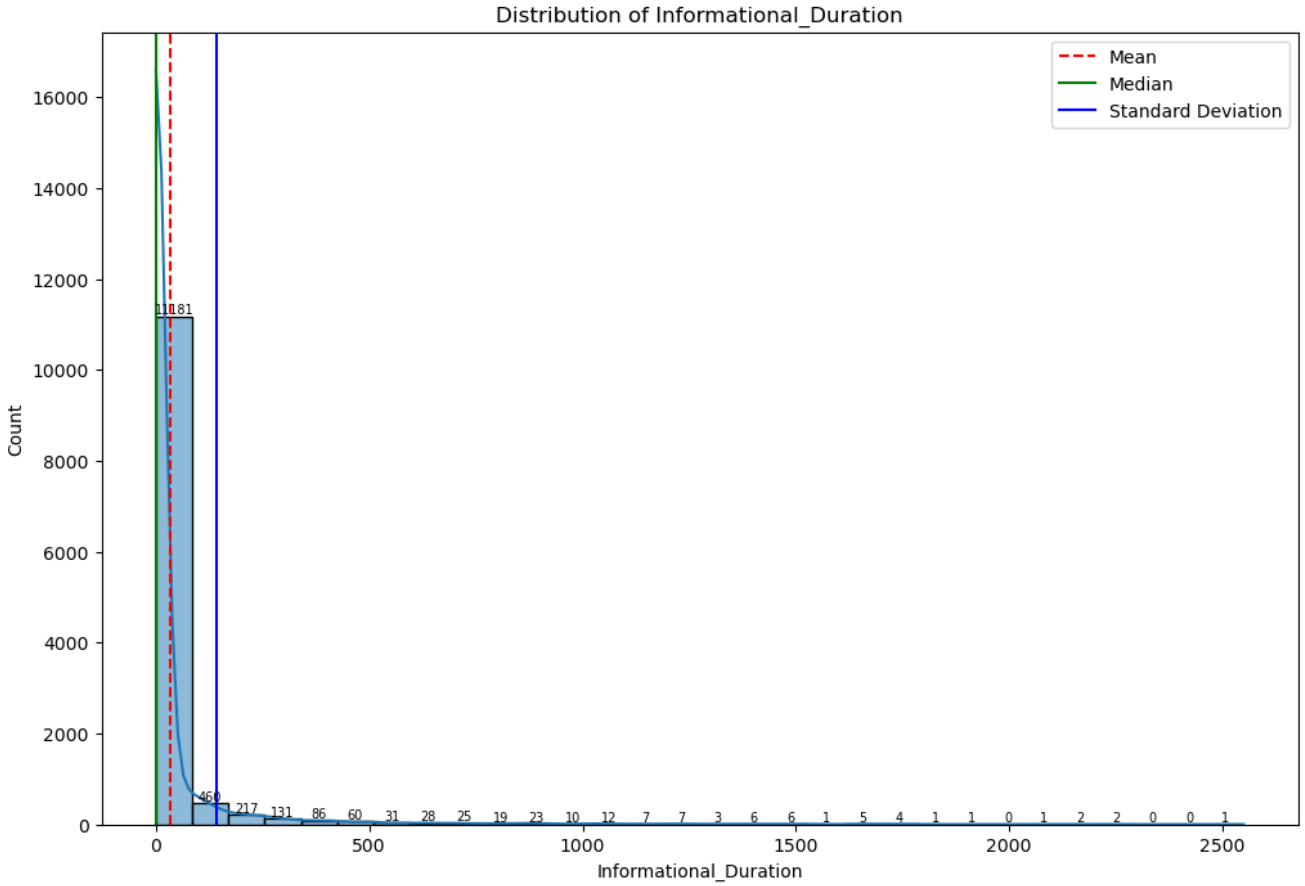
2.2 Descriptive Statistics Visualization

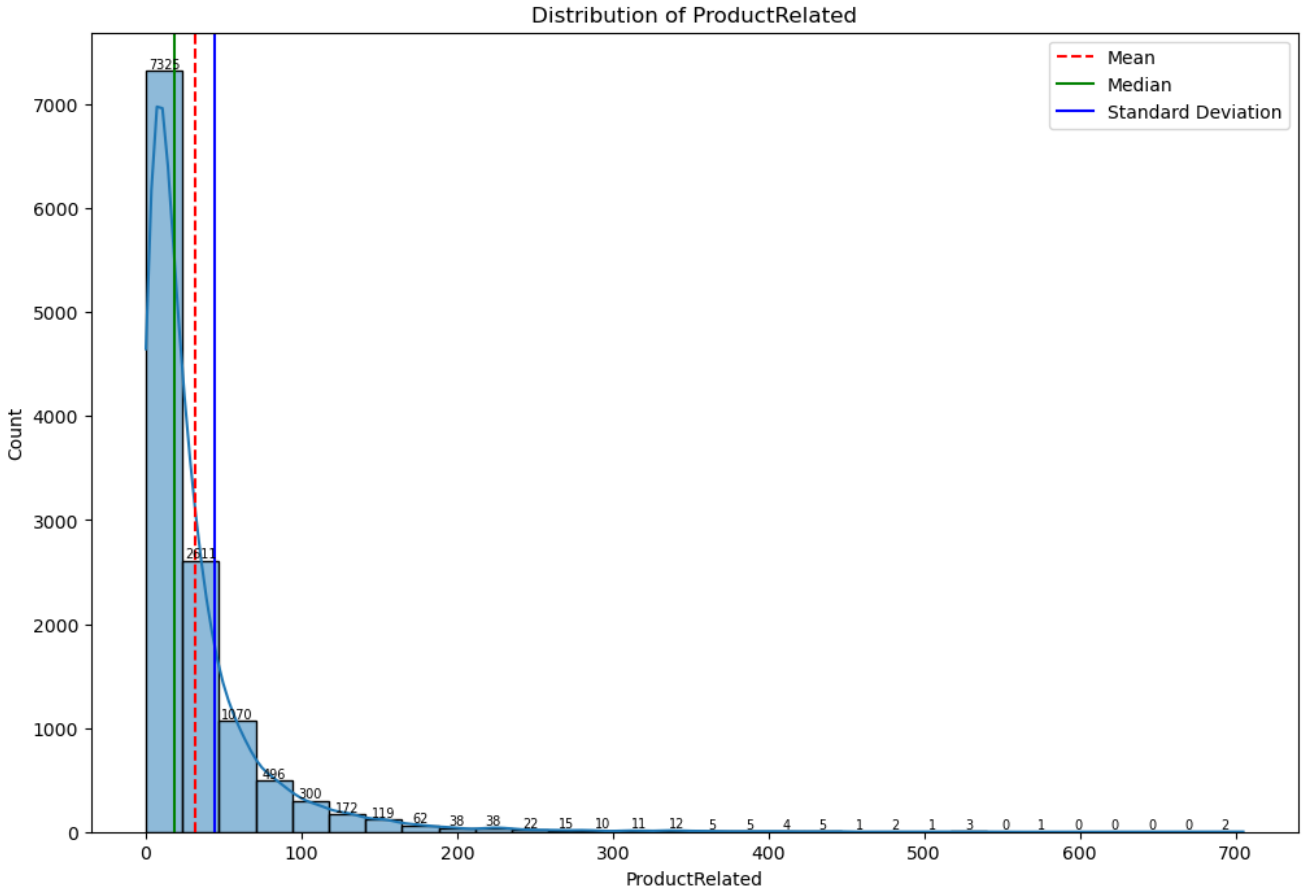
In [5]:
```
# Statistics for numerical columns
num_cols = ['Administrative', 'Administrative_Duration', 'Informational', 'Informational_Duration', 'ProductRela
for column in num_cols:
    plt.figure(figsize=(12,8))
    ax = sns.histplot(df[column], kde=True, bins=30)
    plt.title('Distribution of '+ column)
    plt.axvline(df[column].mean(), color='r', linestyle='--', label='Mean')
    plt.axvline(df[column].median(), color='g', linestyle='-', label='Median')
    plt.axvline(df[column].std(), color='b', linestyle='-', label='Standard Deviation')
    plt.legend()

    for p in ax.patches:
        ax.annotate(f'{p.get_height().astype(int)}', (p.get_x()+p.get_width()/2., p.get_height()+0.5), ha='cente
    plt.show()

# Even though the Revenue is the target variable, I am classing it as a kind of categorical column to see the sp
cat_cols = ['Month', 'OperatingSystems', 'Browser', 'Region', 'TrafficType', 'VisitorType', 'Weekend', 'Revenue'
for column in cat_cols:
    plt.figure(figsize=(12,8))
    ax = sns.countplot(x=column, data=df)
    plt.title('Count of '+ column)
    plt.xticks(rotation=90)

    for p in ax.patches:
        ax.annotate(f'{p.get_height()}', (p.get_x()+p.get_width()/2., p.get_height()+0.5), ha='center', va='bott
    plt.show()

```

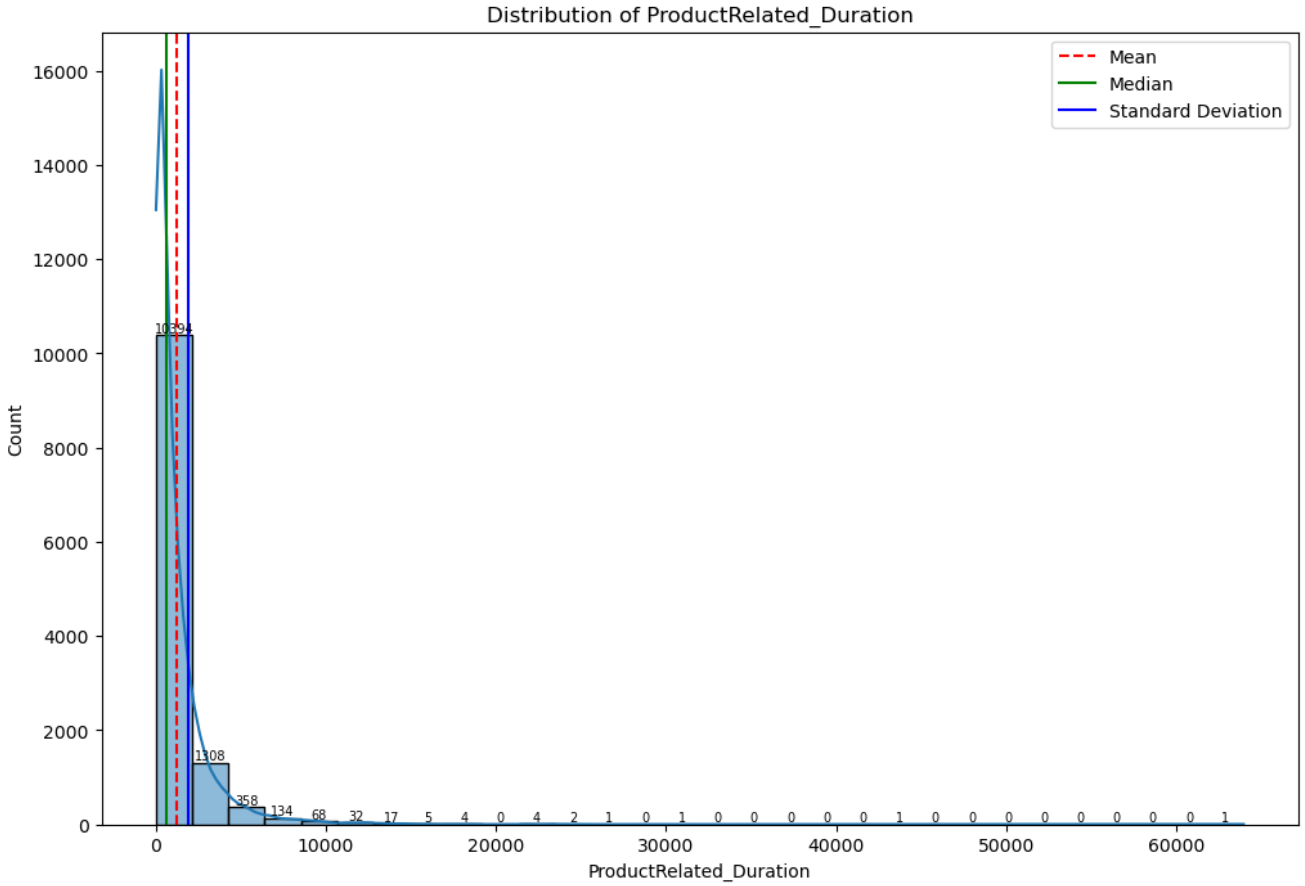


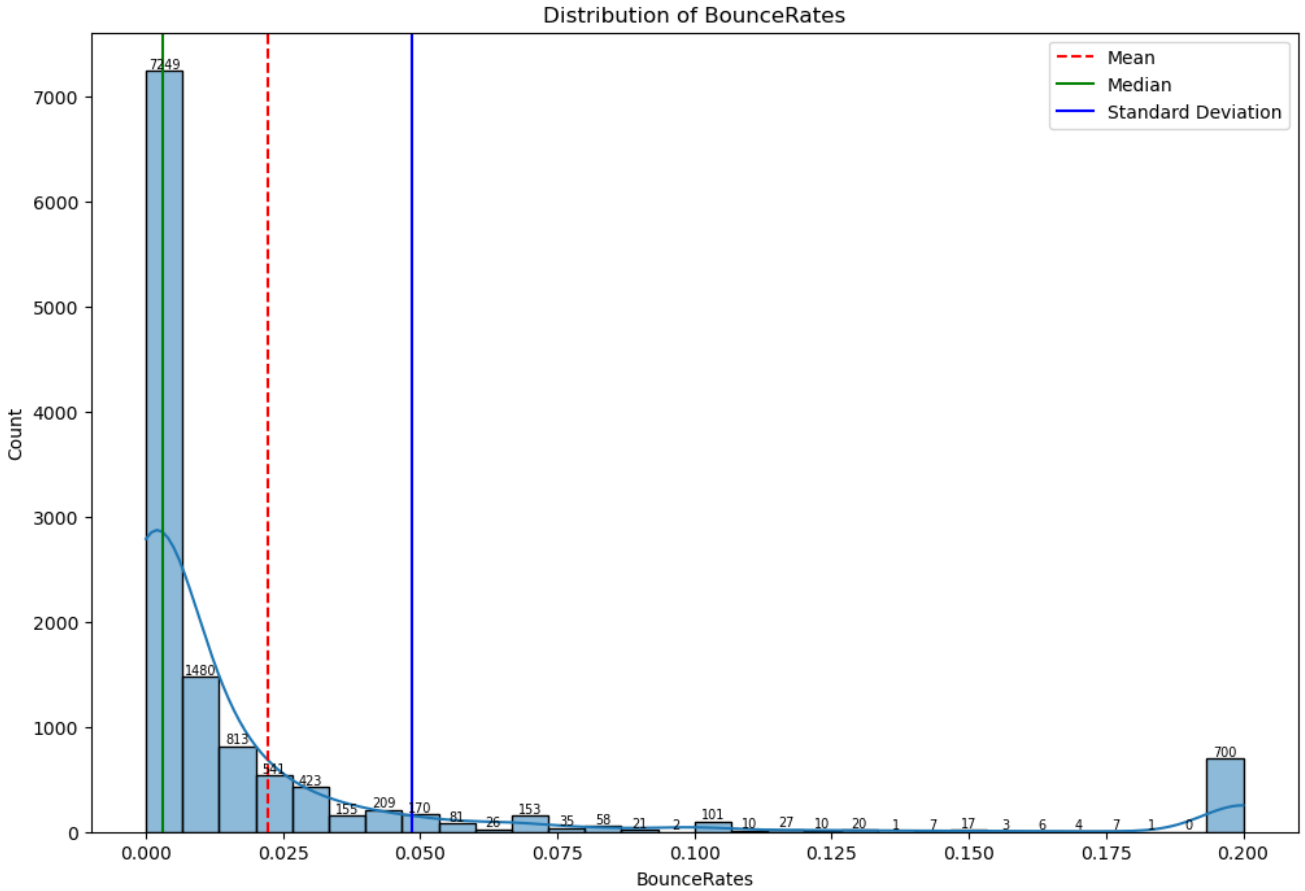

Distribution of Administrative_Duration
Mean
Median
Standard Deviation
Count
Administrative_Duration
0
500
1000
1500
2000
2500
3000
3500
2000
4000
6000
8000
10000
12000
9714
1392
581
247
122
87
47
36
25
12
12
13
9
10
8
3
1
2
3
1
0
1
0
2
1
0
0
0
0
1

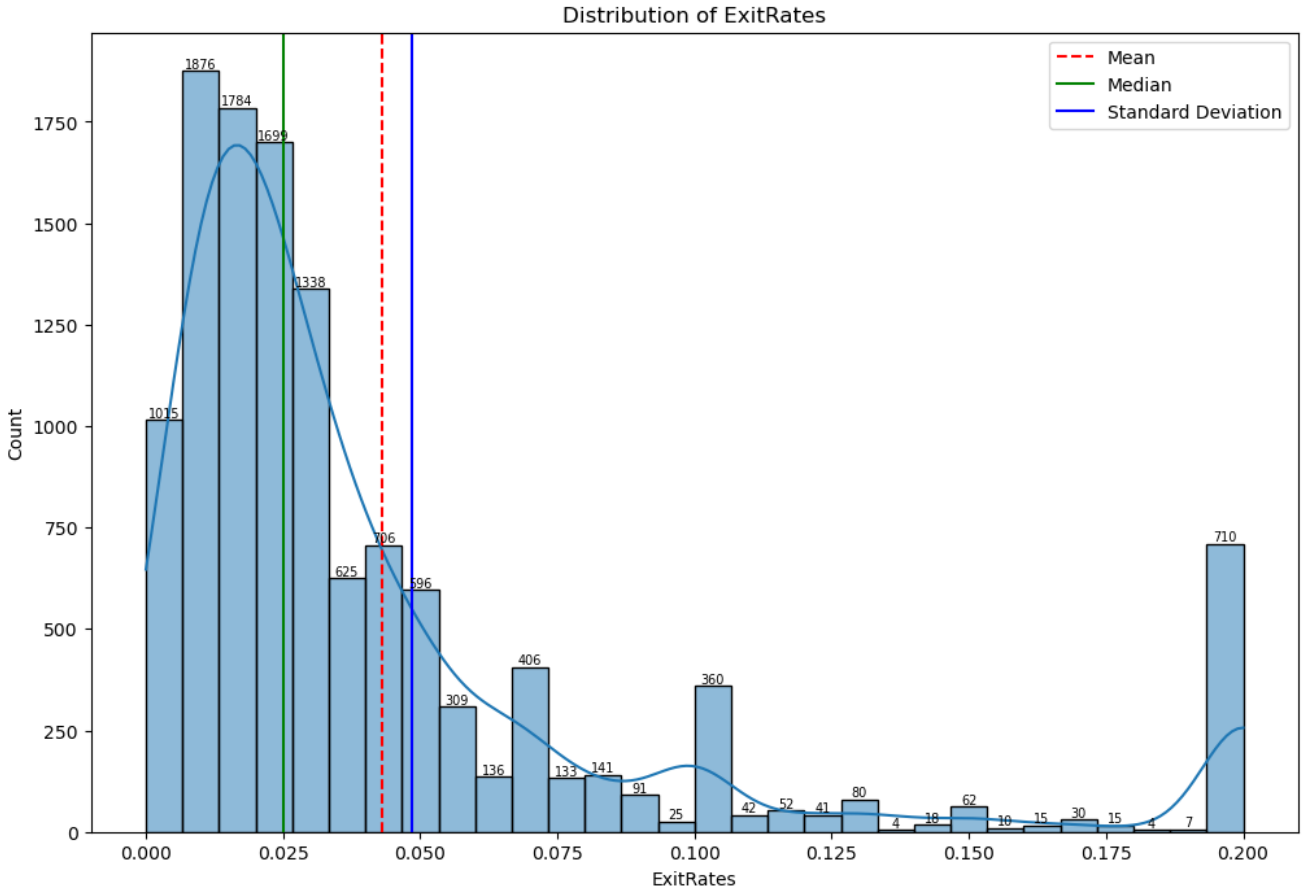

Distribution of Informational
Mean
Median
Standard Deviation
Count
Informational
0
5
10
15
20
25
2000
4000
6000
8000
10000
12000
14000
16000
9699
1041
728
380
0
222
99
78
36
0
14
15
7
1
0
5
1
2
0
0
1
0
0
0
0
0
0
0
0
1

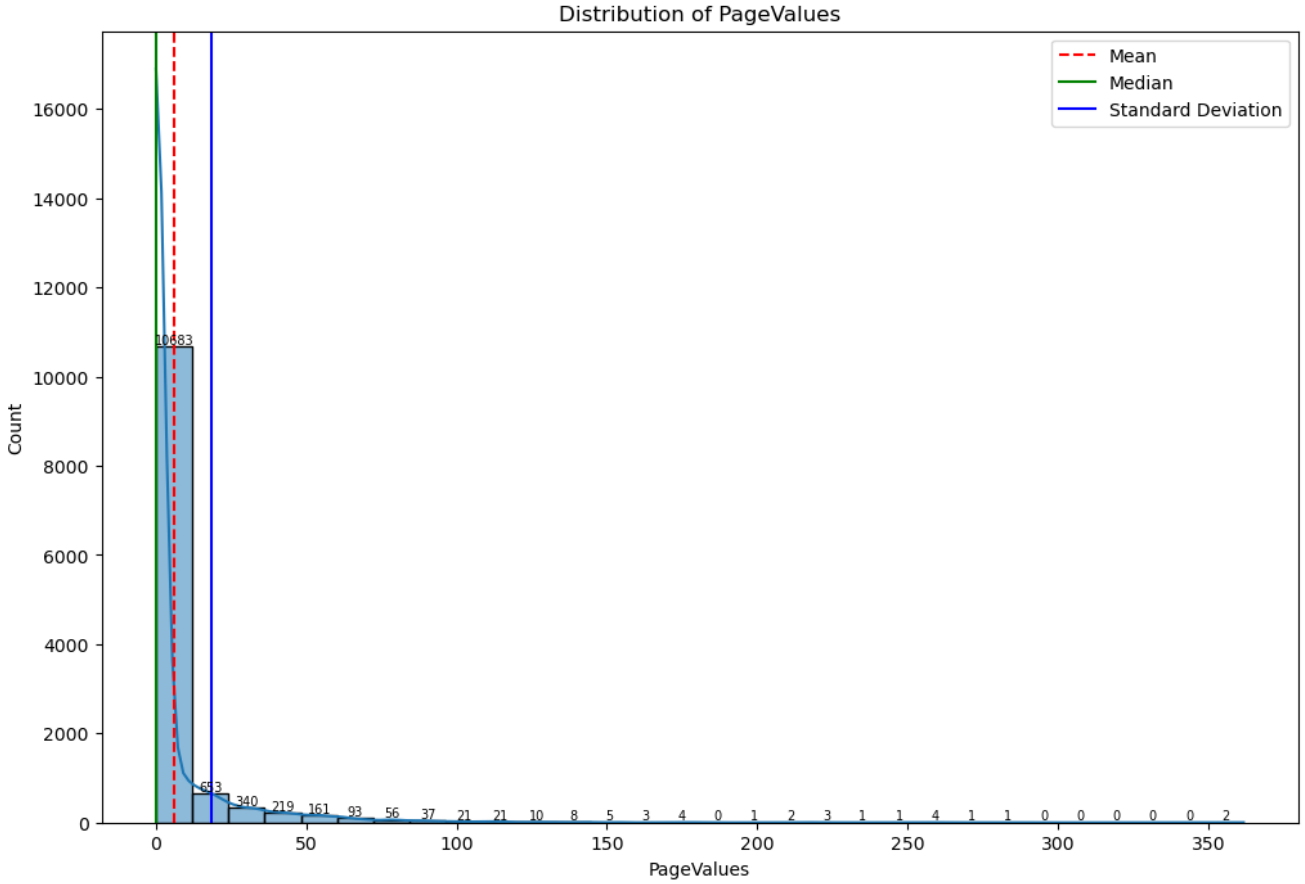

Distribution of Informational_Duration
Mean
Median
Standard Deviation
Count
Informational_Duration
0
500
1000
1500
2000
2500
2000
4000
6000
8000
10000
12000
14000
16000
11181
460
217
131
86
60
31
28
25
19
23
10
12
7
7
3
6
6
1
5
4
1
1
0
1
2
2
0
0
1

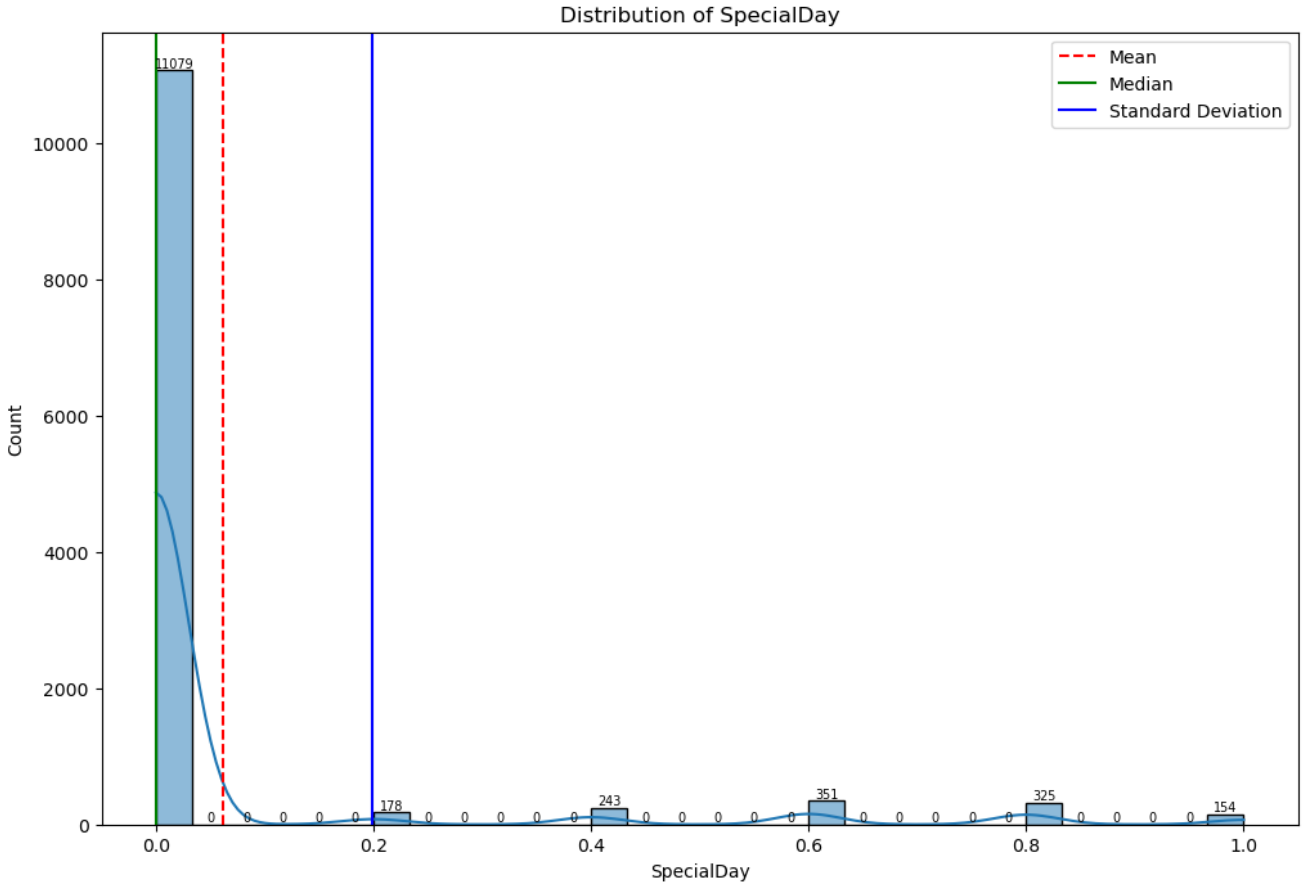

Distribution of ProductRelated
Mean
Median
Standard Deviation
Count
ProductRelated
0
100
200
300
400
500
600
700
1000
2000
3000
4000
5000
6000
7000
7326
2611
1070
496
300
172
119
62
38
38
22
15
10
11
12
5
5
4
5
1
2
1
3
0
1
0
0
0
0
2

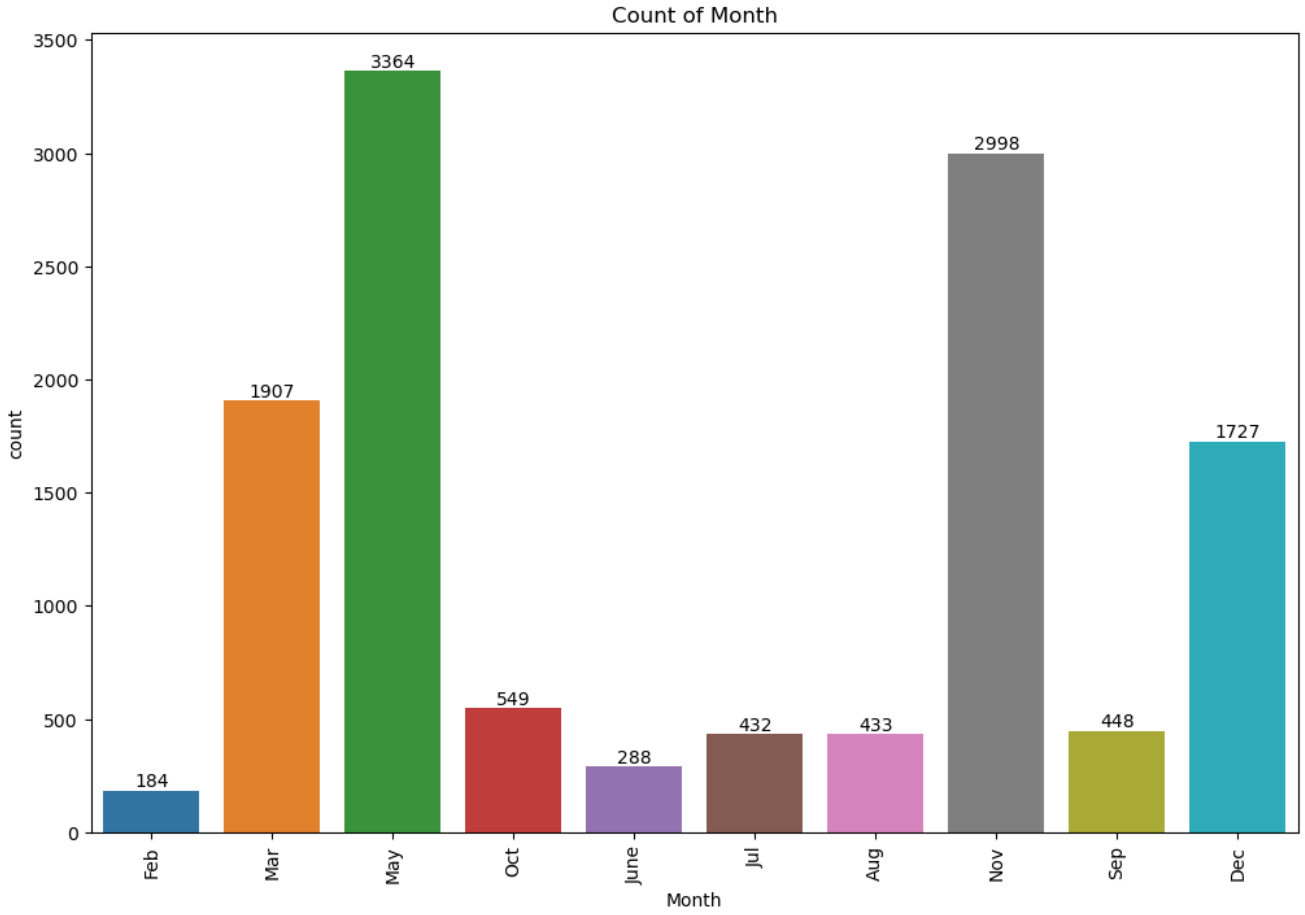

Distribution of ProductRelated_Duration
Mean
Median
Standard Deviation
Count
ProductRelated_Duration
0
10000
20000
30000
40000
50000
60000
2000
4000
6000
8000
10000
12000
14000
16000
1308
358
134
68
32
17
5
4
0
4
2
1
0
1
0
0
0
0
0
1
0
0
0
0
0
0
0
1

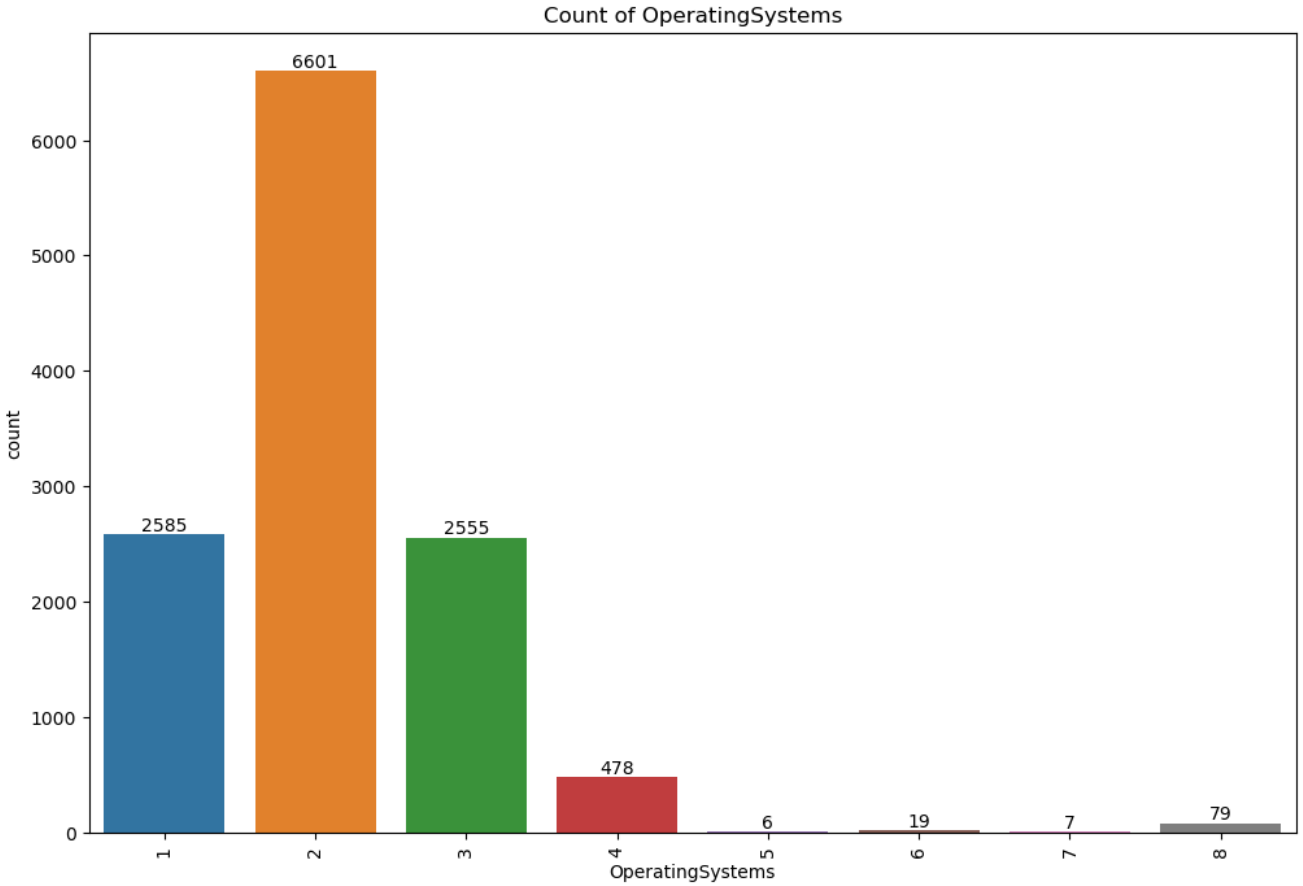

Distribution of BounceRates
Mean
Median
Standard Deviation
Count
BounceRates
0.000
0.025
0.050
0.075
0.100
0.125
0.150
0.175
0.200
1000
2000
3000
4000
5000
6000
7000
7249
1480
813
521
423
155
209
170
81
26
153
35
58
21
2
101
10
27
10
20
1
7
17
3
6
4
7
1
0
700

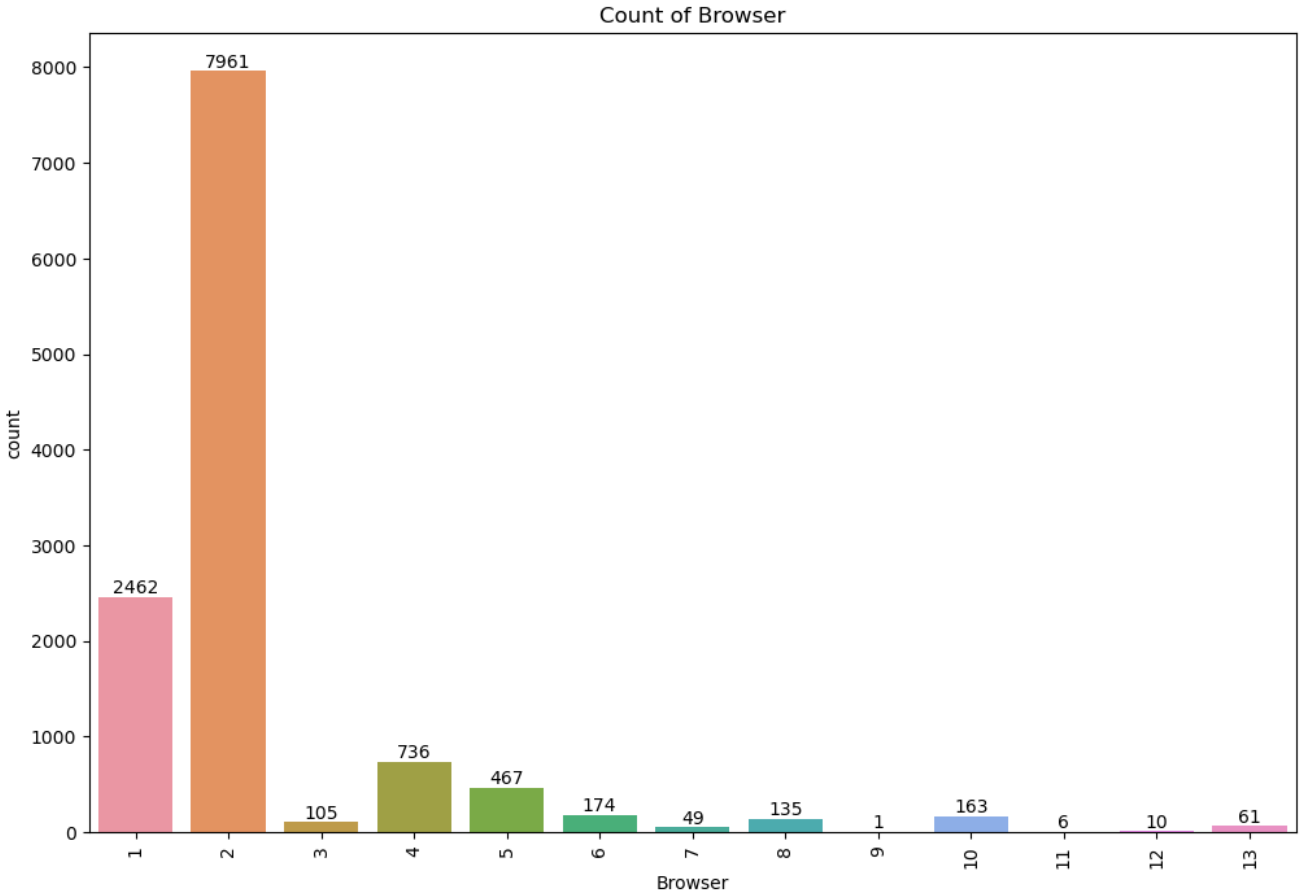

Distribution of ExitRates
Mean
Median
Standard Deviation
Count
ExitRates
0.000
0.025
0.050
0.075
0.100
0.125
0.150
0.175
0.200
0
250
500
750
1000
1250
1500
1750
1876
1784
1699
1338
1015
706
625
596
406
360
309
136
133
141
91
25
42
52
41
80
4
18
62
10
15
30
15
4
7
710

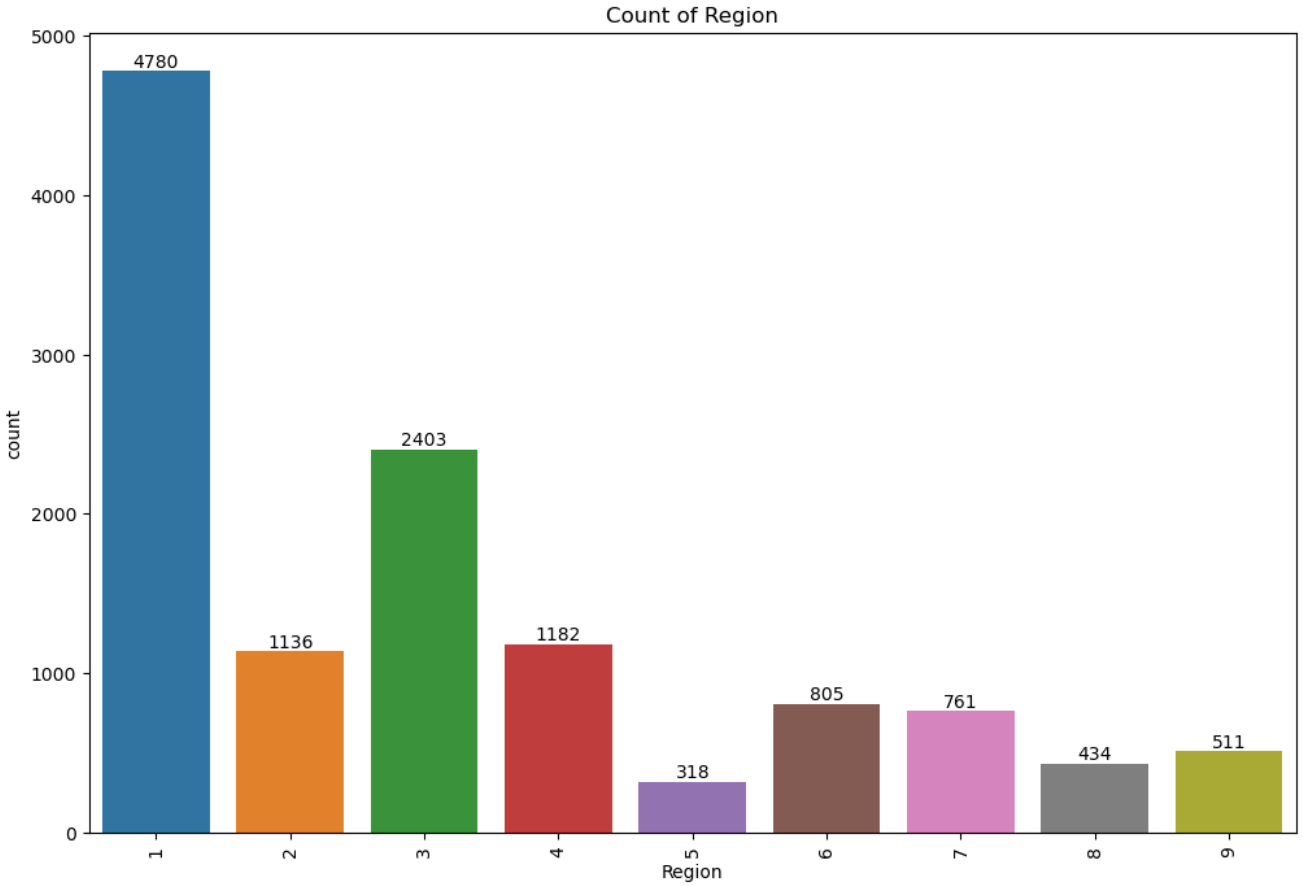

Distribution of PageValues
Mean
Median
Standard Deviation
Count
PageValues
0
50
100
150
200
250
300
350
0
2000
4000
6000
8000
10000
12000
14000
16000
10683
653
340
219
161
93
56
37
21
21
10
8
5
3
4
0
1
2
3
1
1
4
1
1
0
0
0
0
0
2

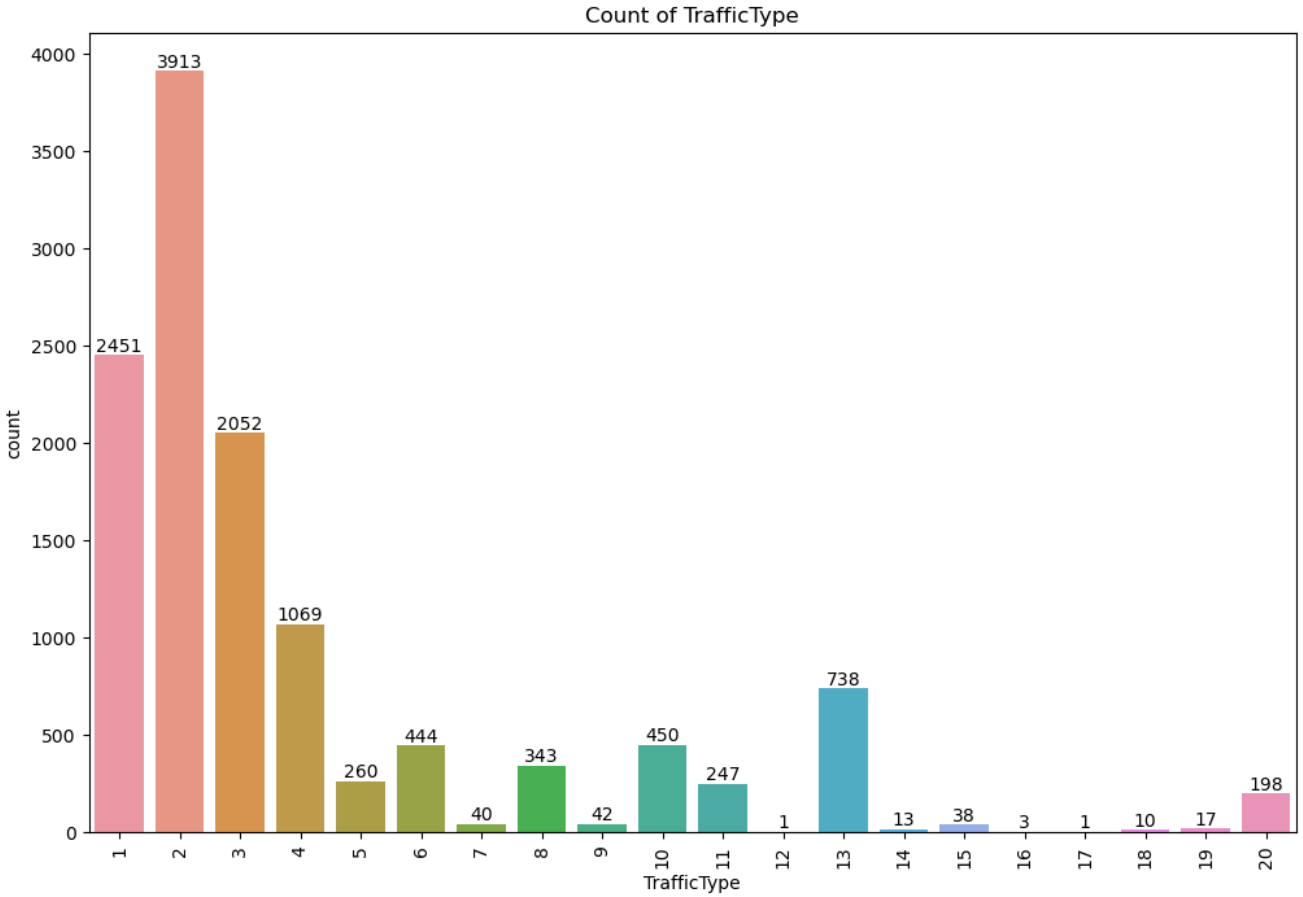

Distribution of SpecialDay
Mean
Median
Standard Deviation
Count
SpecialDay
11079
178
243
351
325
154

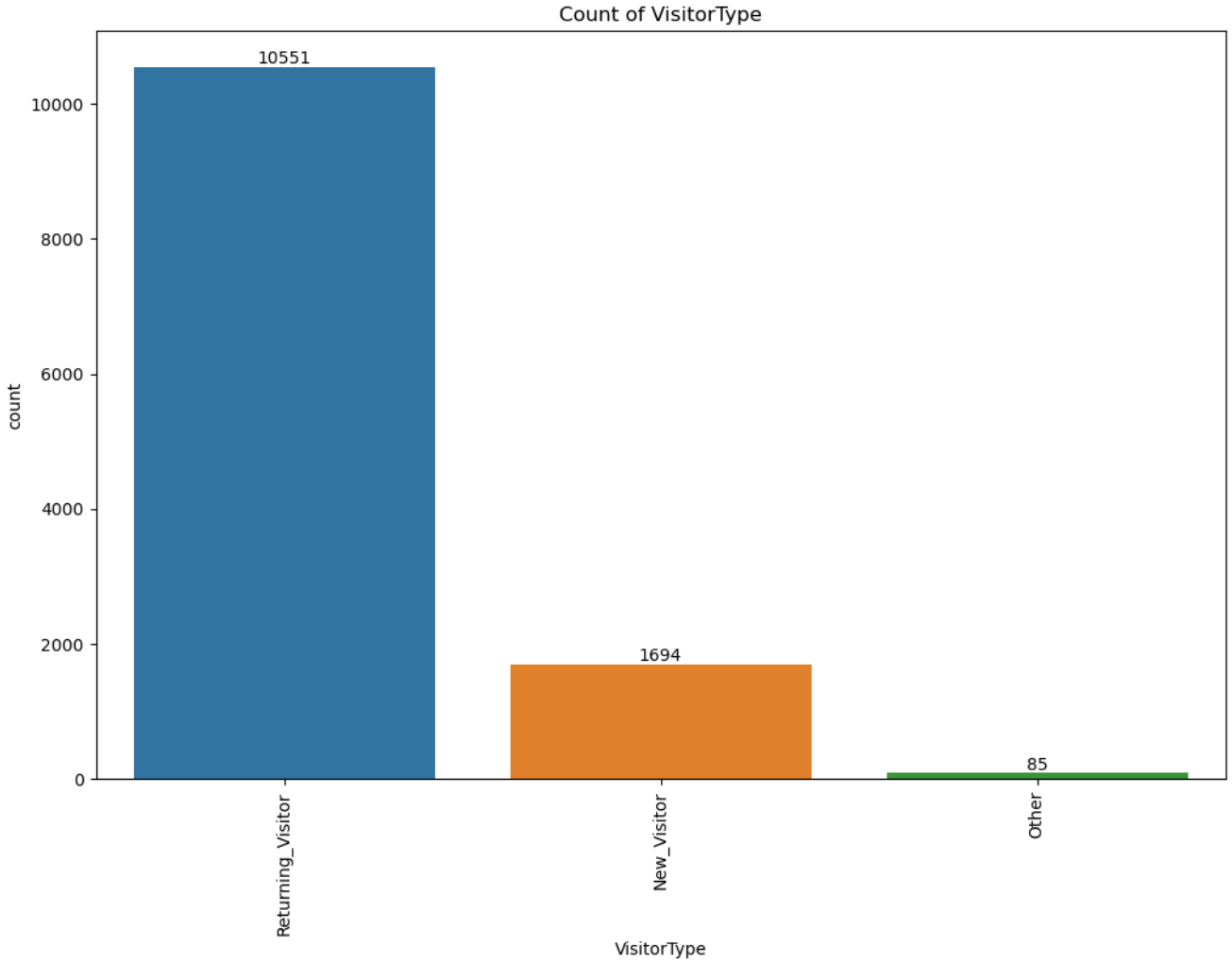

Count of Month
count
Month
184
1907
3364
549
288
432
433
2998
448
1727
Feb
Mar
May
Oct
June
Jul
Aug
Nov
Sep
Dec

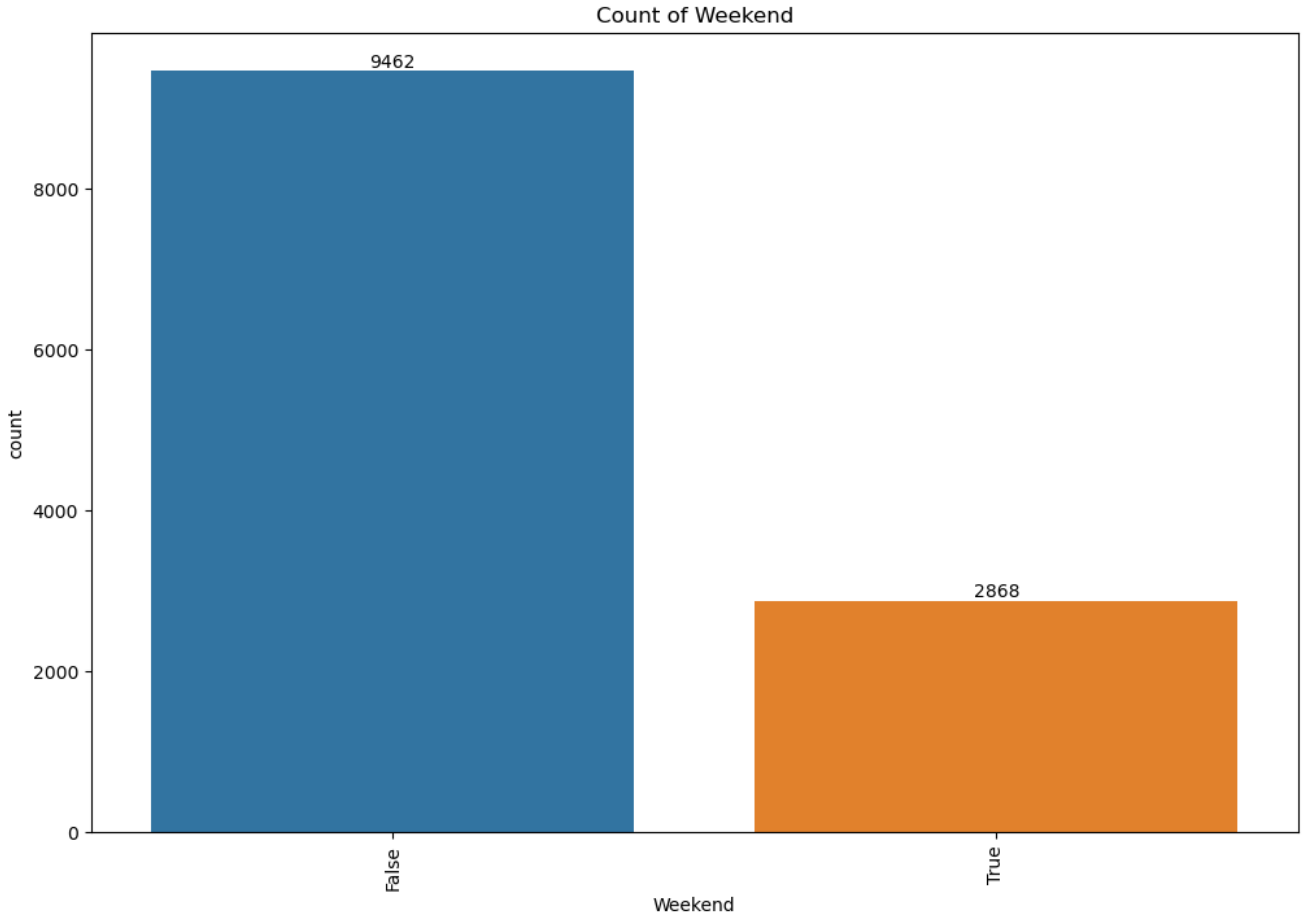

Count of OperatingSystems
count
OperatingSystems
6601
2585
2555
478
6
19
7
79
0
1000
2000
3000
4000
5000
6000
1
2
3
4
5
6
7
8

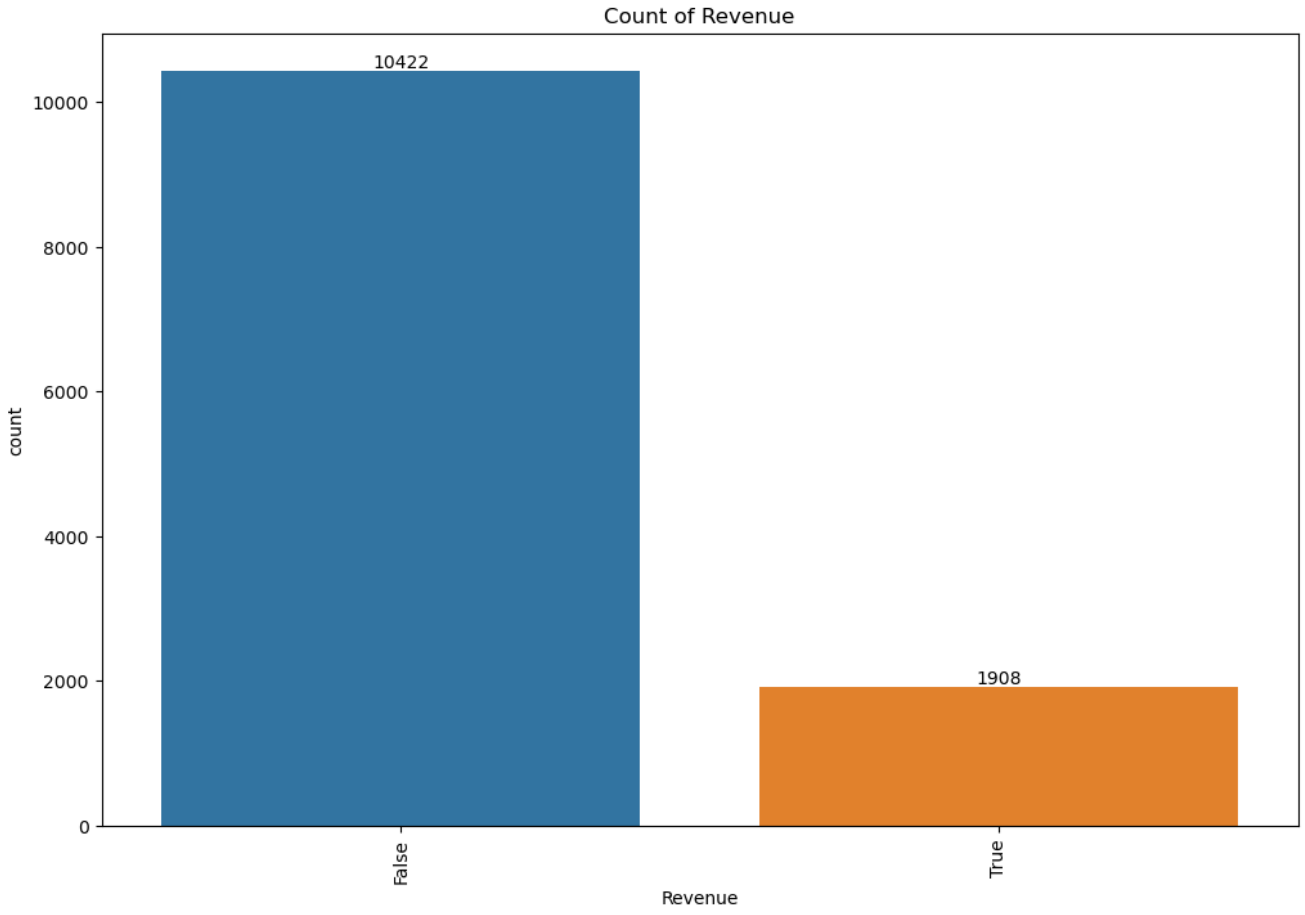

Count of Browser
count
Browser
7961
2462
105
736
467
174
49
135
1
163
6
10
61
0
1000
2000
3000
4000
5000
6000
7000
8000
1
2
3
4
5
6
7
8
9
10
11
12
13


Count of Region
count
Region
4780
1136
2403
1182
318
805
761
434
511


Count of TrafficType
count
TrafficType
2451
3913
2052
1069
260
444
40
343
42
450
247
1
738
13
38
3
1
10
17
198


Count of VisitorType
10551
1694
85
10000
8000
6000
4000
2000
0
count
Returning_Visitor
New_Visitor
Other
VisitorType

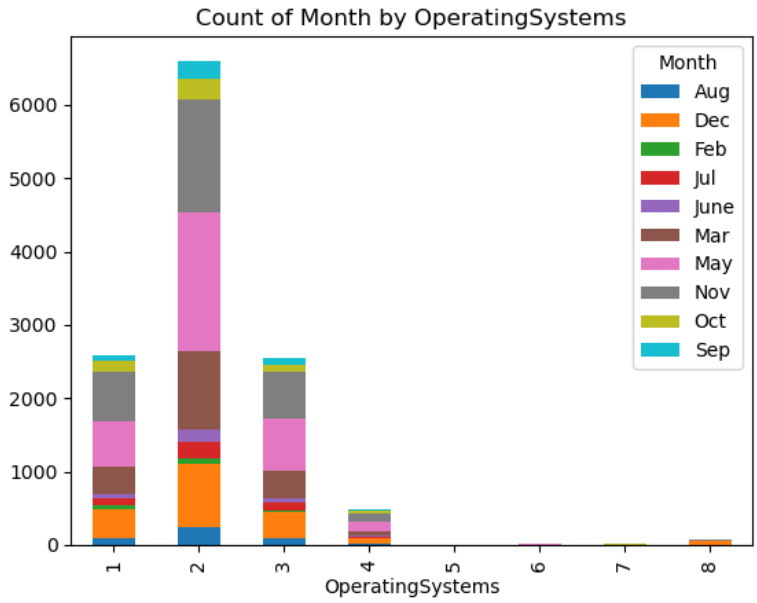

Count of Weekend
9462
2868
8000
6000
4000
2000
0
count
False
True
Weekend


Count of Revenue
10422
1908
10000
8000
6000
4000
2000
0
count
False
True
Revenue

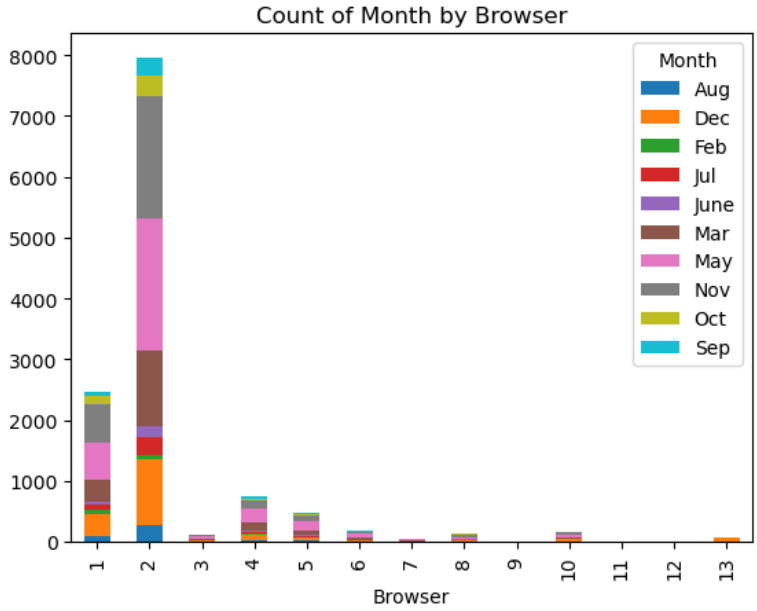
In [6]:

```
compare_with = ['Month', 'Region', 'Revenue']

for var in compare_with:
    other_vars = [col for col in cat_cols if col != var]
    for other_var in other_vars:
        plt.figure(figsize=(19,9))
        df.groupby(var)[other_var].value_counts().unstack(0).plot(kind='bar', stacked=True)
        plt.title(f'Count of {var} by {other_var}')
        plt.xticks(rotation=90)
        plt.show()

```

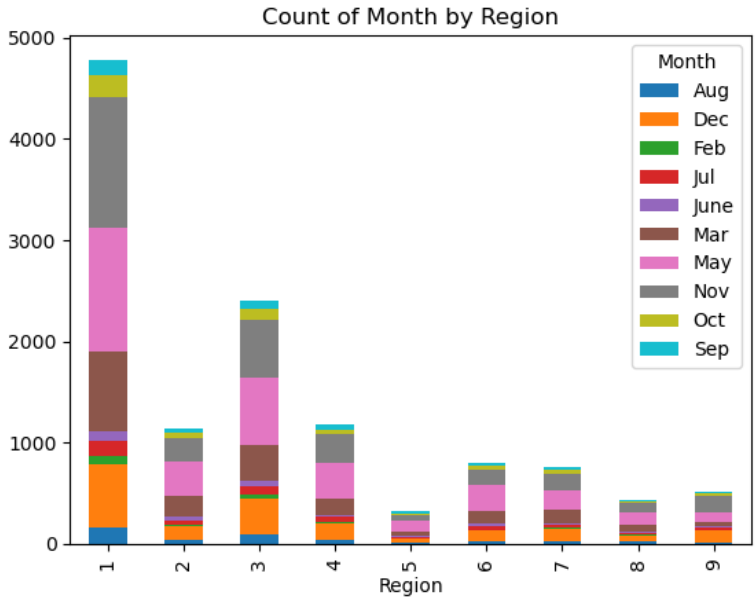
```
<Figure size 1900x900 with 0 Axes>
```



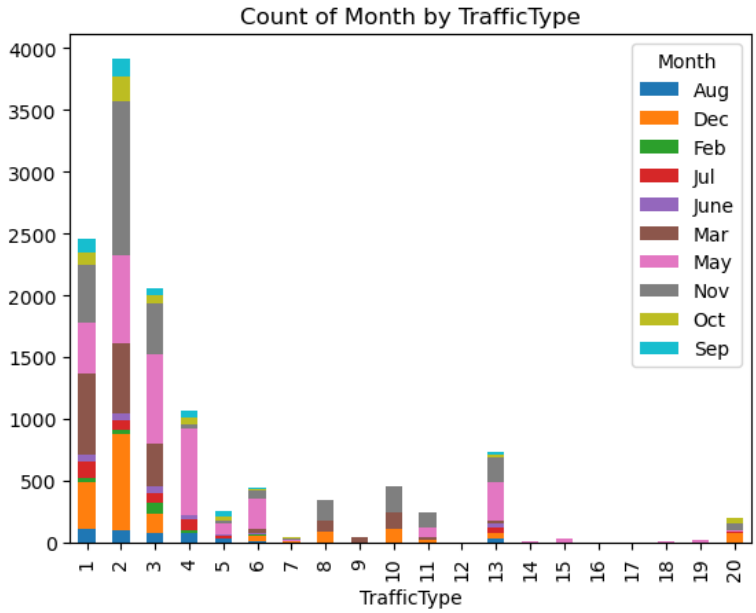

```
<Figure size 1900x900 with 0 Axes>
```



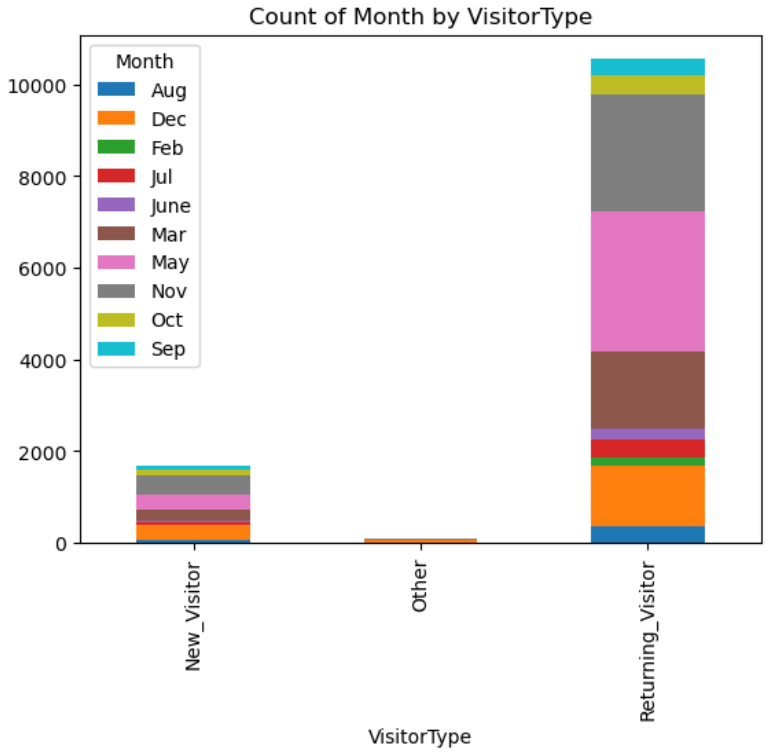

```
<Figure size 1900x900 with 0 Axes>
```



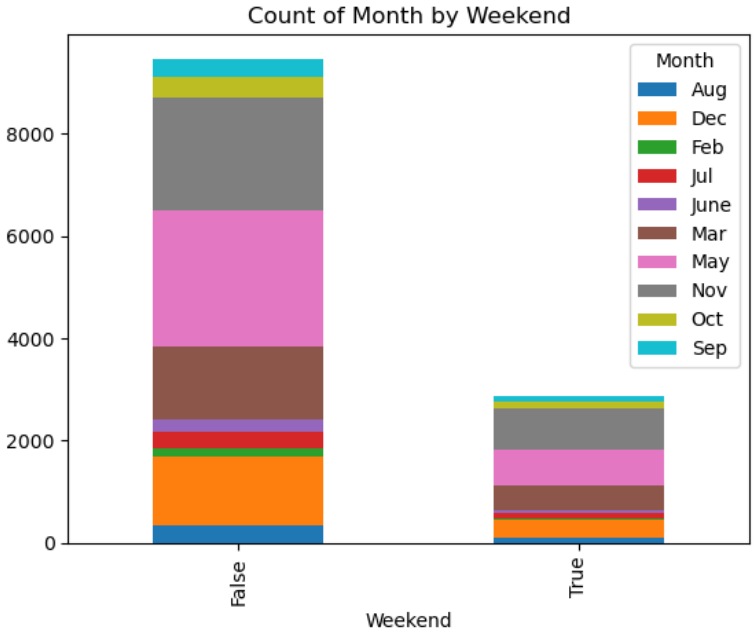

```
<Figure size 1900x900 with 0 Axes>
```



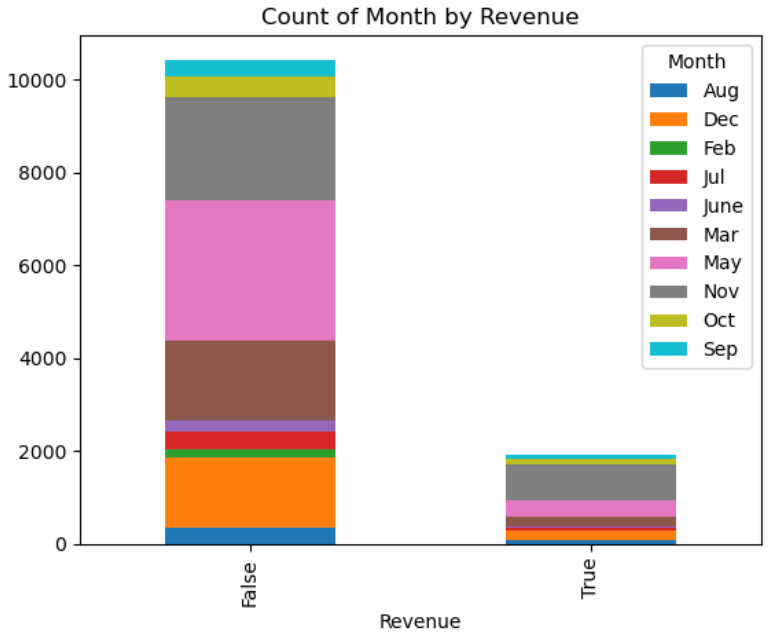

```
<Figure size 1900x900 with 0 Axes>
```



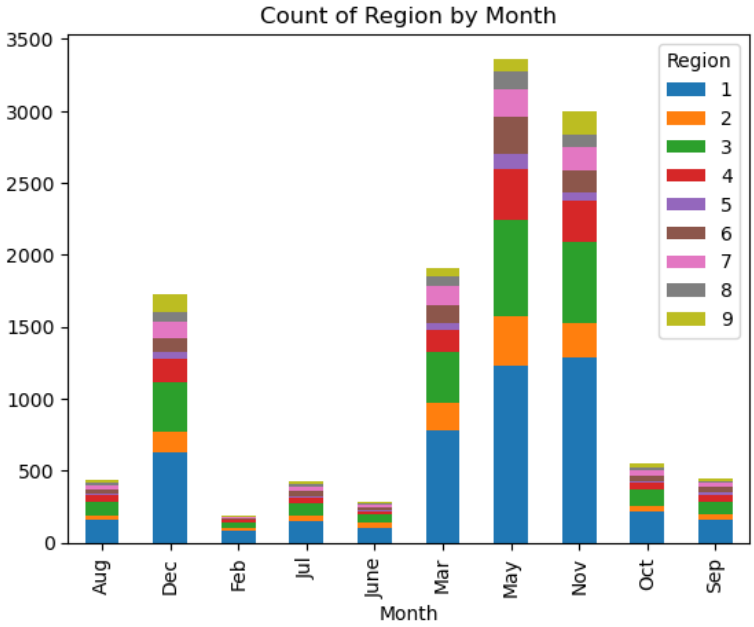

```
<Figure size 1900x900 with 0 Axes>
```



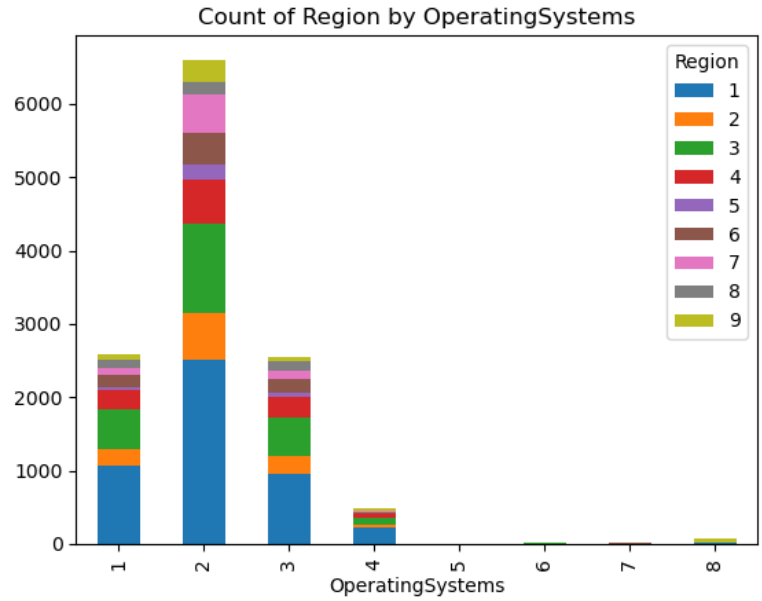

```
<Figure size 1900x900 with 0 Axes>
```



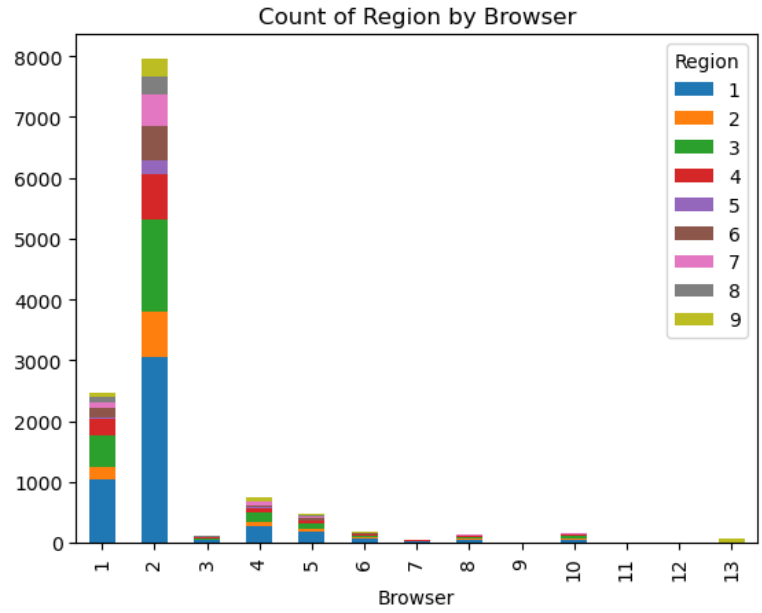
<Figure size 1900x900 with 0 Axes>



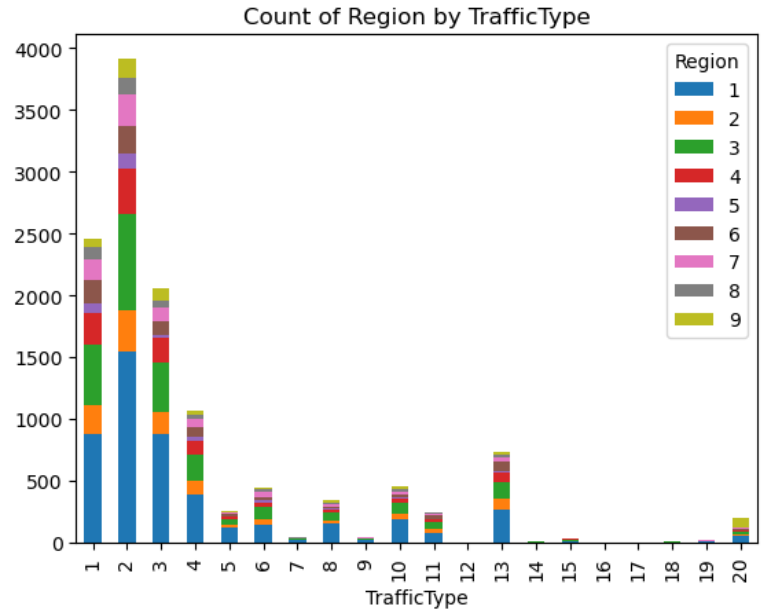
<Figure size 1900x900 with 0 Axes>



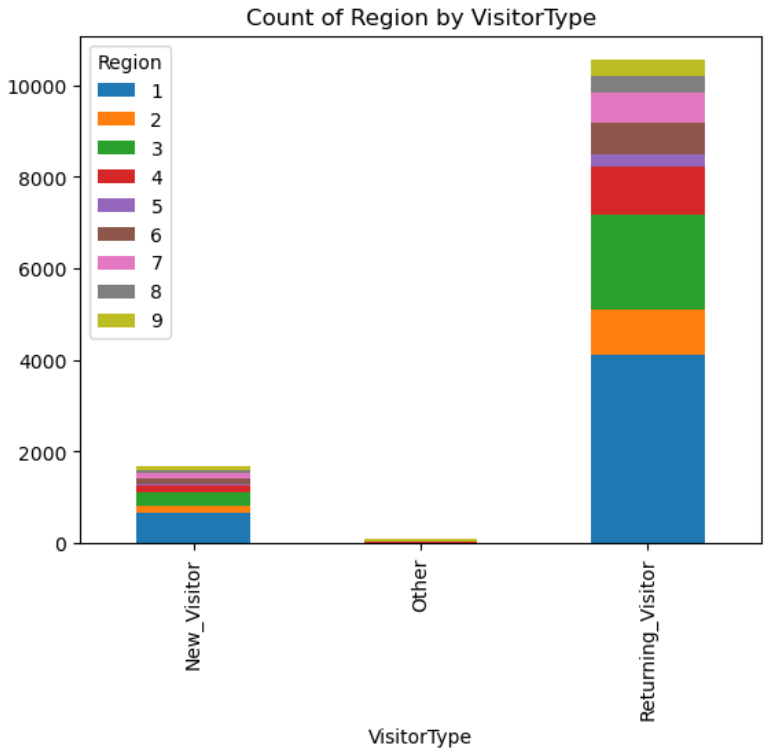

```
<Figure size 1900x900 with 0 Axes>
```



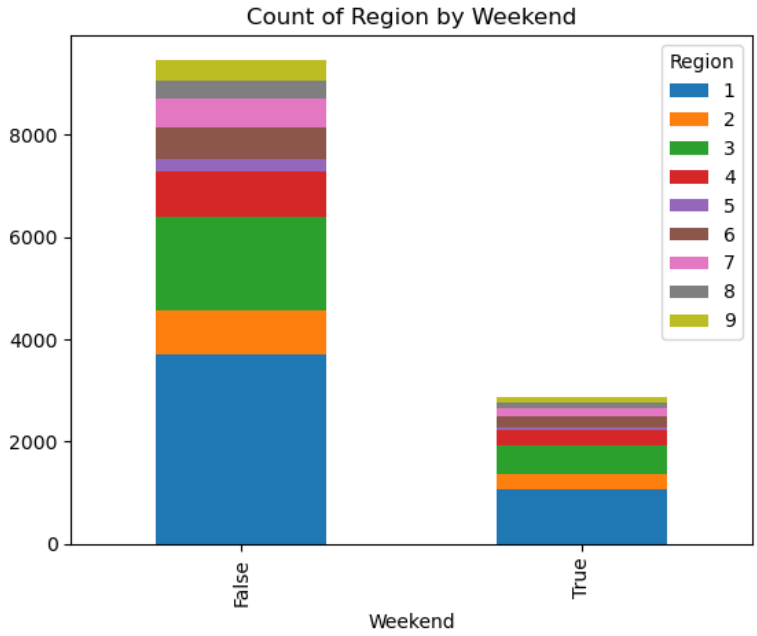

```
<Figure size 1900x900 with 0 Axes>
```



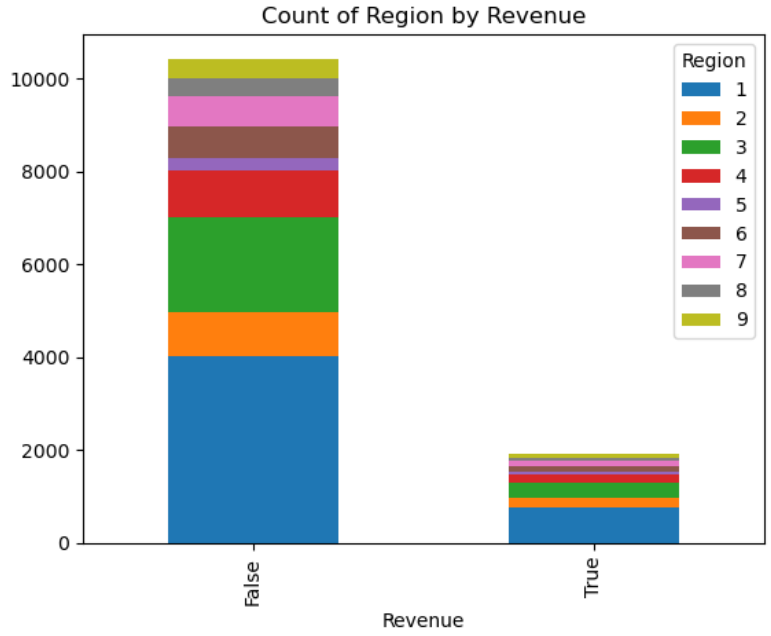

```
<Figure size 1900x900 with 0 Axes>
```



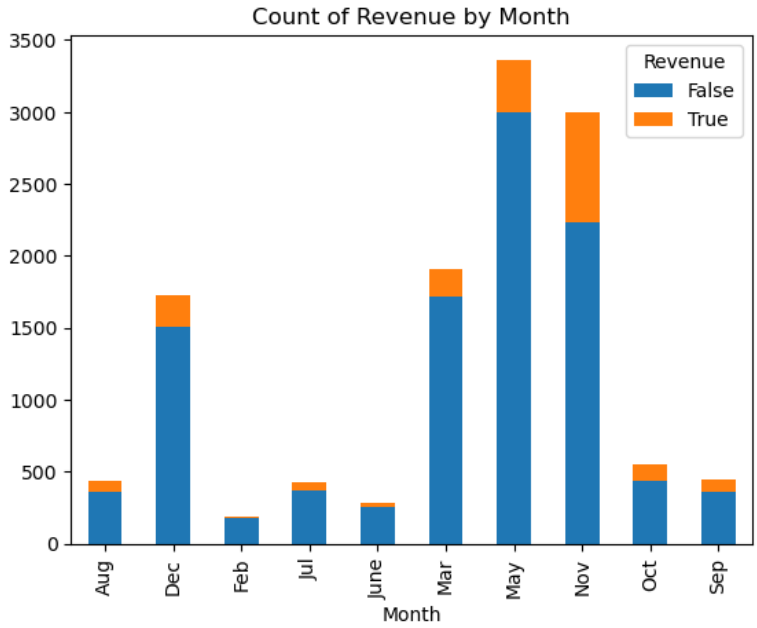

```
<Figure size 1900x900 with 0 Axes>
```



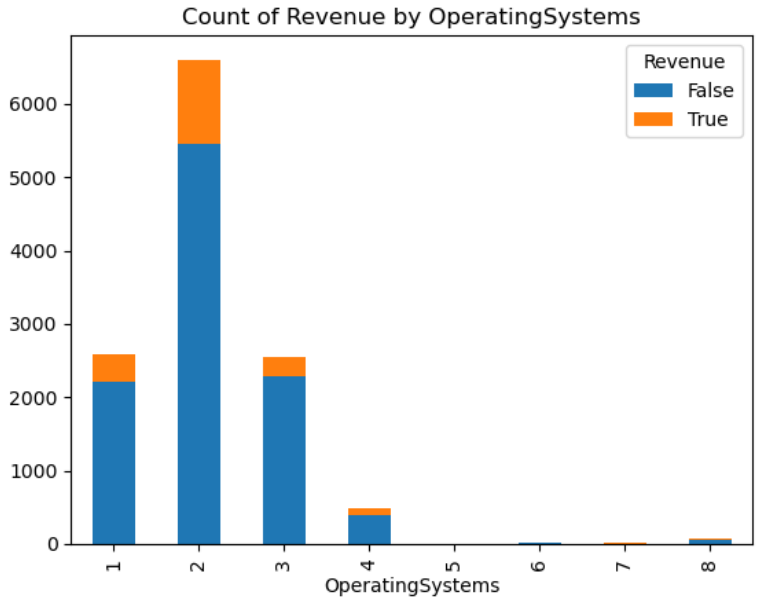
<Figure size 1900x900 with 0 Axes>



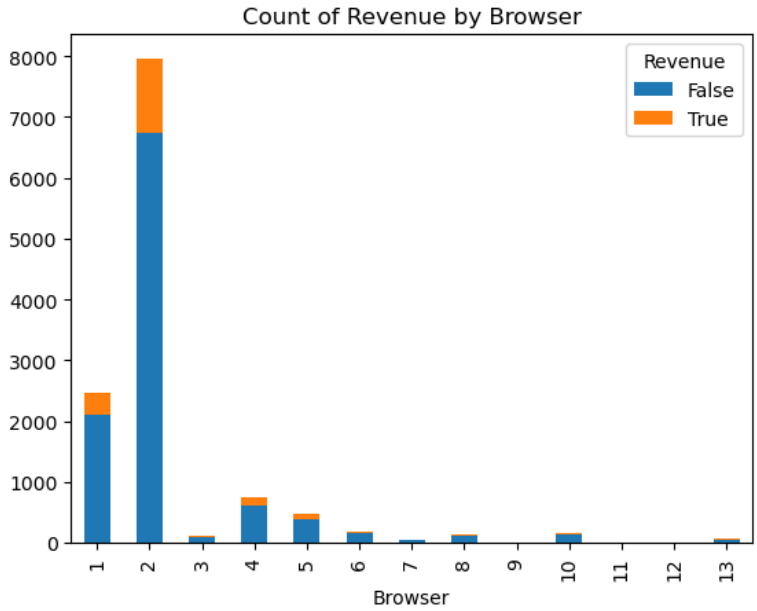
<Figure size 1900x900 with 0 Axes>



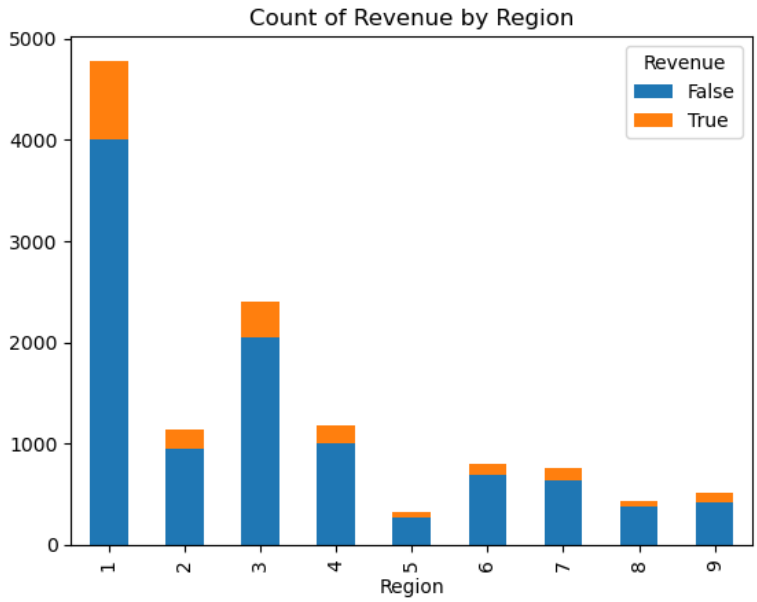

```
<Figure size 1900x900 with 0 Axes>
```



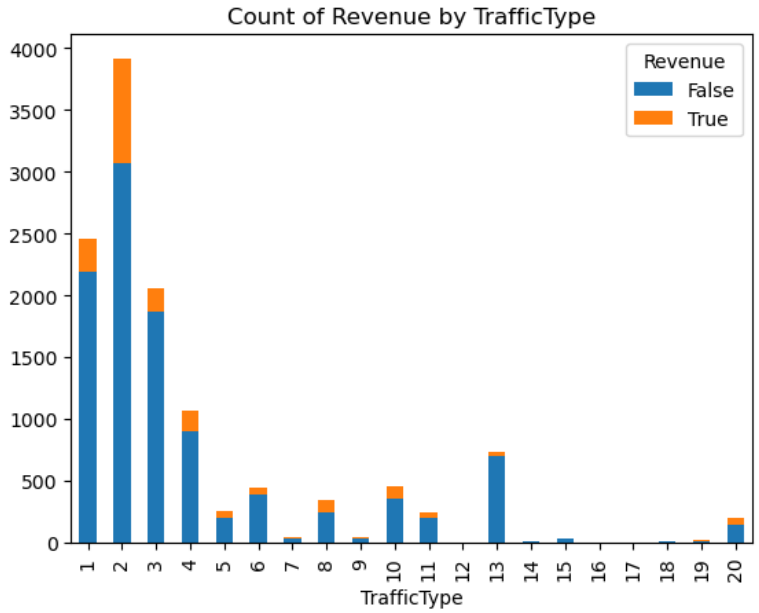

```
<Figure size 1900x900 with 0 Axes>
```



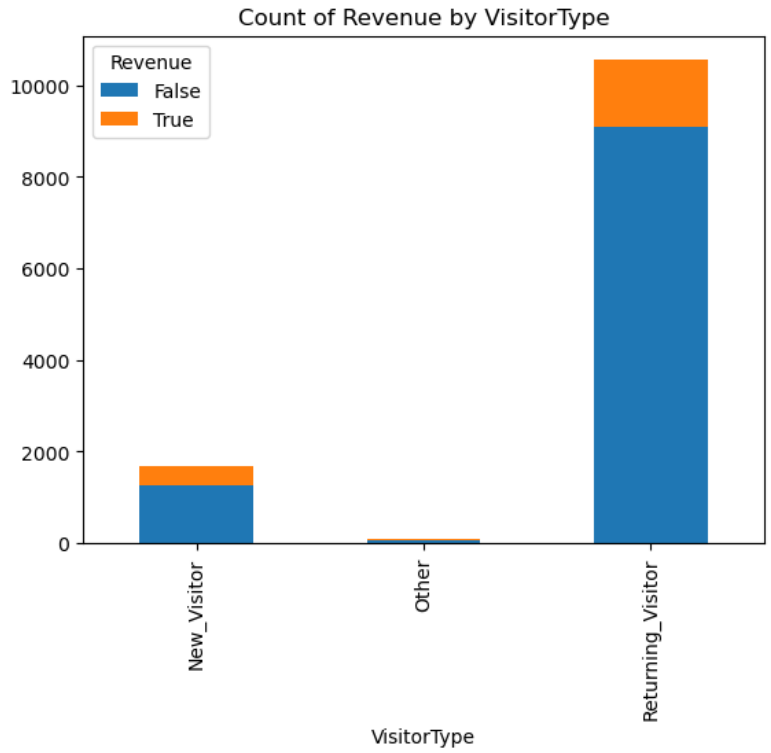

```
<Figure size 1900x900 with 0 Axes>
```



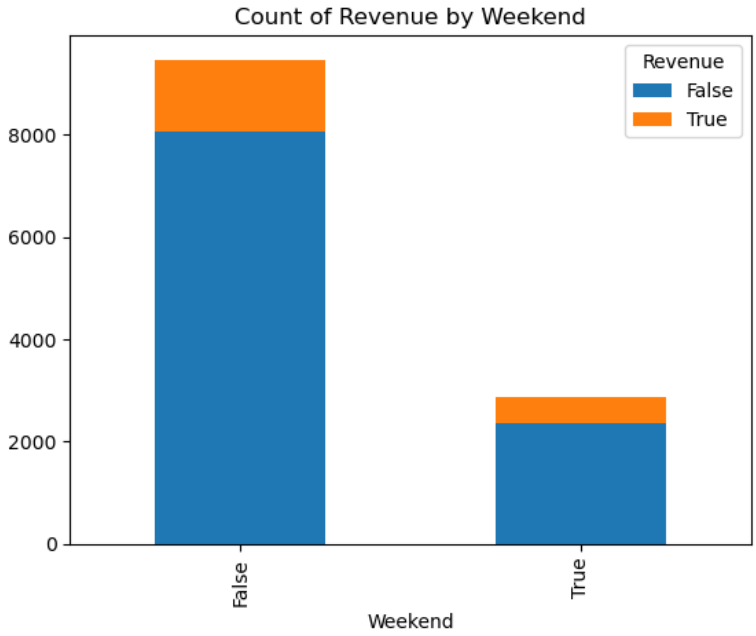

```
<Figure size 1900x900 with 0 Axes>
```




```
<Figure size 1900x900 with 0 Axes>
```

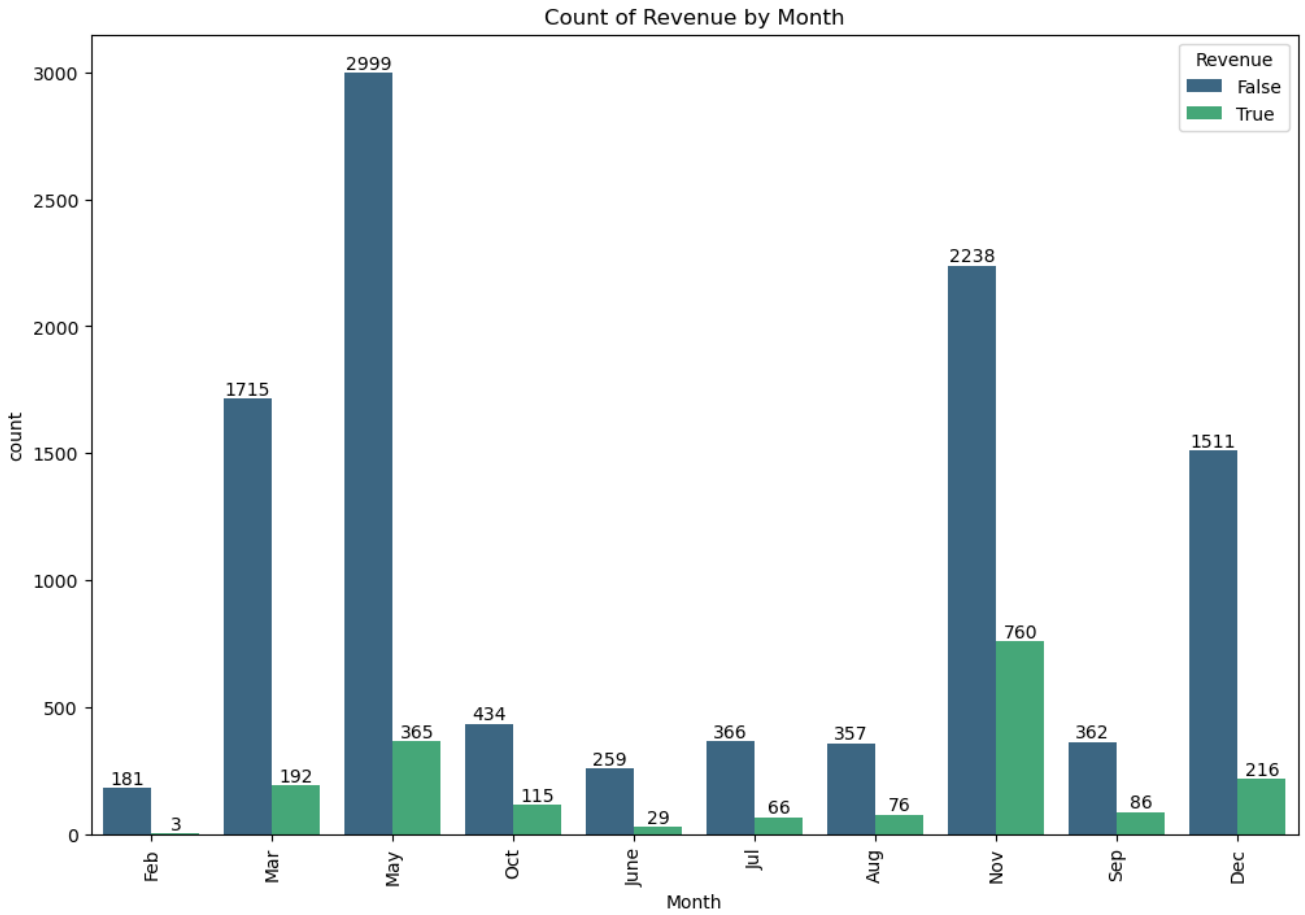



```
<Figure size 1900x900 with 0 Axes>
```



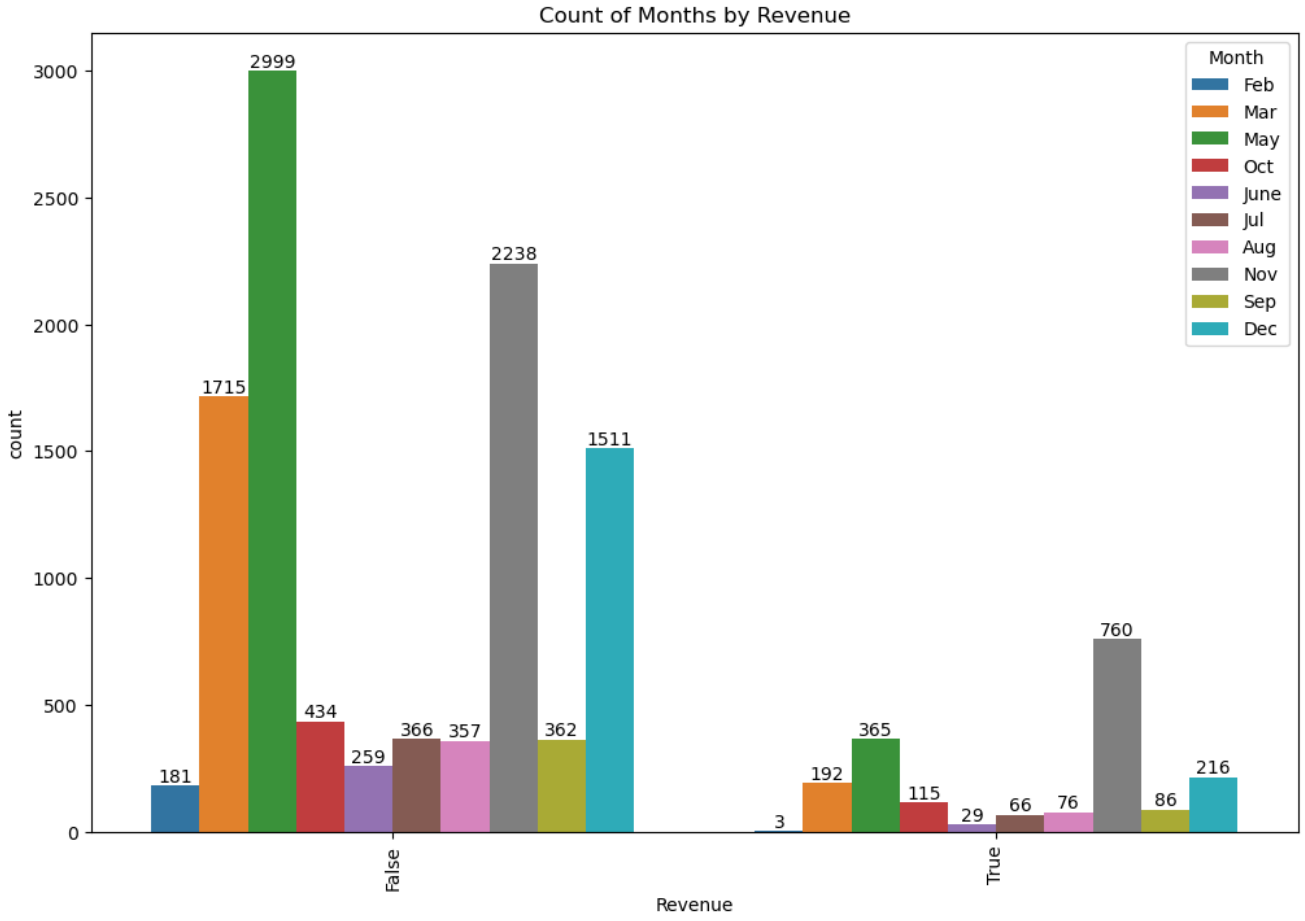
In [7]:

```
plt.figure(figsize=(12,8))
ax = sns.countplot(x='Month', hue='Revenue', data=df, palette='viridis')
plt.title('Count of Revenue by Month')
plt.xticks(rotation=90)

# Display frequency above each bar
for p in ax.patches:
    ax.annotate(f'{p.get_height()}', (p.get_x()+p.get_width()/2., p.get_height()+0.5), ha='center', va='bottom'
plt.show()

```



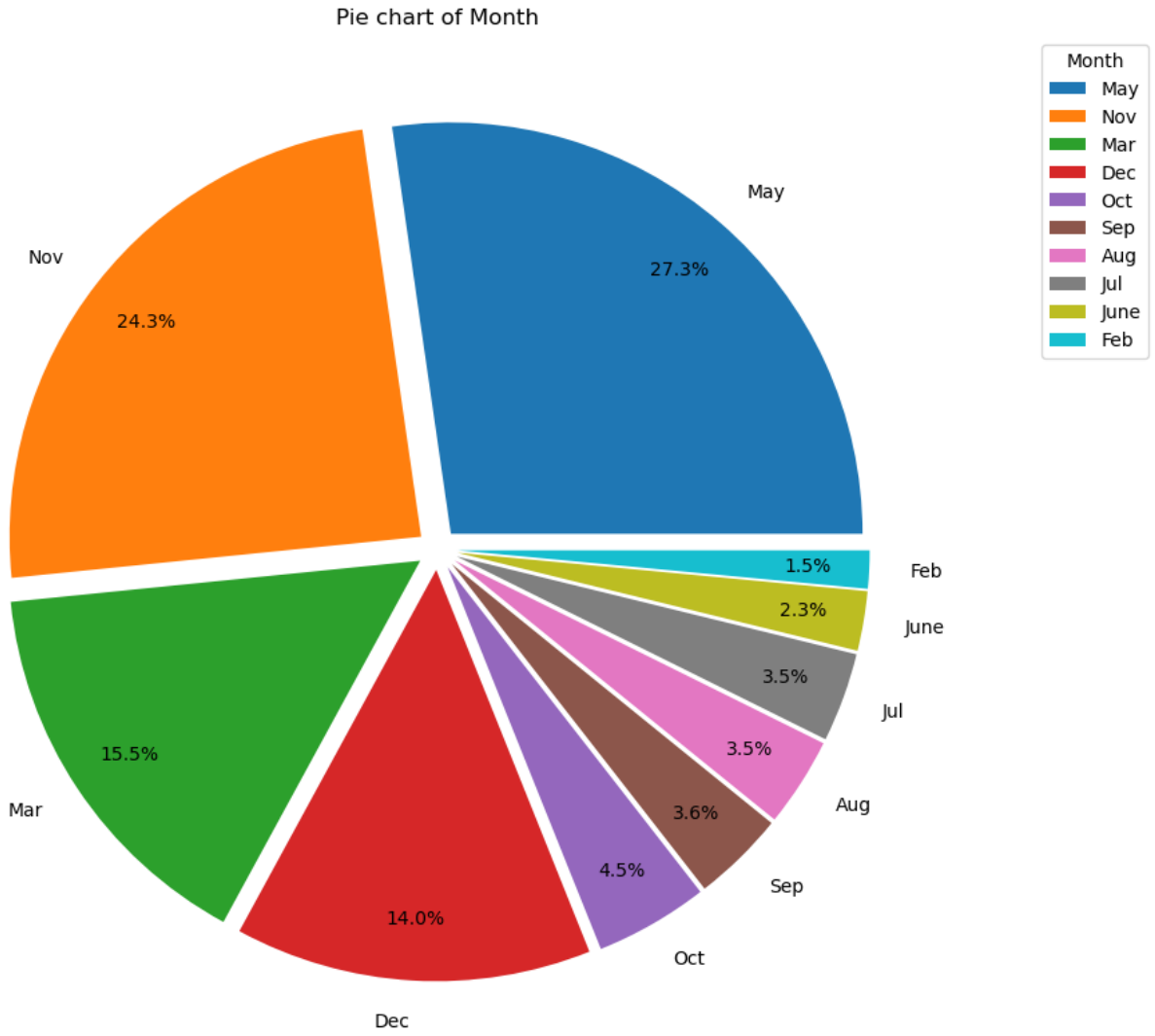
In [8]:

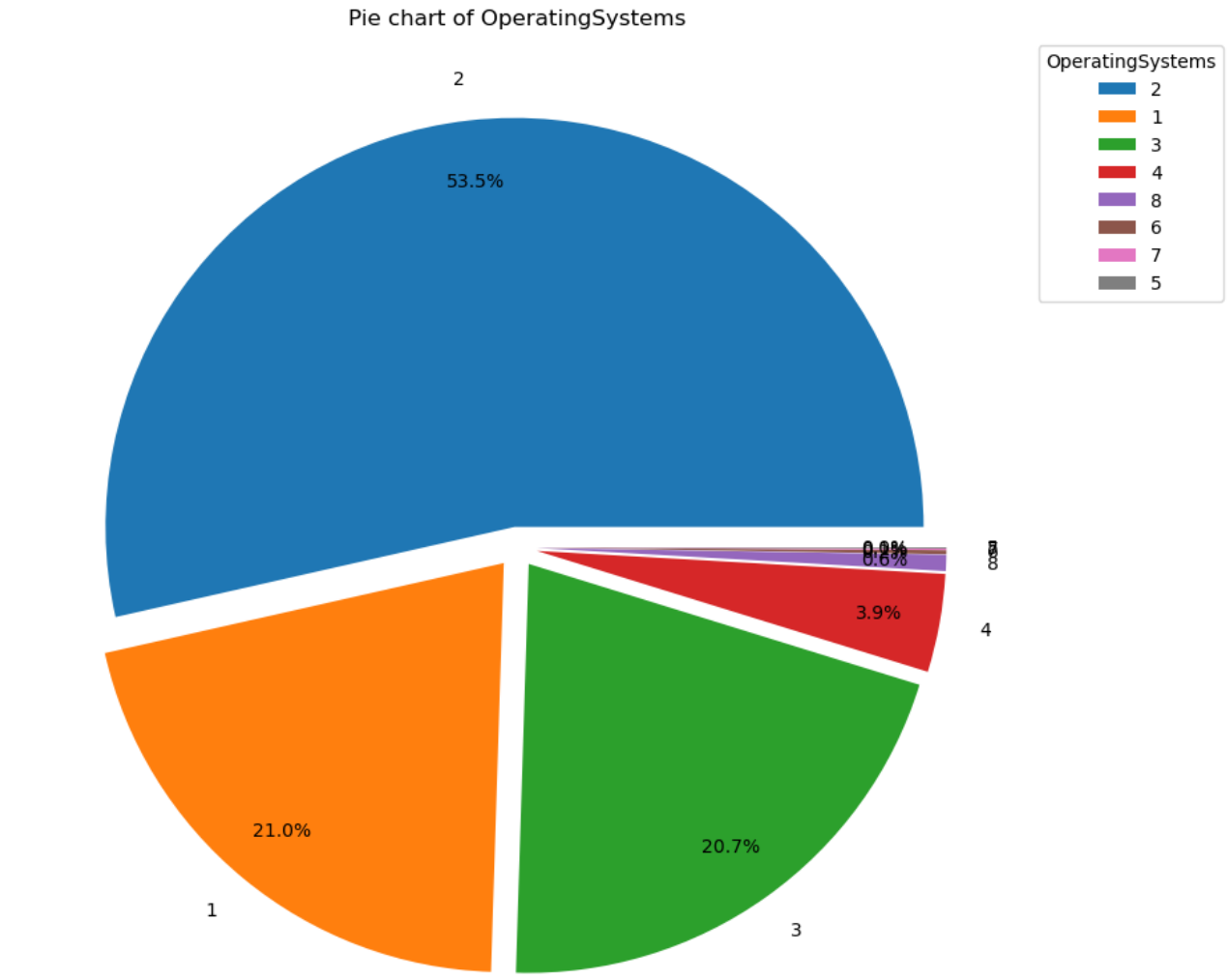
```
plt.figure(figsize=(12,8))
ax = sns.countplot(x='Revenue', hue='Month', data=df, palette='tab10')
plt.title('Count of Months by Revenue')
plt.xticks(rotation=90)

# Displaying the frequency above each bar
for p in ax.patches:
    ax.annotate(f'{p.get_height()}', (p.get_x()+p.get_width()/2., p.get_height()+0.5), ha='center', va='bottom'
plt.show()

```

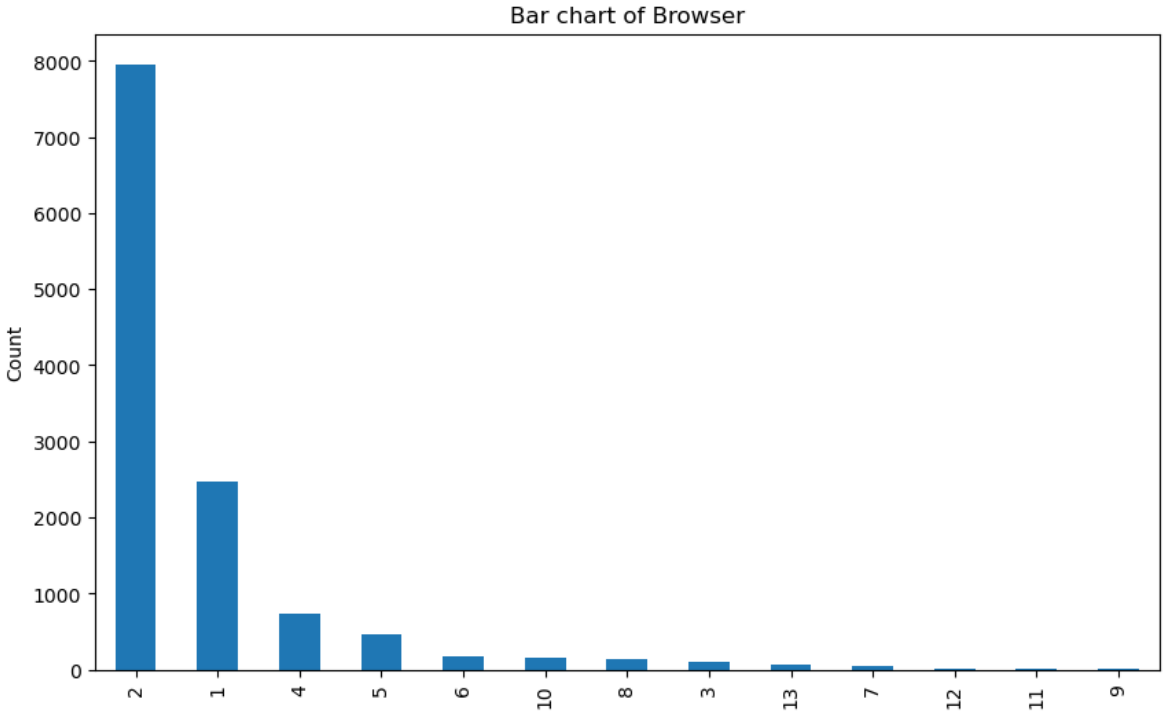


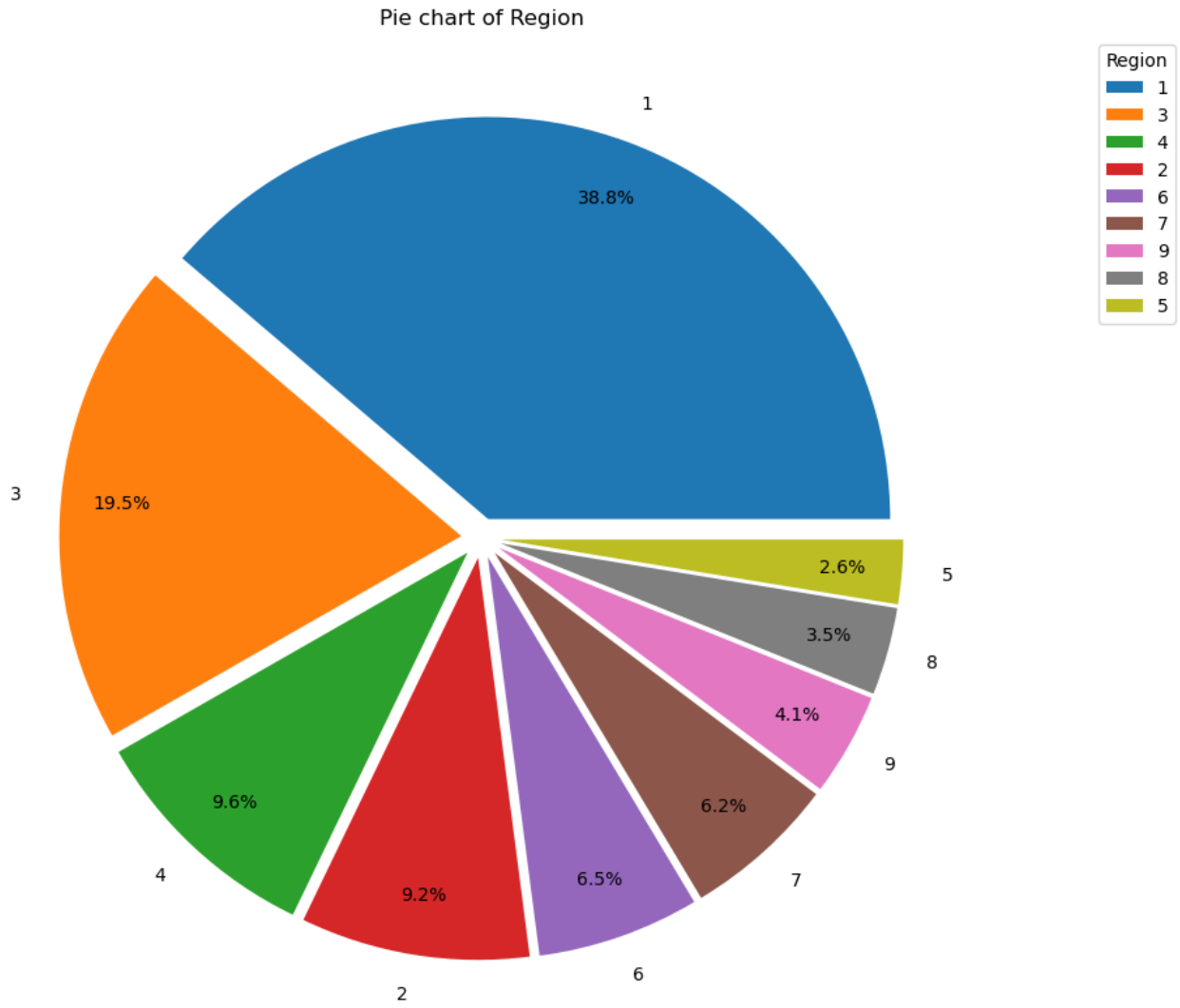
In [9]:

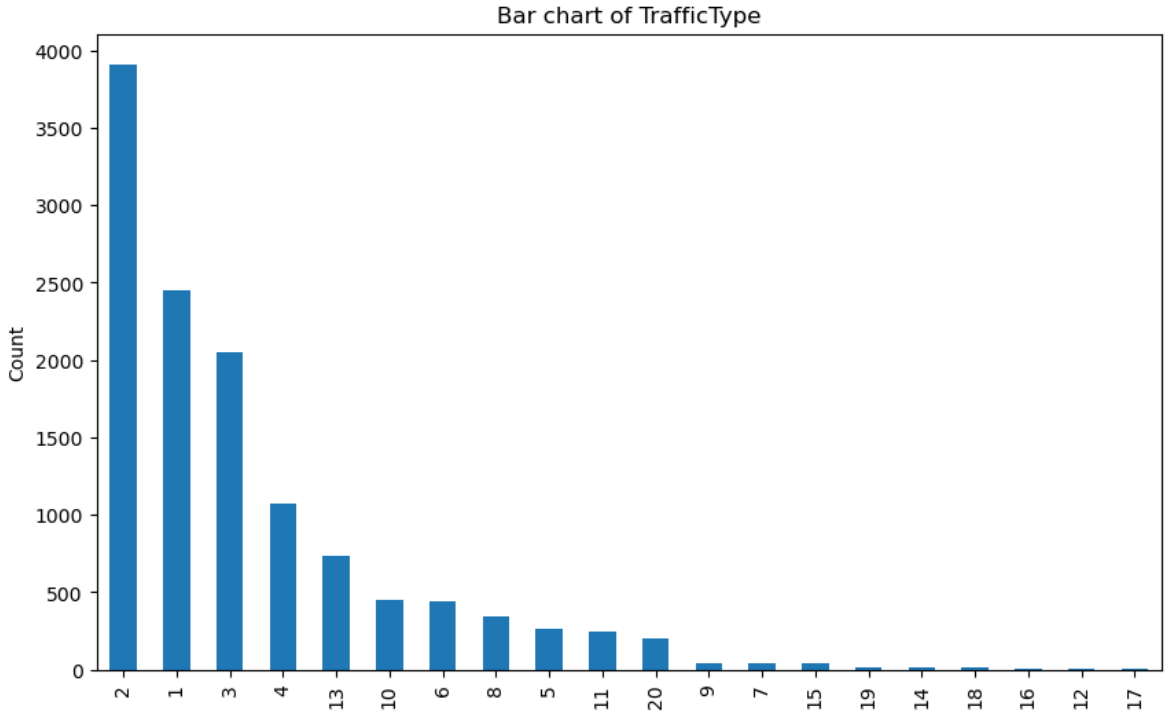
```
# For categorical columns
# if more than 10 columns, there will be a bar chart
cat_cols = ['Month', 'OperatingSystems', 'Browser', 'Region', 'TrafficType', 'VisitorType', 'Weekend', 'Revenue
for column in cat_cols:
    if len(df[column].value_counts()) > 10:
        plt.figure(figsize=(10,6))
        df[column].value_counts().plot(kind='bar')
        plt.title('Bar chart of '+ column)
        plt.ylabel('Count')
    else:
        plt.figure(figsize=(10,10))
        explode = [0.05]*len(df[column].value_counts())
        df[column].value_counts().plot(kind='pie', autopct='%1.1f%%', explode=explode, pctdistance=0.85)
        plt.title('Pie chart of '+ column)
        plt.ylabel('')
        plt.legend(title=column, loc='upper right', bbox_to_anchor=(1.2, 1))
    plt.show()

```

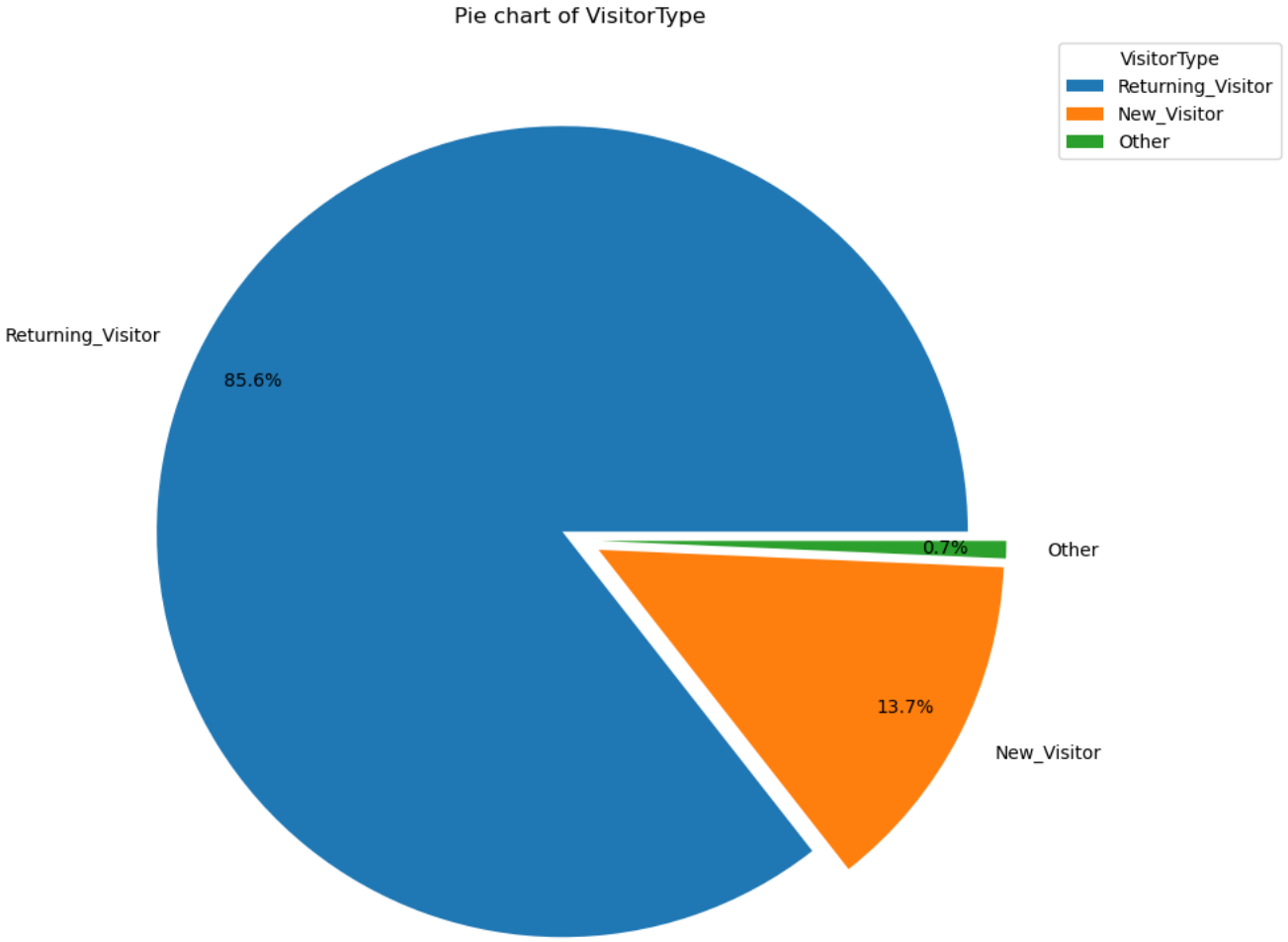


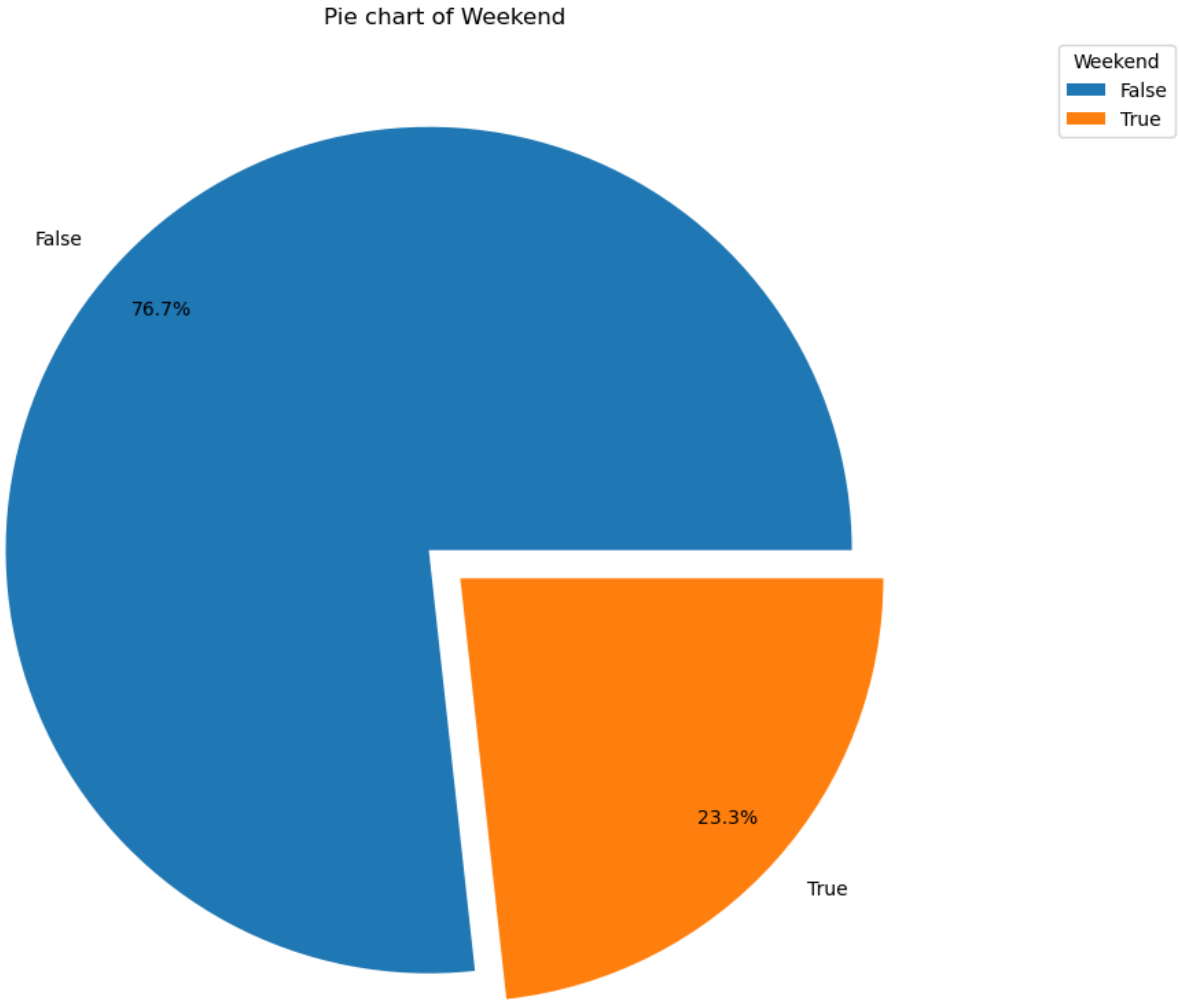

Pie chart of OperatingSystems
OperatingSystems
2
1
3
4
8
6
7
5
2
53.5%
21.0%
20.7%
3.9%
1
3
4

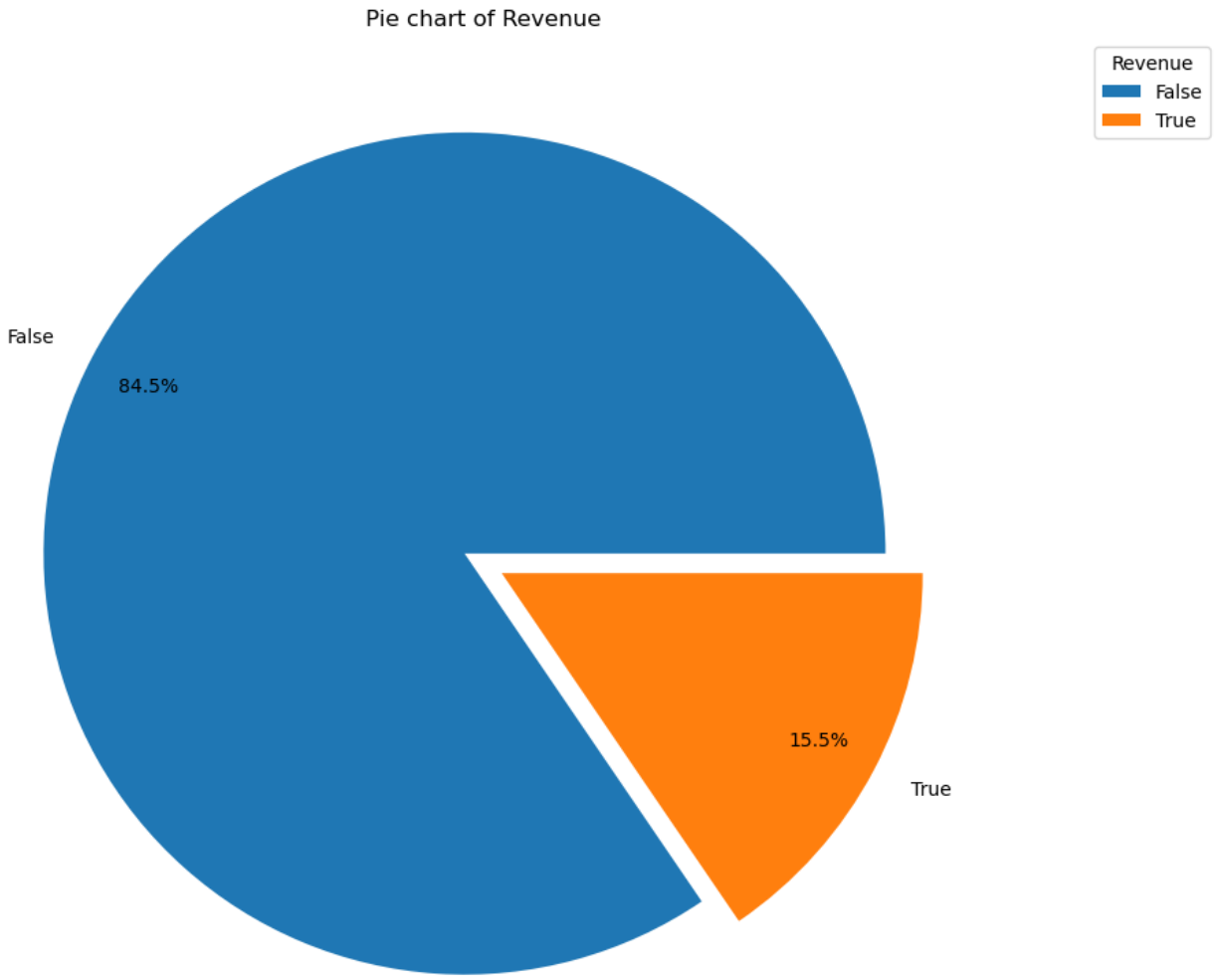

Bar chart of Browser
Count
8000
7000
6000
5000
4000
3000
2000
1000
0
2
1
4
5
6
10
8
3
13
7
12
11
9


Pie chart of Region
Region
1
3
4
2
6
7
9
8
5
38.8%
19.5%
9.6%
9.2%
6.5%
6.2%
4.1%
3.5%
2.6%


Bar chart of TrafficType
Count
4000
3500
3000
2500
2000
1500
1000
500
0
2
1
3
4
13
10
6
8
5
11
20
9
7
15
19
14
18
16
12
17


Pie chart of VisitorType
VisitorType
Returning_Visitor
New_Visitor
Other
Returning_Visitor
85.6%
0.7%
Other
13.7%
New_Visitor

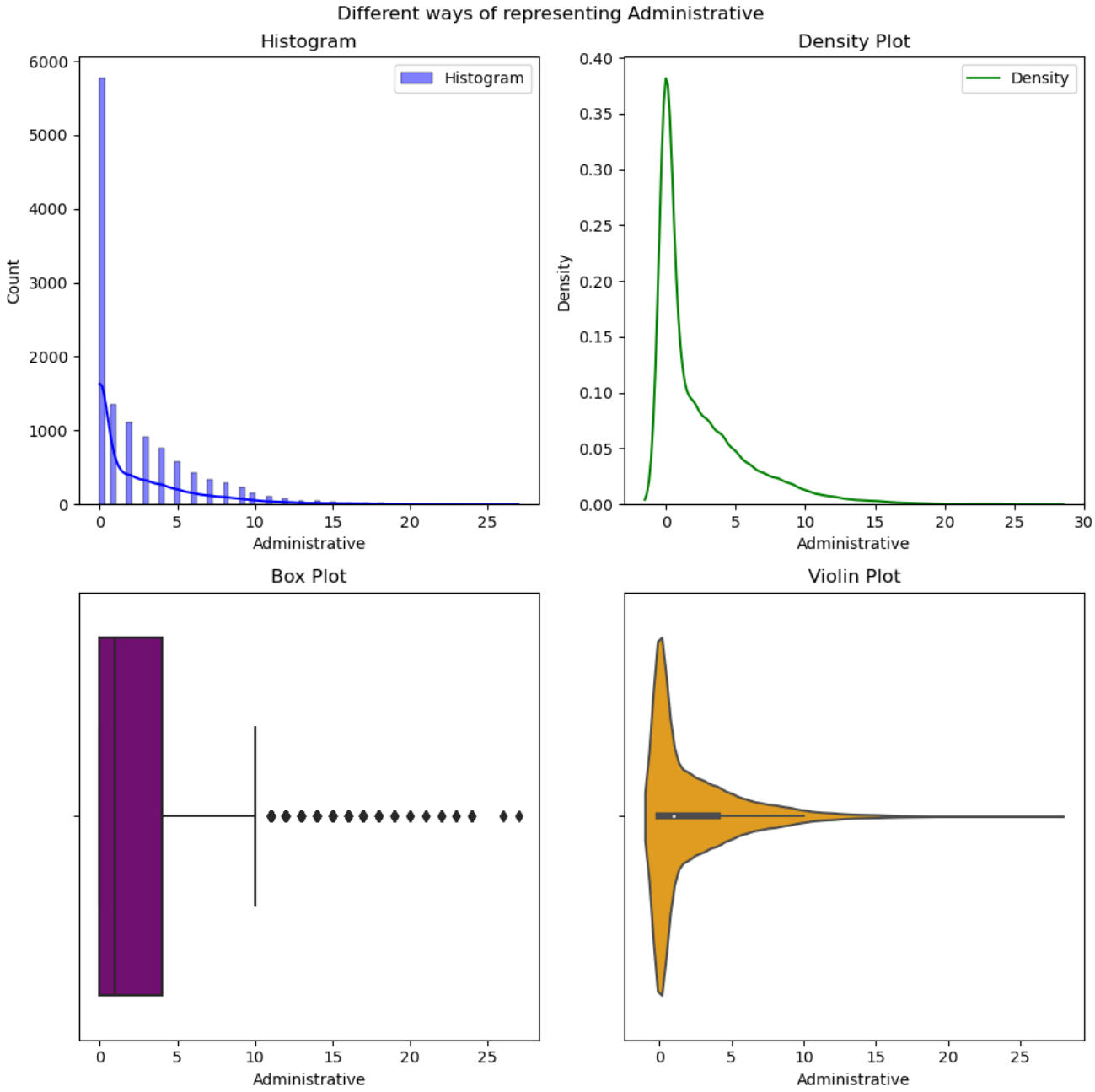

Pie chart of Weekend
Weekend
False
True
False
76.7%
23.3%
True

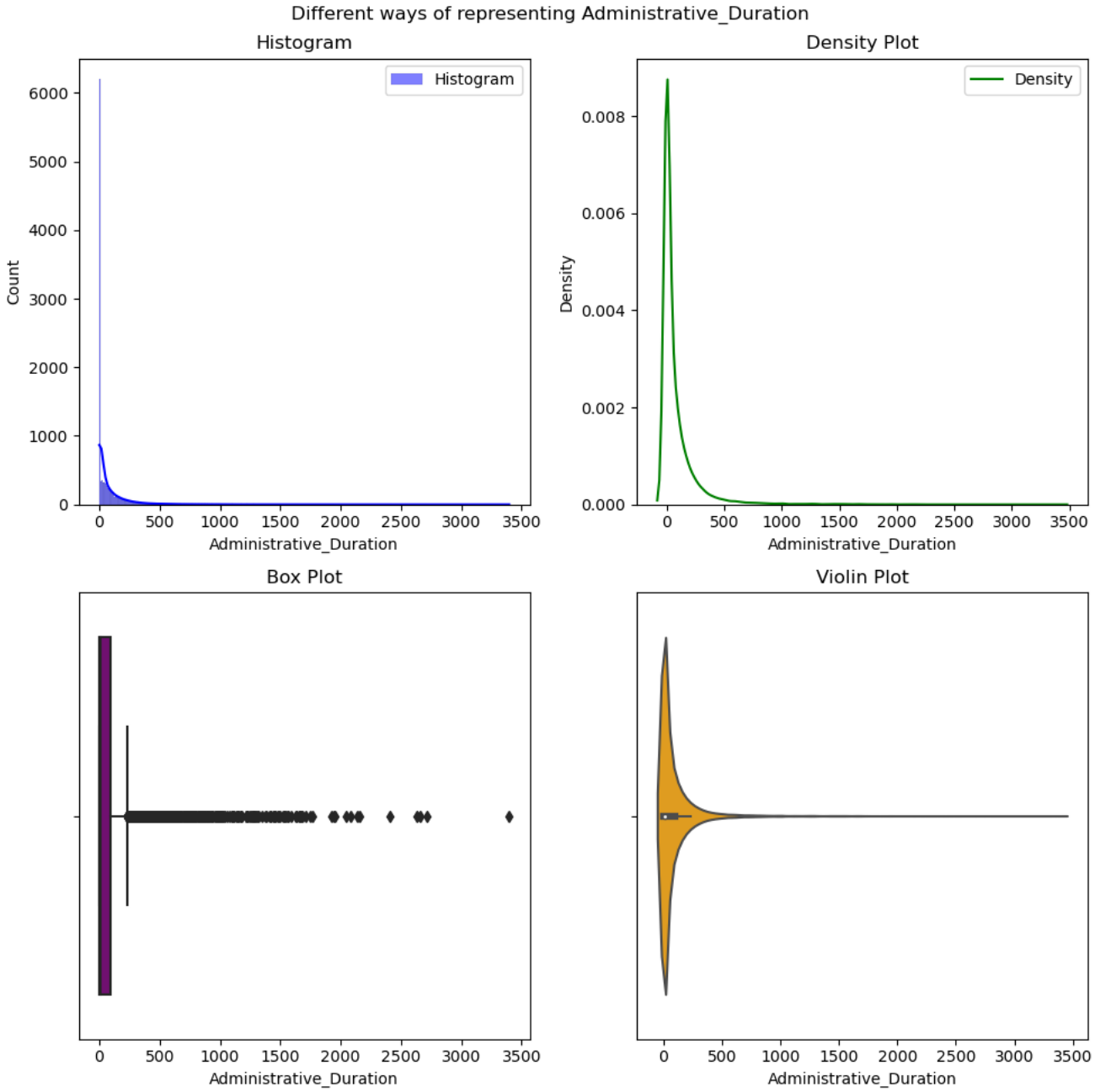


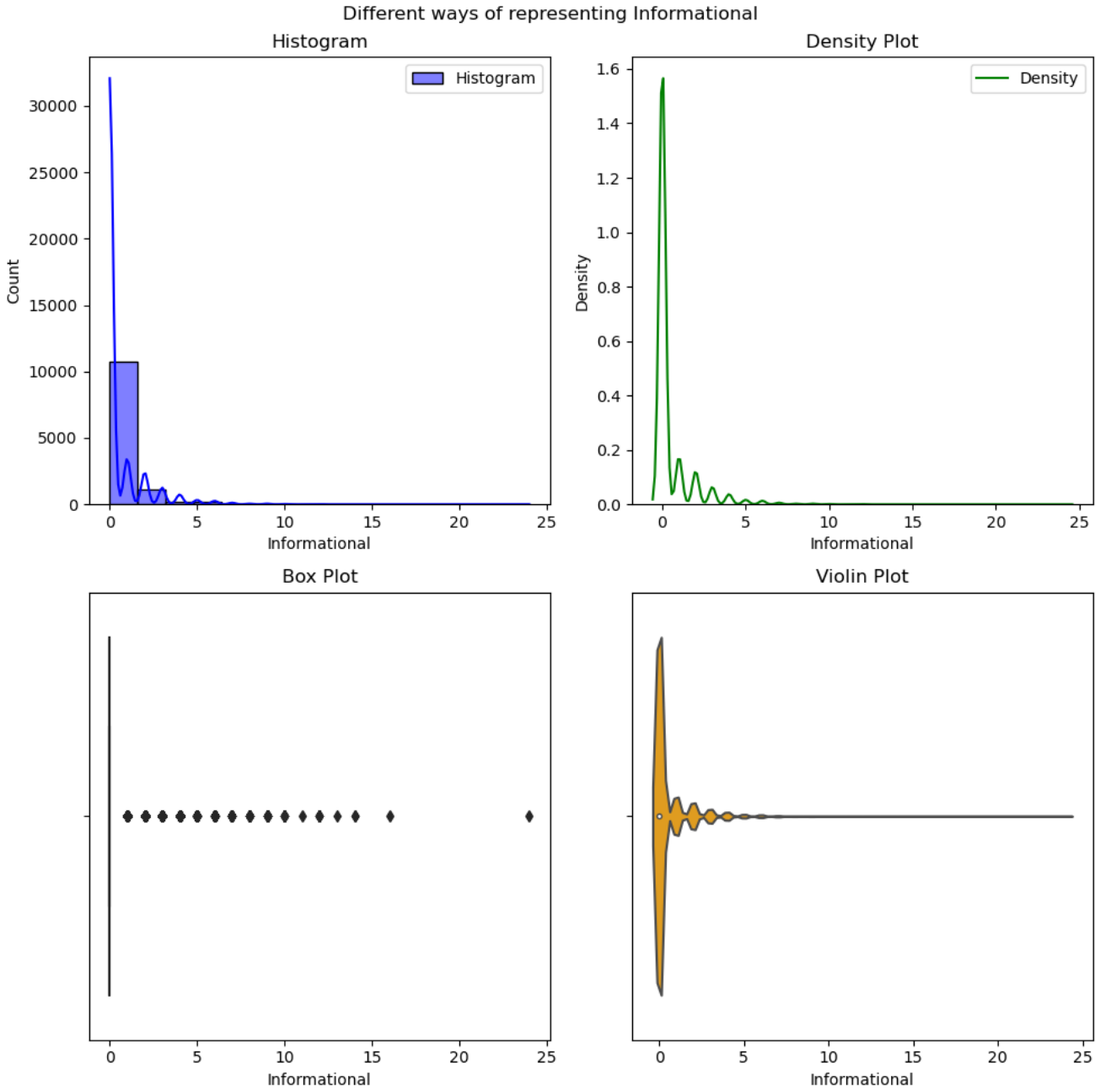
Task 1.1 Checking for Outliers in the Numerical Variables

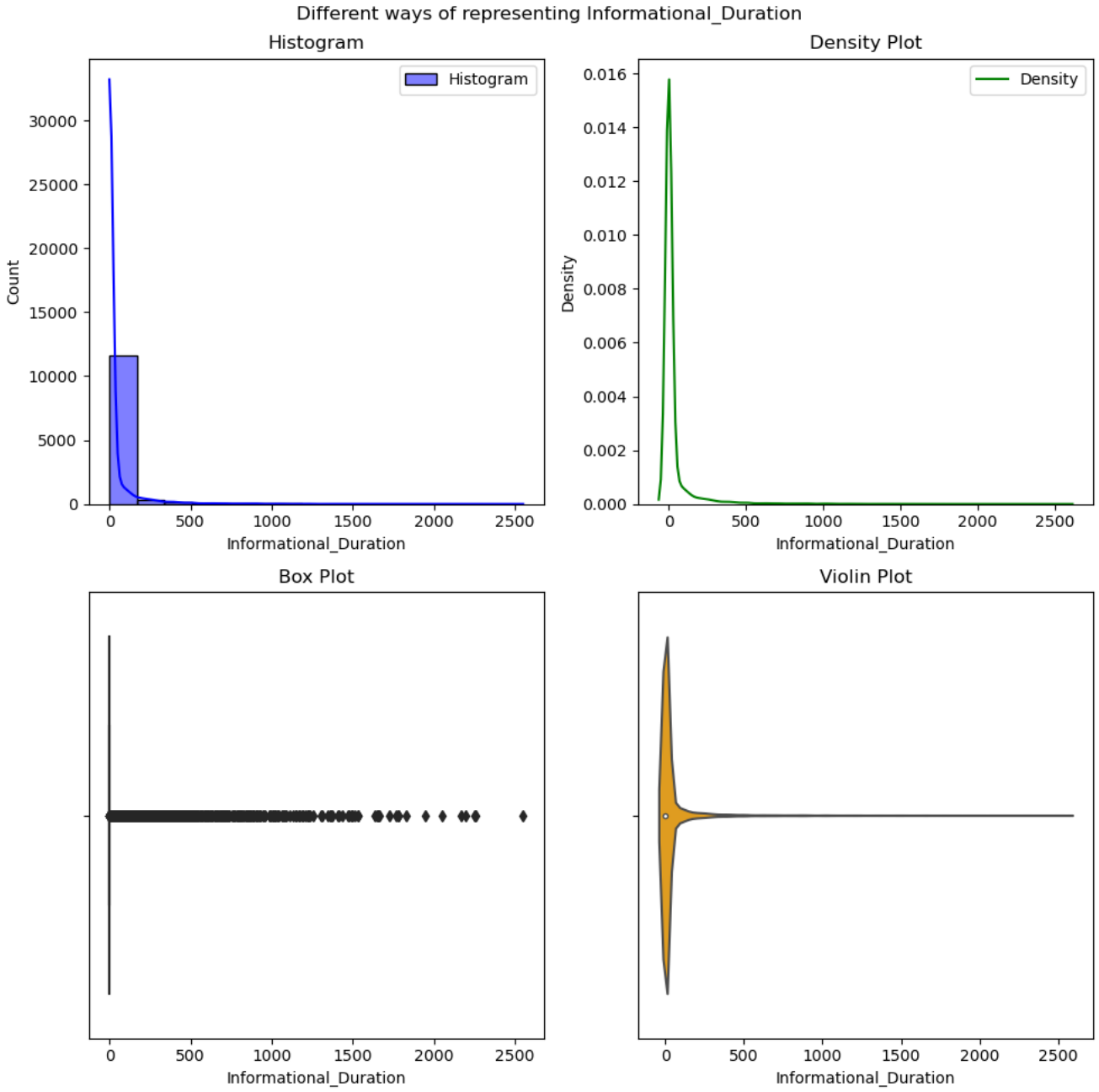
In [10]:

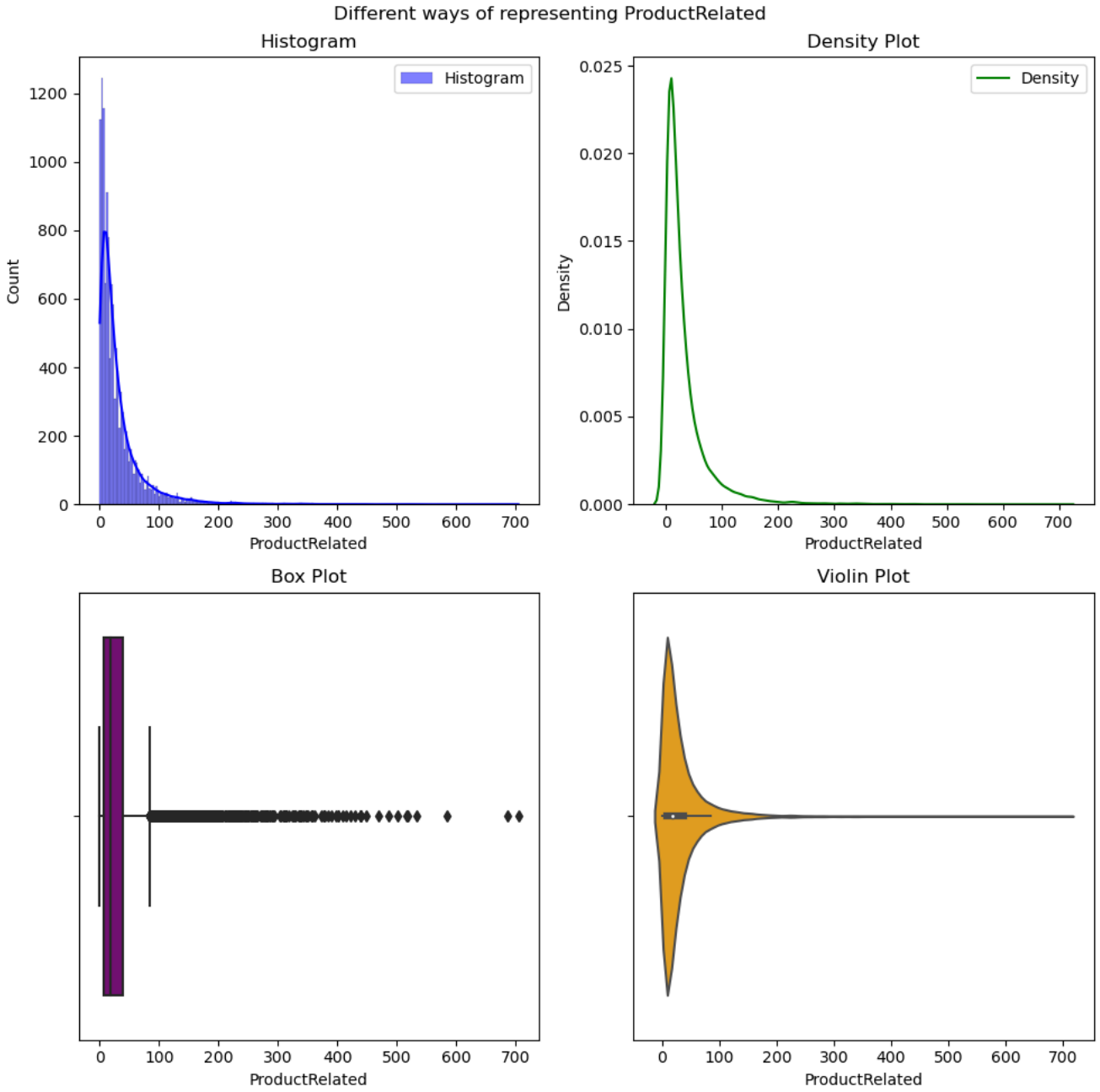
```
import seaborn as sns
import matplotlib.pyplot as plt

# This is for checking and viewing the spread of outliers across the Numerical Variables
num_cols = ['Administrative', 'Administrative_Duration', 'Informational', 'Informational_Duration', 'ProductRela

for column in num_cols:
    fig, axs = plt.subplots(2, 2, figsize=(10, 10))
    fig.suptitle('Different ways of representing ' + column)

    # Code for Histogram
    sns.histplot(df[column], ax=axs[0, 0], kde=True, color='blue', label='Histogram')
    axs[0, 0].set_title('Histogram')
    axs[0, 0].legend()

    # Code for Density plot
    sns.kdeplot(df[column], ax=axs[0, 1], color='green', label='Density')
    axs[0, 1].set_title('Density Plot')
    axs[0, 1].legend()

    # Code for Box plot
    sns.boxplot(x=df[column], ax=axs[1, 0], color='purple')
    axs[1, 0].set_title('Box Plot')

    # Code for Violin plot
    sns.violinplot(x=df[column], ax=axs[1, 1], color='orange')
    axs[1, 1].set_title('Violin Plot')

    plt.tight_layout()
    plt.show()

```

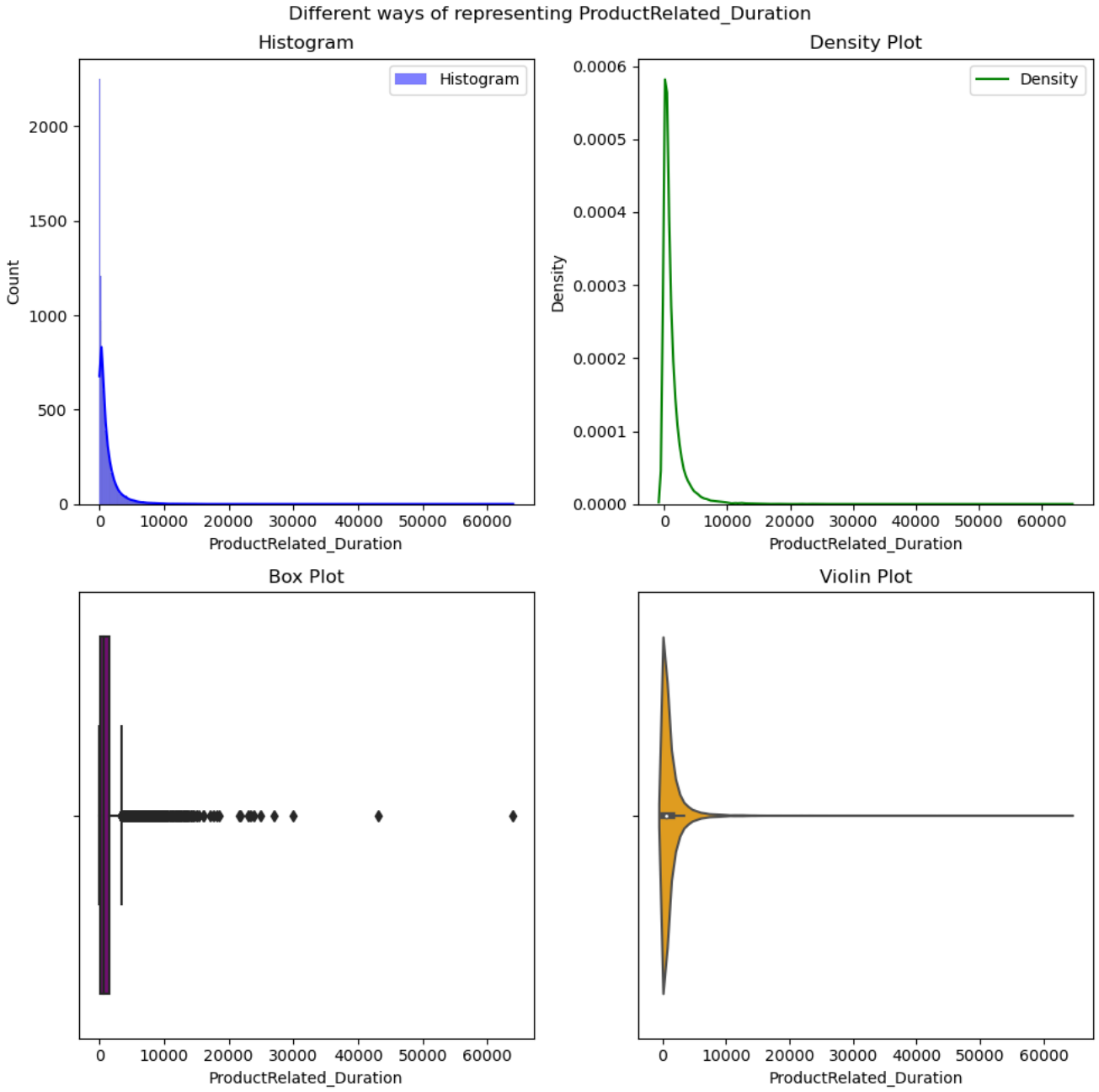

Different ways of representing Administrative
Histogram
Histogram
Count
Administrative
Density Plot
Density
Density
Administrative
Box Plot
Administrative
Violin Plot
Administrative

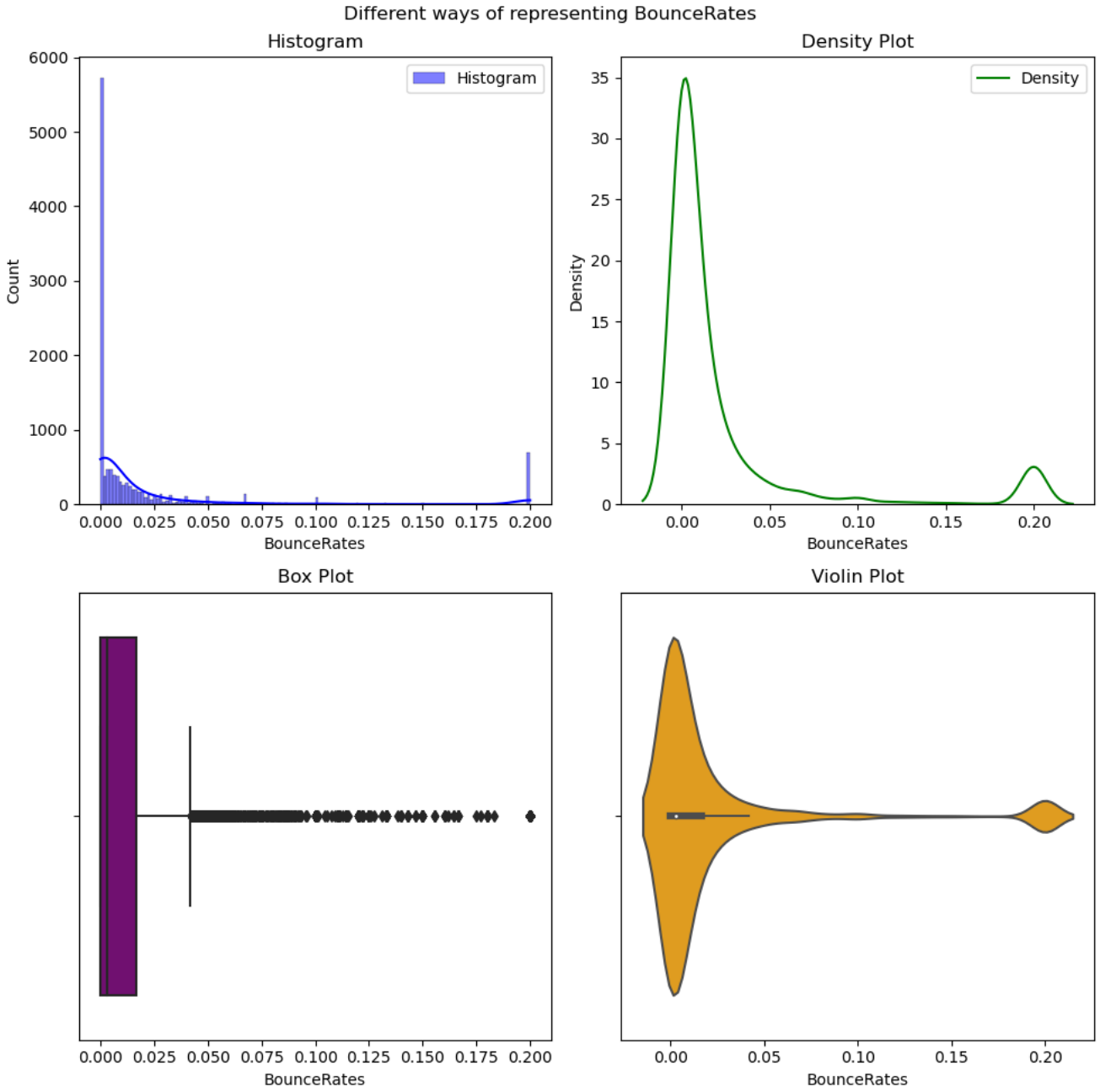

Different ways of representing Administrative_Duration
Histogram
Histogram
Count
Administrative_Duration
Density Plot
Density
Density
Administrative_Duration
Box Plot
Administrative_Duration
Violin Plot
Administrative_Duration

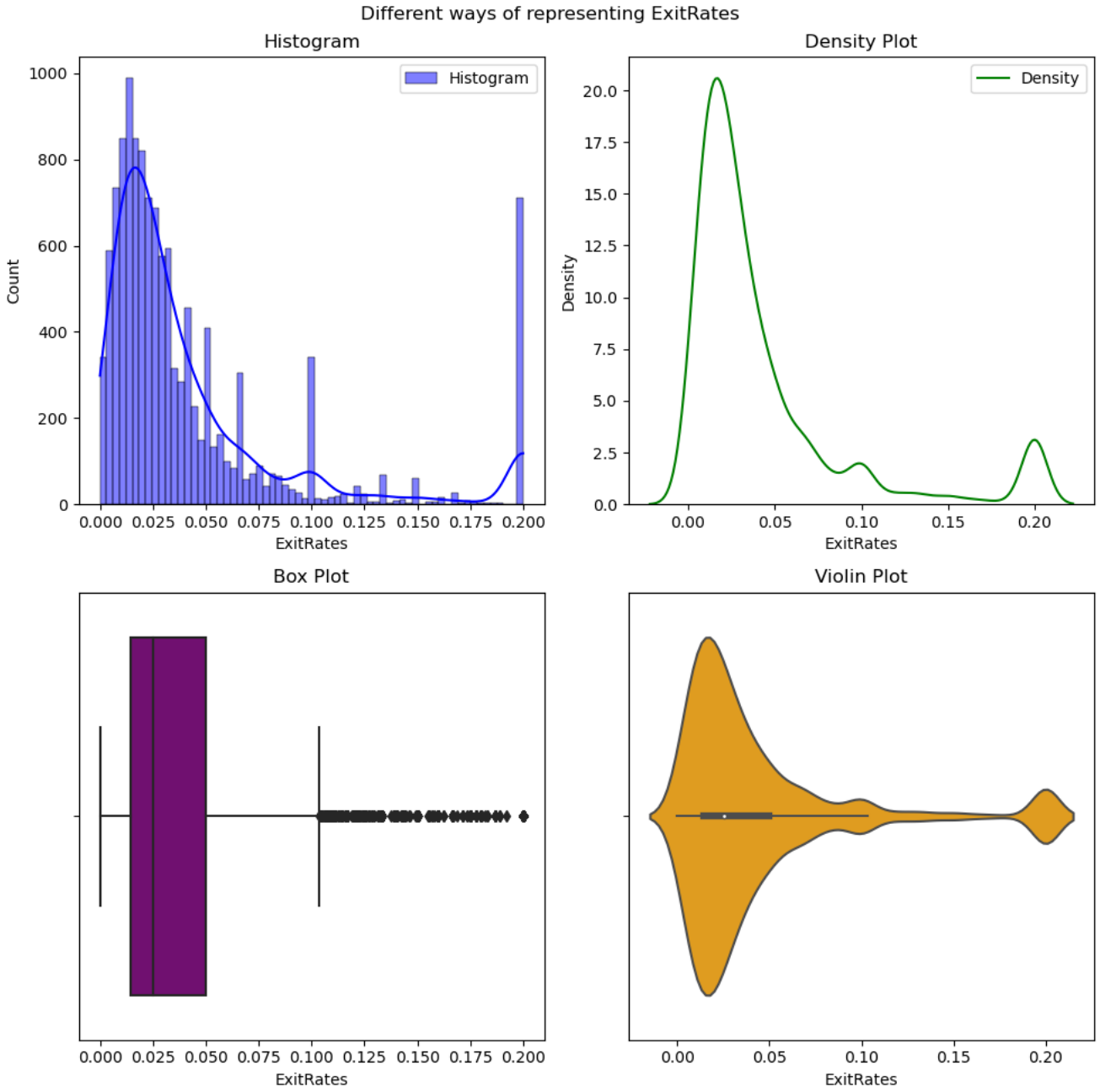

Different ways of representing Informational
Histogram
Histogram
Count
Informational
Density Plot
Density
Density
Informational
Box Plot
Informational
Violin Plot
Informational

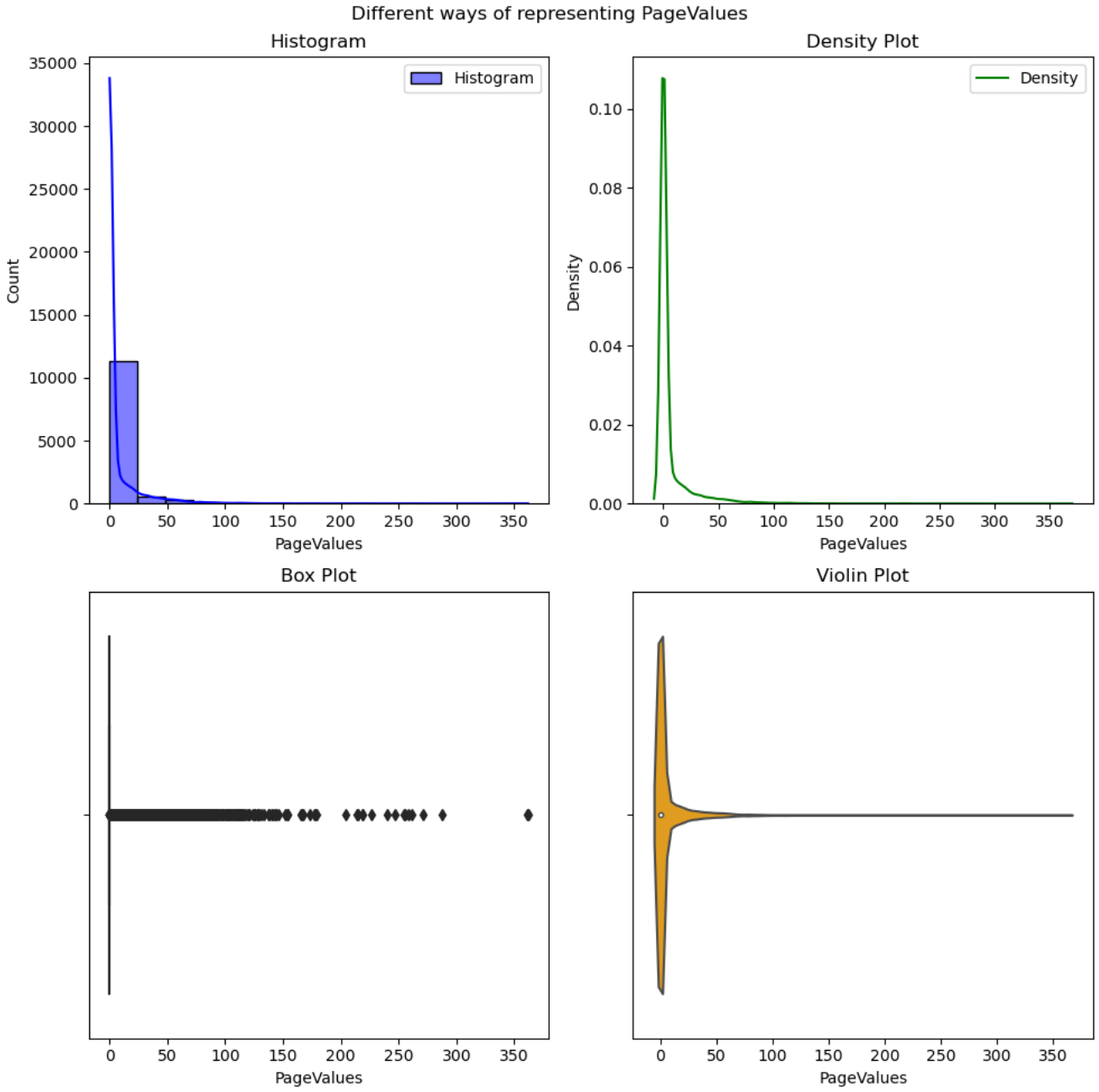

Different ways of representing Informational_Duration
Histogram
Histogram
Count
Informational_Duration
Density Plot
Density
Density
Informational_Duration
Box Plot
Informational_Duration
Violin Plot
Informational_Duration

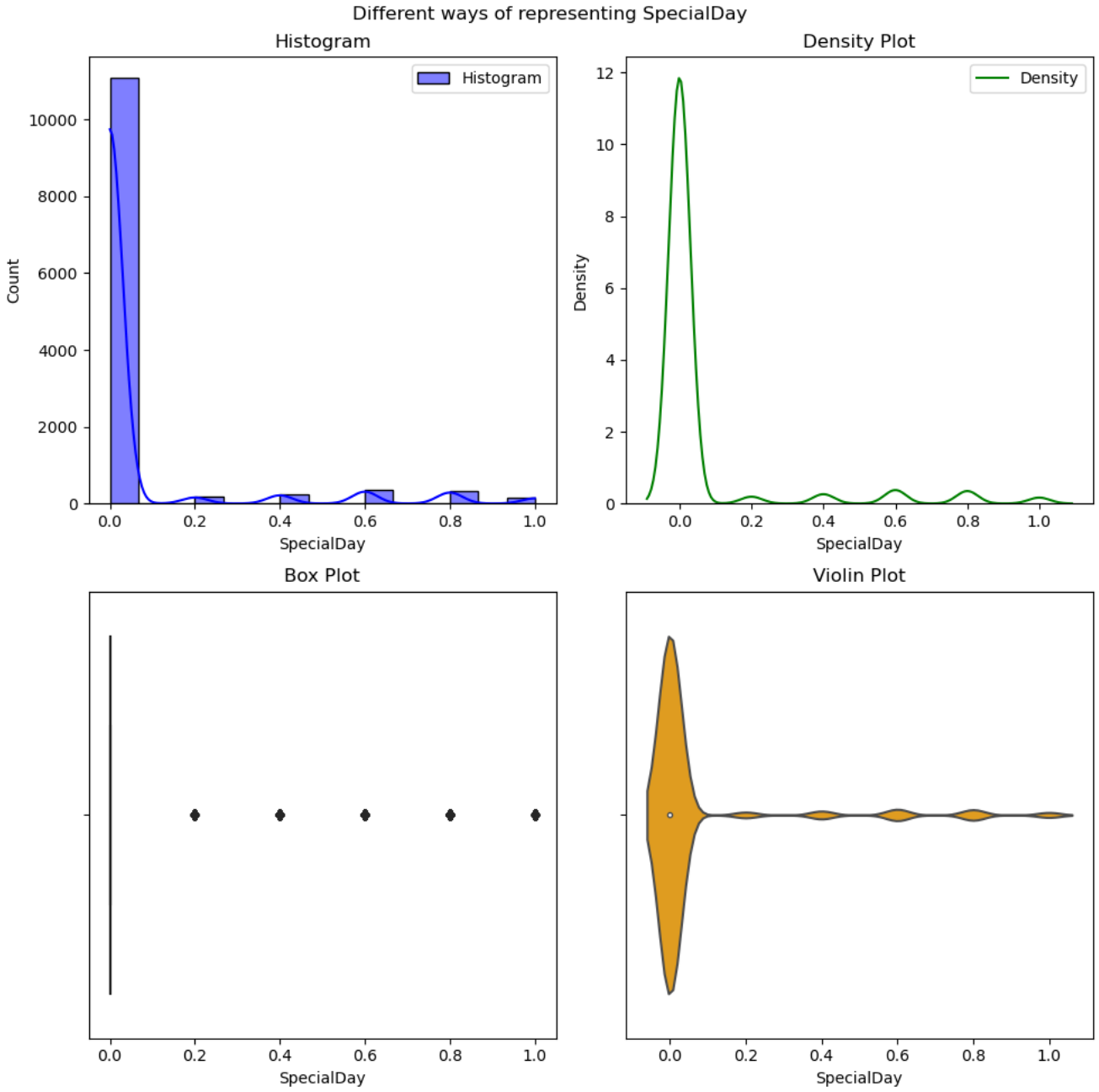

Different ways of representing ProductRelated
Histogram
Histogram
Count
ProductRelated
Density Plot
Density
Density
ProductRelated
Box Plot
ProductRelated
Violin Plot
ProductRelated


Different ways of representing ProductRelated_Duration
Histogram
Histogram
Count
ProductRelated_Duration
Density Plot
Density
Density
ProductRelated_Duration
Box Plot
ProductRelated_Duration
Violin Plot
ProductRelated_Duration


Different ways of representing BounceRates
Histogram
Histogram
Count
BounceRates
Density Plot
Density
Density
BounceRates
Box Plot
BounceRates
Violin Plot
BounceRates


Different ways of representing ExitRates
Histogram
Histogram
Count
ExitRates
Density Plot
Density
Density
ExitRates
Box Plot
ExitRates
Violin Plot
ExitRates

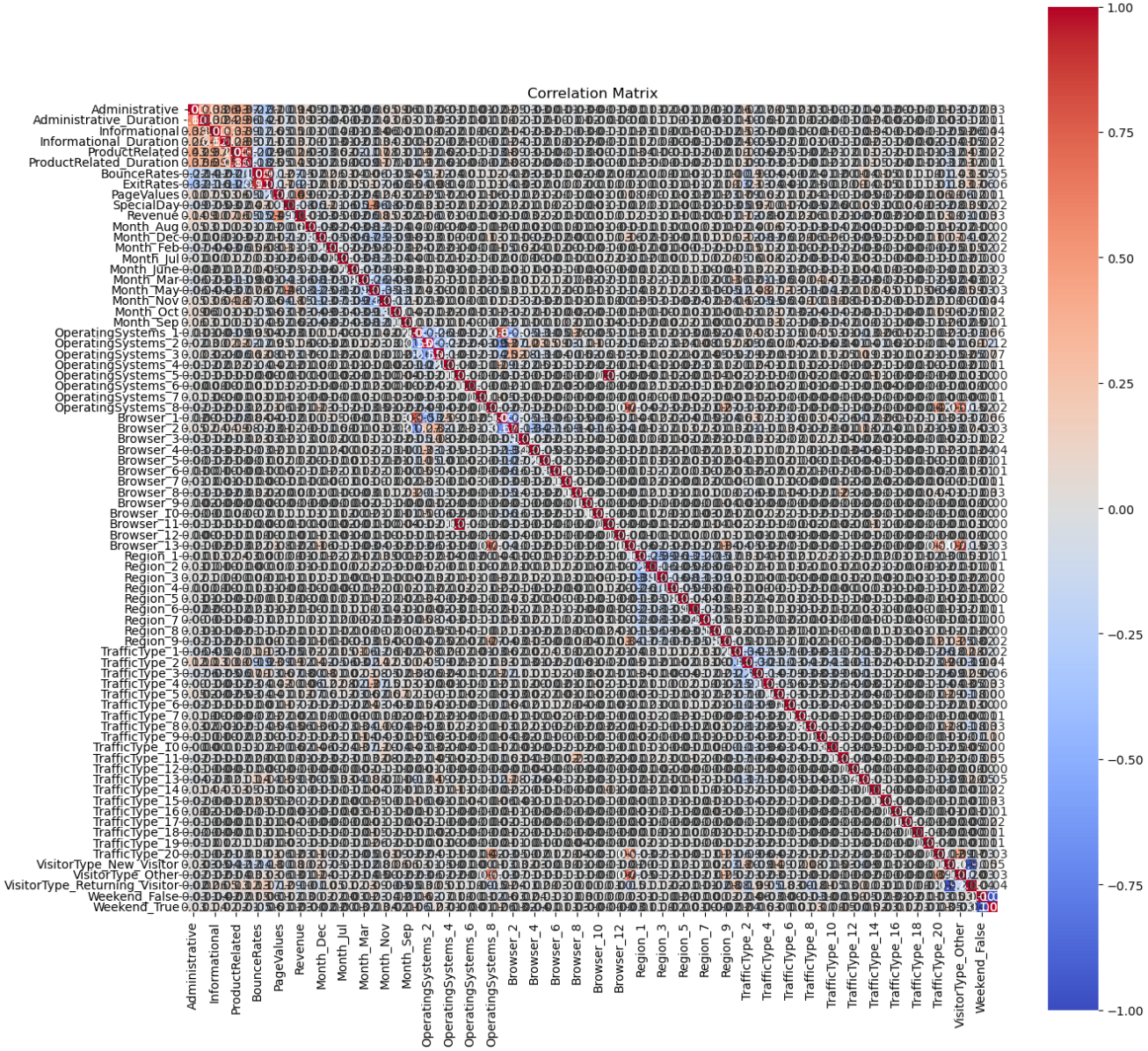

Different ways of representing PageValues
Histogram
Histogram
Count
PageValues
Density Plot
Density
Density
PageValues
Box Plot
PageValues
Violin Plot
PageValues



Task 3: Data Modelling - Correlation Matrix

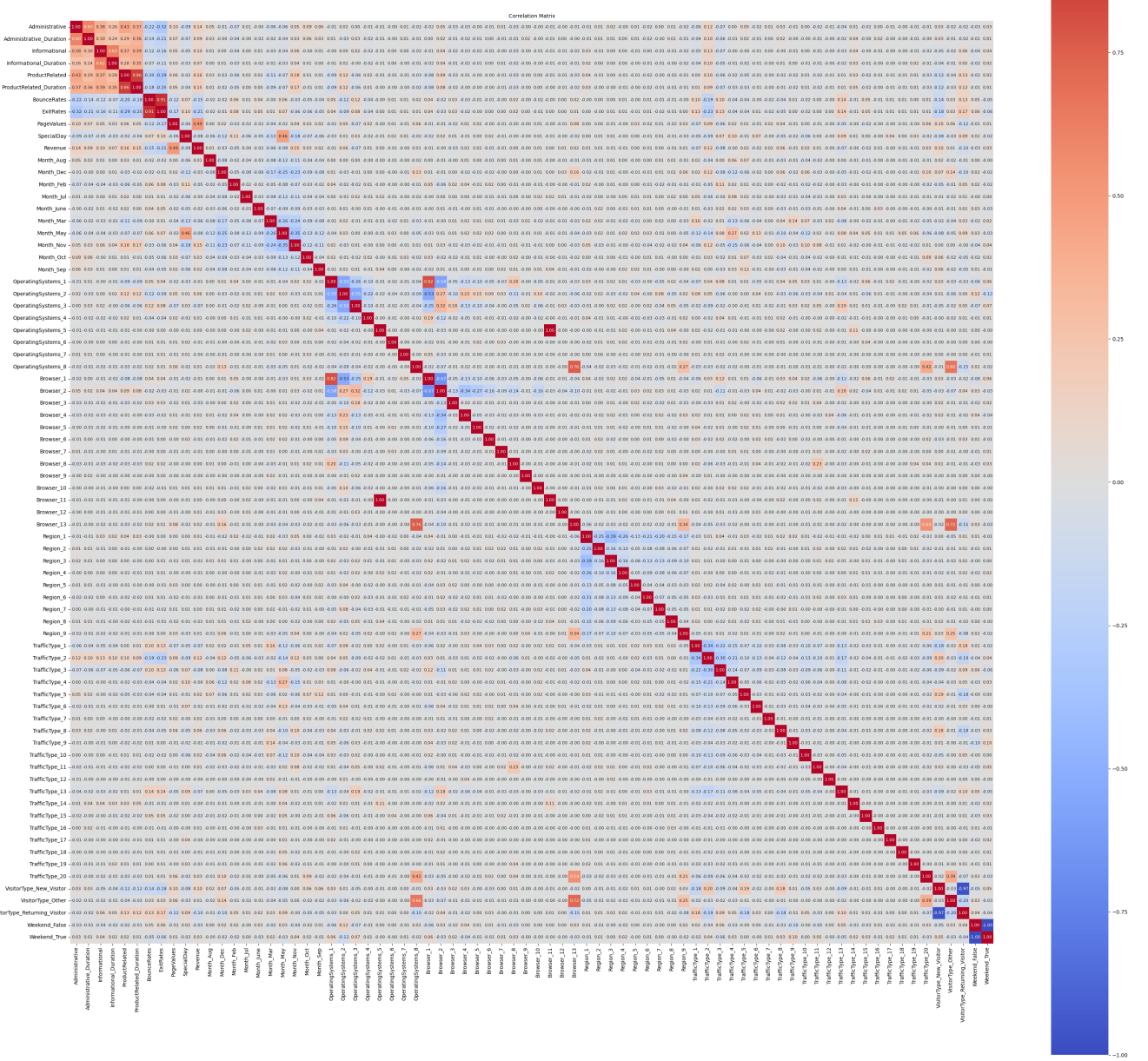
In [11]:

```
# Creating dummy variables for the categorical columns
df_encoded = pd.get_dummies(df, columns=['Month', 'OperatingSystems', 'Browser', 'Region', 'TrafficType', 'Visi

# Converting 'Revenue' to integer type (False to 0, True to 1)
df_encoded['Revenue'] = df_encoded['Revenue'].astype(int)

# Now I am taking a look at the correlation matrix
corr_matrix = df_encoded.corr()

# Now, I am plotting a heatmap of the correlation matrix
plt.figure(figsize=(15, 15))
sns.heatmap(corr_matrix, annot=True, fmt=".2f", cmap='coolwarm', square=True)
plt.title('Correlation Matrix')
plt.show()

```



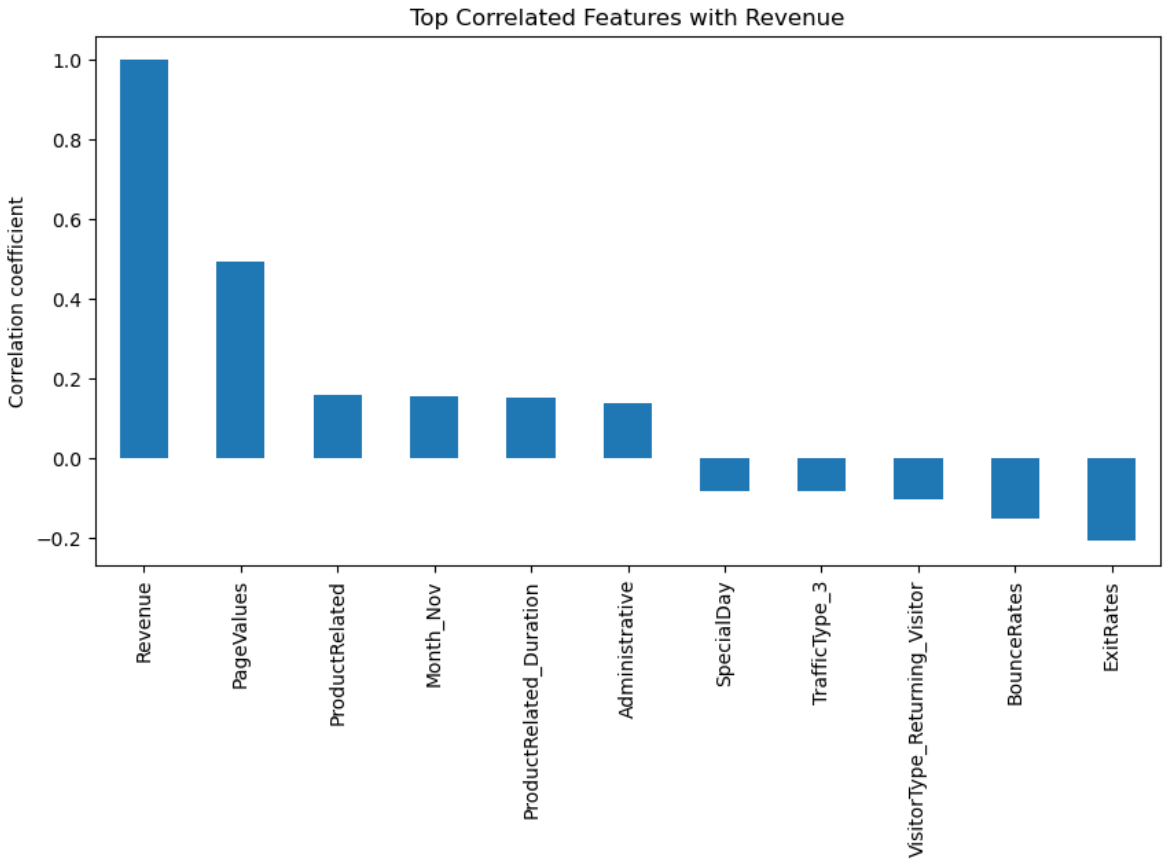
In [12]:

```
#larger plot
plt.figure(figsize=(40, 40))
sns.heatmap(corr_matrix, annot=True, fmt=".2f", cmap='coolwarm', square=True, annot_kws={"size": 8})
plt.title('Correlation Matrix', fontsize=10)
plt.show()
```



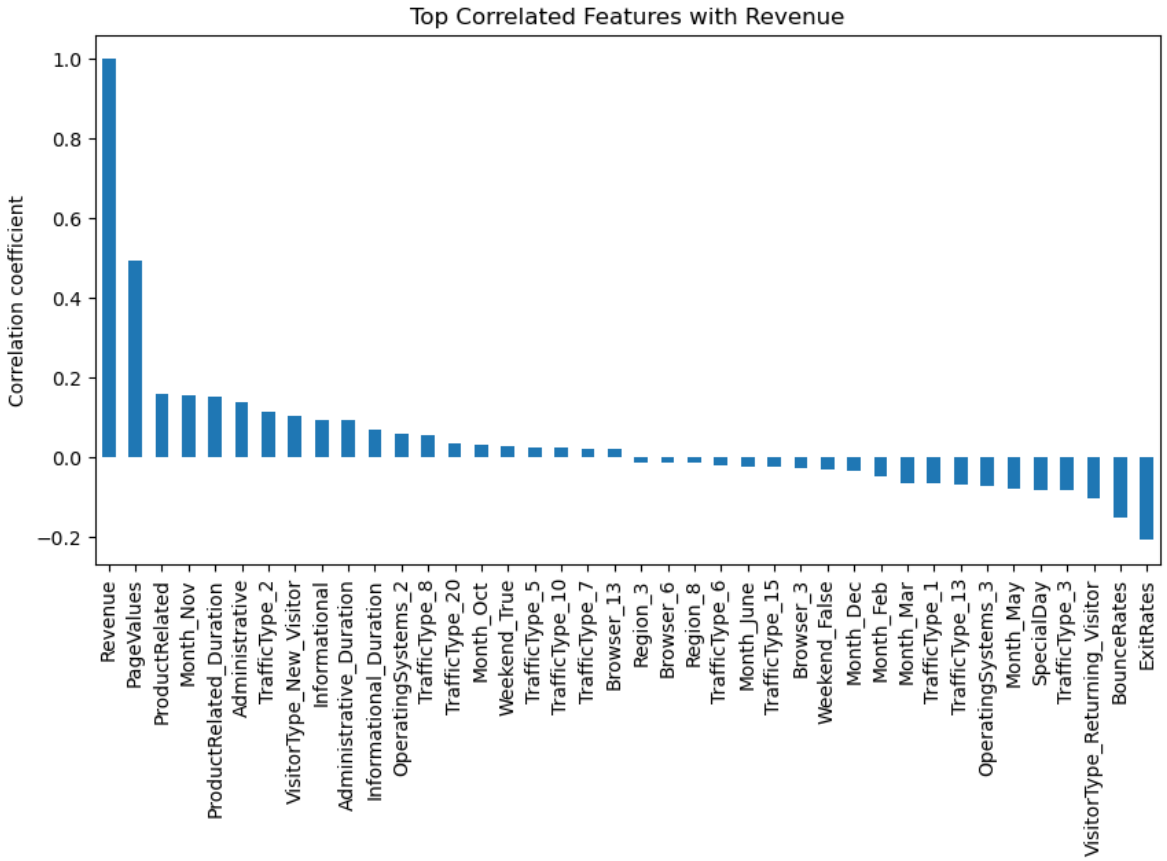
In [13]:

```
# Compute the correlation with 'Revenue' for all variables
correlation_with_revenue = df_encoded.corr()['Revenue'].sort_values(ascending=False)

# Select top 10 most correlated features
top_corr_features = pd.concat([correlation_with_revenue.head(6), correlation_with_revenue.tail(5)])

plt.figure(figsize=(10,5))
top_corr_features.plot(kind='bar')
plt.title('Top Correlated Features with Revenue')
plt.ylabel('Correlation coefficient')
plt.show()

```



In [14]:

```
# Compute the correlation with 'Revenue' for all variables
correlation_with_revenue = df_encoded.corr()['Revenue'].sort_values(ascending=False)

top_corr_features = pd.concat([correlation_with_revenue.head(20), correlation_with_revenue.tail(20)])

plt.figure(figsize=(10,5))
top_corr_features.plot(kind='bar')
plt.title('Top Correlated Features with Revenue')
plt.ylabel('Correlation coefficient')
plt.show()
```



Task 3: Machine Learning (KNN) for prediction with ~ 10 features

In [15]:

```
correlation_with_revenue = df_encoded.corr()['Revenue'].sort_values(ascending=False)

top_corr_features = pd.concat([correlation_with_revenue.head(20), correlation_with_revenue.tail(20)])

print("Top Correlated Features with Revenue:")
print(top_corr_features)

```

```
Top Correlated Features with Revenue:
Revenue                          1.000000
PageValues                       0.492569
ProductRelated                   0.158538
Month_Nov                        0.154774
ProductRelated_Duration          0.152373
Administrative                   0.138917
TrafficType_2                    0.116347
VisitorType_New_Visitor          0.104136
Informational                    0.095200
Administrative_Duration          0.093587
Informational_Duration           0.070345
OperatingSystems_2               0.060040
TrafficType_8                    0.057167
TrafficType_20                   0.034540
Month_Oct                        0.032666
Weekend_True                     0.029295
TrafficType_5                    0.024609
TrafficType_10                   0.024354
TrafficType_7                    0.022913
Browser_13                       0.020969
Region_3                        -0.012937
Browser_6                       -0.013167
Region_8                        -0.013579
TrafficType_6                   -0.018905
Month_June                      -0.023112
TrafficType_15                  -0.023790
Browser_3                       -0.027451
Weekend_False                   -0.029295
Month_Dec                       -0.033112
Month_Feb                       -0.047114
Month_Mar                       -0.063941
TrafficType_1                   -0.065901
TrafficType_13                  -0.067310
OperatingSystems_3              -0.070472
Month_May                       -0.078320
SpecialDay                      -0.082305
TrafficType_3                   -0.082808
VisitorType_Returning_Visitor   -0.103843
BounceRates                     -0.150673
ExitRates                       -0.207071
Name: Revenue, dtype: float64
```

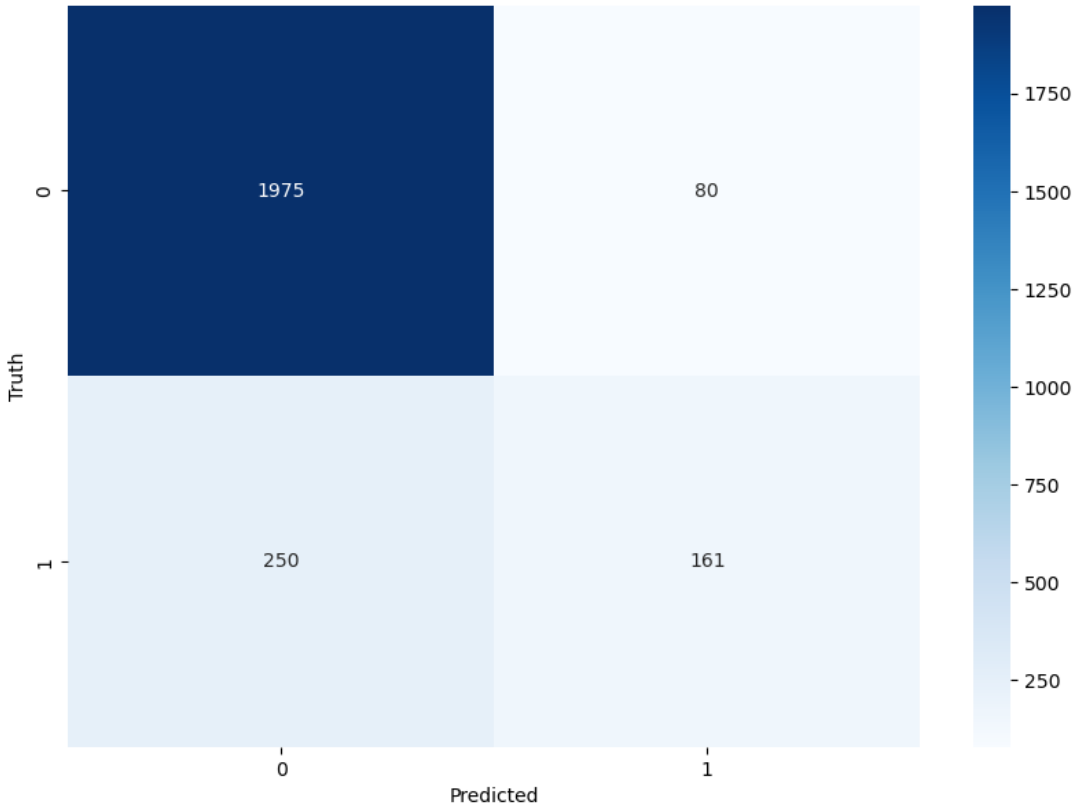
In [16]:

```
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, confusion_matrix

X = df_encoded[['PageValues', 'ProductRelated', 'ProductRelated_Duration', 'Month_Nov', 'Administrative', 'Traf

# Selecting the target
y = df_encoded['Revenue']

# Splitting the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Creating the classifier
knn = KNeighborsClassifier(n_neighbors=5)

# Training the classifier
knn.fit(X_train, y_train)

# Making predictions on the test set
y_pred = knn.predict(X_test)

# Printing the classification report
print(classification_report(y_test, y_pred))

# Printing the confusion matrix
print(confusion_matrix(y_test, y_pred))

```

```
              precision    recall  f1-score   support

           0       0.89      0.96      0.92      2055
           1       0.67      0.39      0.49       411

    accuracy                           0.87      2466
   macro avg       0.78      0.68      0.71      2466
weighted avg       0.85      0.87      0.85      2466

[[1975   80]
 [ 250  161]]
```

In [17]:

```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix

# Compute confusion matrix
cm = confusion_matrix(y_test, y_pred)

# Use seaborn to plot confusion matrix
plt.figure(figsize=(10,7))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Truth')
plt.show()

```



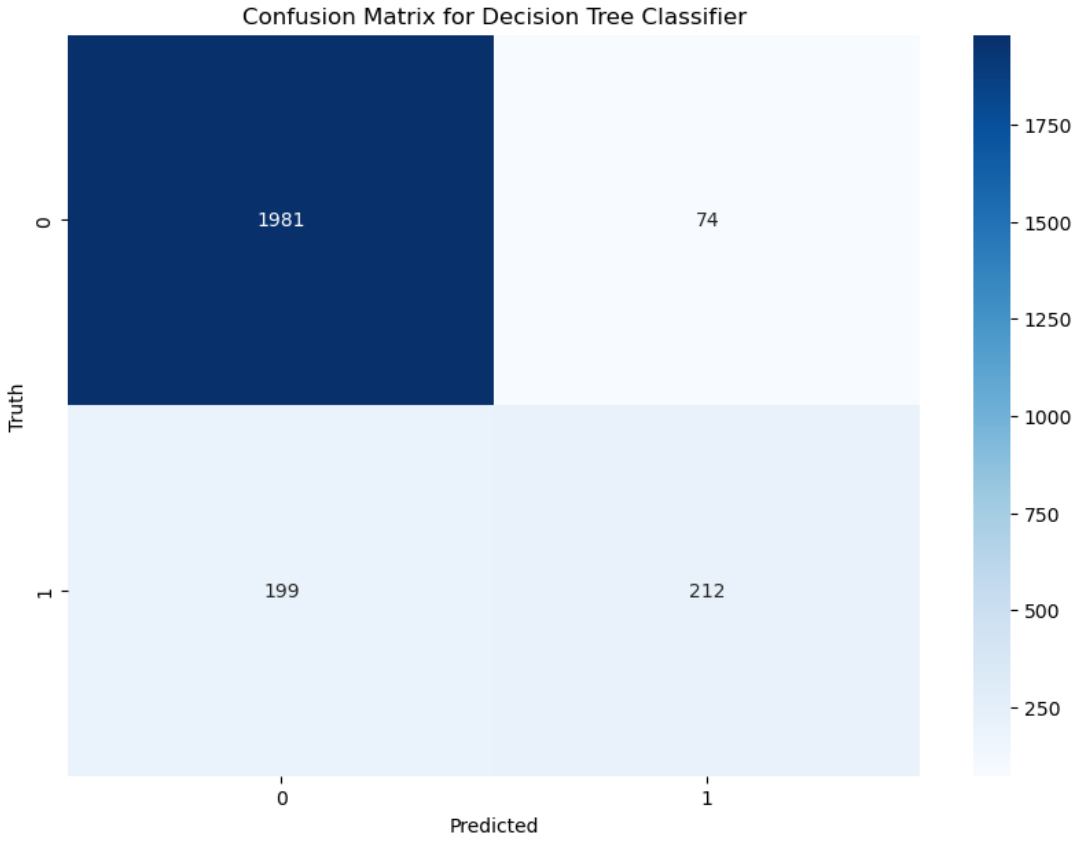
In [18]:

```
from sklearn.tree import DecisionTreeClassifier

dtree = DecisionTreeClassifier(max_depth=5)

dtree.fit(X_train, y_train)

y_pred = dtree.predict(X_test)

print(classification_report(y_test, y_pred))

print(confusion_matrix(y_test, y_pred))

```

```
              precision    recall  f1-score   support

           0       0.91      0.96      0.94      2055
           1       0.74      0.52      0.61       411

    accuracy                           0.89      2466
   macro avg       0.82      0.74      0.77      2466
weighted avg       0.88      0.89      0.88      2466

[[1981   74]
 [ 199  212]]
```

In [19]:

```
# Compute confusion matrix for Decision Tree
cm_dtree = confusion_matrix(y_test, y_pred)

# Use seaborn to plot confusion matrix
plt.figure(figsize=(10,7))
sns.heatmap(cm_dtree, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Truth')
plt.title('Confusion Matrix for Decision Tree Classifier')
plt.show()

```



In [20]:

```
from sklearn.model_selection import GridSearchCV

# Defining the parameter values that should be searched
param_grid = {
    'max_depth': [None, 5, 10, 15, 20],
    'min_samples_split': [2, 5, 10, 20],
    'min_samples_leaf': [1, 2, 5, 10],
    'max_features': [None, 'sqrt', 'log2']
}

# Instantiating the grid
grid = GridSearchCV(DecisionTreeClassifier(), param_grid, cv=5, scoring='accuracy')

# Fitting grid
grid.fit(X_train, y_train)

# Viewing complete results
print(grid.cv_results_)

# Viewing best params for the model found using grid search
print("Best parameters: ", grid.best_params_)

y_pred = grid.predict(X_test)

print(classification_report(y_test, y_pred))

print(confusion_matrix(y_test, y_pred))

```

{'mean_fit_time': array([0.02794075, 0.02385001, 0.02291331, 0.0246295 , 0.02391295,
       0.02245698, 0.02214851, 0.0216681 , 0.02182126, 0.02220182,
       0.02243443, 0.02116799, 0.02001858, 0.02001896, 0.01993976,
       0.01989298, 0.0096076 , 0.00978403, 0.00972009, 0.00853353,
       0.00927401, 0.00955482, 0.00880723, 0.00836358, 0.0085638 ,
       0.00840154, 0.00857277, 0.00828381, 0.00773826, 0.00818715,
       0.00800867, 0.00756598, 0.0103971 , 0.00927262, 0.00952992,
       0.00914326, 0.00955515, 0.00897784, 0.00907979, 0.00988388,
       0.01021104, 0.01292682, 0.01257453, 0.01152191, 0.0113564 ,
       0.01093297, 0.01022363, 0.01271939, 0.01718564, 0.01541276,
       0.01702685, 0.01691656, 0.01683893, 0.0169528 , 0.01705322,
       0.01690068, 0.01731944, 0.01727004, 0.01797333, 0.01715264,
       0.01677222, 0.01694126, 0.01638637, 0.01732955, 0.00784793,
       0.00786796, 0.00736098, 0.00745196, 0.00809102, 0.00736079,
       0.00775962, 0.00691786, 0.00762067, 0.00758796, 0.00723977,
       0.00727906, 0.00716834, 0.00713668, 0.00768003, 0.00762157,
       0.0075376 , 0.0073493 , 0.00716467, 0.00826182, 0.00730195,
       0.00715656, 0.0073772 , 0.00766082, 0.00738997, 0.00739698,
       0.00735044, 0.00722032, 0.00765858, 0.006951 , 0.00800371,
       0.00718155, 0.02586656, 0.02406259, 0.02004499, 0.01957617,
       0.01920695, 0.01921411, 0.01969204, 0.01993756, 0.02604146,
       0.02584615, 0.02474785, 0.02318335, 0.01934028, 0.02011466,
       0.0230454 , 0.02392278, 0.00924959, 0.0106627 , 0.01057386,
       0.01012225, 0.01074152, 0.01102619, 0.01005569, 0.00983853,
       0.00982003, 0.00981321, 0.01014032, 0.01026878, 0.01018562,
       0.01005588, 0.01136379, 0.0095582 , 0.00934401, 0.01053128,
       0.01072321, 0.01054859, 0.0107079 , 0.01016378, 0.01089983,
       0.0104239 , 0.01089516, 0.01075873, 0.01065307, 0.01051083,
       0.01095753, 0.01070743, 0.01049027, 0.00812912, 0.02956138,
       0.02796059, 0.02672768, 0.02286239, 0.02266974, 0.02217717,
       0.02198362, 0.02257404, 0.02344661, 0.0256206 , 0.02684889,
       0.0239418 , 0.02028341, 0.01997023, 0.02117796, 0.0229475 ,
       0.01218085, 0.00980568, 0.00916476, 0.00964913, 0.00954394,
       0.00872059, 0.00941124, 0.0086391 , 0.00843091, 0.00838466,
       0.00921664, 0.00952301, 0.01010666, 0.01061578, 0.01078491,
       0.00964341, 0.01014342, 0.01101503, 0.01096401, 0.01208792,
       0.01251698, 0.0127634 , 0.01229715, 0.01134801, 0.01278634,
       0.0111784 , 0.01103249, 0.01013694, 0.00939703, 0.01072407,
       0.01086693, 0.01156087, 0.03196039, 0.02803454, 0.02987061,
       0.02392306, 0.02371287, 0.02573037, 0.02966018, 0.02558217,
       0.03109565, 0.03013911, 0.023353 , 0.02711778, 0.02067032,
       0.02047334, 0.02014318, 0.020123 , 0.00901155, 0.01042852,
       0.00961175, 0.00911589, 0.01169457, 0.01191387, 0.01168494,
       0.01131072, 0.01204491, 0.01139503, 0.01141686, 0.01170058,
       0.01166625, 0.01137958, 0.0111341 , 0.01087079, 0.01250801,
       0.01060429, 0.00924859, 0.00871401, 0.00883298, 0.00952287,
       0.00912766, 0.00841236, 0.00853658, 0.00819187, 0.00891466,
       0.01012216, 0.01025972, 0.0102726 , 0.01149402, 0.01102996]), 'std_fit_time': array([0.00327083, 0.0006815
, 0.000403 , 0.0035974 , 0.00183386,
       0.00034289, 0.00043003, 0.00050261, 0.0002224 , 0.00042128,
       0.00043758, 0.0003496 , 0.00064167, 0.00037639, 0.00069739,
       0.00069162, 0.00022188, 0.00066136, 0.00096508, 0.00044887,
       0.00035806, 0.0006999 , 0.00044059, 0.00043674, 0.00052062,
       0.00032931, 0.00063546, 0.00034787, 0.00076878, 0.00045353,
       0.0002987 , 0.00048476, 0.00100666, 0.00039239, 0.00055779,
       0.00062037, 0.00048713, 0.00035722, 0.0002704 , 0.00083824,
       0.00127611, 0.00136356, 0.00077223, 0.00128775, 0.00049039,
       0.00043088, 0.00057342, 0.00093962, 0.00058205, 0.00053326,
       0.00073235, 0.00037882, 0.00054924, 0.0004288 , 0.000322 ,
       0.00037087, 0.00063504, 0.00033462, 0.00025251, 0.00040186,
       0.0002997 , 0.0002023 , 0.00085232, 0.00023785, 0.00093388,
       0.00047309, 0.00047865, 0.00136346, 0.00081159, 0.00039852,
       0.00069836, 0.00038519, 0.00104141, 0.00034976, 0.00084476,
       0.00059512, 0.00074461, 0.00036286, 0.00067633, 0.00044212,
       0.00019919, 0.00110007, 0.00027045, 0.00069424, 0.00039936,
       0.0004798 , 0.00047838, 0.00037439, 0.00050586, 0.00082705,
       0.00036793, 0.00054934, 0.00062643, 0.00090868, 0.00070589,
       0.0002842 , 0.00108514, 0.00253793, 0.00055256, 0.00040956,
       0.00068004, 0.00035162, 0.00138443, 0.00045375, 0.00289722,
       0.00113548, 0.00088016, 0.00060678, 0.00031068, 0.00162757,
       0.00082651, 0.00061877, 0.00045032, 0.00072458, 0.00073077,
       0.00032721, 0.00093499, 0.00053476, 0.00028208, 0.00045776,
       0.00089987, 0.0008679 , 0.00089025, 0.00080026, 0.00070535,
       0.00068237, 0.00060925, 0.00097778, 0.0010946 , 0.00051977,
       0.00075652, 0.00084818, 0.00100365, 0.00098223, 0.00023684,
       0.00044098, 0.00036098, 0.00043974, 0.00058629, 0.00080512,
       0.00101369, 0.0003121 , 0.0004921 , 0.00060653, 0.00134808,
       0.00098809, 0.00300444, 0.00060486, 0.00059399, 0.00044995,
       0.00100193, 0.00071261, 0.00251193, 0.00148817, 0.00078297,
       0.00208526, 0.00051427, 0.00048389, 0.00101491, 0.00154658,
       0.00086635, 0.00074776, 0.00045258, 0.00050806, 0.00029786,
       0.00055111, 0.00079653, 0.00053112, 0.00037825, 0.00032188,
       0.00076142, 0.00031352, 0.0004543 , 0.00067287, 0.00088239,

```
       0.00054708, 0.00080546, 0.00136566, 0.0005135 , 0.00087823,
       0.00094081, 0.00066847, 0.00113299, 0.00036734, 0.00121991,
       0.00049178, 0.00049788, 0.00090466, 0.00106141, 0.00076884,
       0.00034043, 0.0008968 , 0.00058671, 0.00152454, 0.00083394,
       0.00129199, 0.00079804, 0.00131455, 0.00214417, 0.00333422,
       0.0005938 , 0.00077093, 0.0009153 , 0.00183949, 0.00065874,
       0.00067504, 0.00048618, 0.00076419, 0.00055851, 0.00037315,
       0.00022382, 0.00022369, 0.00060422, 0.00064599, 0.00070578,
       0.00062934, 0.00046639, 0.00033707, 0.0009382 , 0.00104169,
       0.00086998, 0.00078886, 0.00074005, 0.00068729, 0.00042281,
       0.00072568, 0.00016793, 0.00014369, 0.00024419, 0.00073461,
       0.0011399 , 0.00031365, 0.00014603, 0.00024764, 0.00064877,
       0.00032703, 0.00025051, 0.00073377, 0.00055236, 0.00120277]), 'mean_score_time': array([0.00185328, 0.0014
9045, 0.00144839, 0.00182776, 0.00157938,
       0.00150218, 0.00145774, 0.00144553, 0.00151343, 0.00156598,
       0.00162997, 0.00145121, 0.00144768, 0.00142484, 0.00141683,
       0.00141468, 0.00146565, 0.00150523, 0.00155873, 0.001436  ,
       0.00145102, 0.00146365, 0.00141063, 0.00139198, 0.00141063,
       0.00140429, 0.00141301, 0.00140667, 0.00138516, 0.00139213,
       0.00139813, 0.00140061, 0.00166879, 0.00162239, 0.00157666,
       0.0016067 , 0.00153437, 0.00150347, 0.00149879, 0.0017849 ,
       0.00196233, 0.00245323, 0.00245638, 0.00210748, 0.00214448,
       0.00219817, 0.0020844 , 0.00241766, 0.00241079, 0.00197783,
       0.00233583, 0.00223608, 0.00229006, 0.00228415, 0.00221672,
       0.00226231, 0.00223603, 0.00224781, 0.00234194, 0.00227795,
       0.00236053, 0.00221734, 0.00214558, 0.00242558, 0.00230722,
       0.00219045, 0.00233727, 0.00191946, 0.00241556, 0.00217113,
       0.00213575, 0.002109  , 0.00216632, 0.00201283, 0.00215378,
       0.00203795, 0.002176  , 0.00218883, 0.00213571, 0.0022594 ,
       0.00252104, 0.00236363, 0.00211415, 0.00222411, 0.00229321,
       0.00217342, 0.00219417, 0.00235271, 0.0020268 , 0.00210257,
       0.00222468, 0.00211992, 0.00226064, 0.00216961, 0.00231724,
       0.00201364, 0.00213237, 0.00186009, 0.0015276 , 0.00144439,
       0.00139565, 0.00137005, 0.00150399, 0.00170078, 0.00238686,
       0.00223465, 0.002179  , 0.00192952, 0.00159202, 0.00186162,
       0.00212574, 0.0020905 , 0.00200534, 0.00218558, 0.00223365,
       0.00210814, 0.00224566, 0.00223308, 0.00215282, 0.00211673,
       0.00205746, 0.00208001, 0.00212669, 0.00227361, 0.00225158,
       0.00222631, 0.00226083, 0.00190177, 0.00191245, 0.00226569,
       0.00225959, 0.00227141, 0.00227728, 0.00216222, 0.00242157,
       0.00240288, 0.00231709, 0.0022851 , 0.00231719, 0.00215192,
       0.00236049, 0.00225654, 0.00209894, 0.00158911, 0.00216045,
       0.00195212, 0.0018816 , 0.0014842 , 0.00142365, 0.00142498,
       0.00155816, 0.0017128 , 0.00192928, 0.00201192, 0.00217838,
       0.0017673 , 0.00155978, 0.00143476, 0.0016932 , 0.00201201,
       0.00218158, 0.00163379, 0.00145812, 0.00147958, 0.00145445,
       0.0014502 , 0.00151129, 0.00143733, 0.00144091, 0.00148373,
       0.00169778, 0.00179801, 0.00198054, 0.00215521, 0.00218225,
       0.00196896, 0.00173259, 0.00202403, 0.00204186, 0.00232725,
       0.00220881, 0.00230684, 0.00214958, 0.00210032, 0.00230207,
       0.00228257, 0.00221334, 0.00185313, 0.00180182, 0.00216269,
       0.00226703, 0.00235586, 0.00228577, 0.00199881, 0.00213962,
       0.00162501, 0.00166154, 0.00186906, 0.00211625, 0.00211759,
       0.00255675, 0.00225477, 0.00178514, 0.00208125, 0.0015697 ,
       0.00148702, 0.00148501, 0.00142727, 0.00145235, 0.00202346,
       0.00162039, 0.00179644, 0.002107  , 0.00217886, 0.00214882,
       0.00216331, 0.0022532 , 0.00214152, 0.00228577, 0.00231228,
       0.00237579, 0.00226479, 0.00227079, 0.00217271, 0.00213795,
       0.00163889, 0.00152082, 0.00146084, 0.0014318 , 0.0015533 ,
       0.00151777, 0.00143404, 0.00143142, 0.00147591, 0.00174341,
       0.00184269, 0.00210443, 0.00213032, 0.00222845, 0.00199499]), 'std_score_time': array([3.80549913e-04, 2.7
1422982e-05, 2.13434570e-05, 3.79907072e-04,
       9.08897905e-05, 3.58547300e-05, 6.60104194e-05, 6.77093653e-05,
       8.69631759e-05, 1.25081976e-04, 1.42987603e-04, 1.87084311e-05,
       5.39894648e-05, 2.04169845e-05, 5.01701129e-05, 5.33705408e-05,
       3.13059482e-05, 8.99659927e-05, 1.02088530e-04, 4.83936861e-06,
       8.88888926e-06, 9.24498592e-06, 6.76470442e-06, 4.28251803e-06,
       3.33229009e-05, 1.78715361e-05, 1.46083482e-05, 3.89625857e-05,
       1.83946694e-05, 1.84425671e-05, 2.69486831e-06, 3.09246452e-06,
       6.76889787e-05, 5.43197688e-05, 5.24741906e-05, 5.09214310e-05,
       2.45441940e-05, 2.88870453e-06, 2.32071310e-05, 4.88034387e-05,
       1.22351397e-04, 3.51551784e-04, 1.46484297e-04, 1.54532028e-05,
       6.67825038e-05, 1.11831317e-04, 1.09509949e-04, 2.43410942e-04,
       4.07184777e-04, 1.24297152e-04, 1.46272423e-04, 8.13397261e-05,
       1.19835107e-04, 1.75395239e-04, 8.96576356e-05, 1.43312644e-04,
       4.35863072e-05, 1.13546032e-04, 8.97993885e-05, 8.52747769e-05,
       1.85080545e-04, 6.80460826e-05, 1.53121418e-04, 2.09097014e-04,
       1.78437243e-04, 1.93087940e-04, 2.70242065e-04, 2.00573795e-04,
       2.73040621e-04, 7.06896573e-05, 1.09016175e-04, 9.27495955e-05,
       6.68386235e-05, 8.41509647e-05, 7.62138244e-05, 1.35173150e-04,
       6.60183413e-05, 1.13402022e-04, 4.39861021e-05, 7.36357007e-05,
       5.01696053e-04, 1.94144493e-04, 1.53566190e-04, 1.89164588e-04,
       2.20700544e-04, 5.82333170e-05, 4.52349855e-05, 3.97905195e-04,
       1.39727689e-04, 1.10253928e-04, 1.58133178e-04, 9.63485556e-05,
```

```
        1.61528360e-04, 9.42406850e-05, 1.81441858e-04, 1.91523198e-04,
        1.03494910e-04, 3.31339638e-04, 1.31230004e-04, 3.52112521e-05,
        5.91478775e-05, 1.40667770e-05, 1.46683930e-04, 7.96924006e-05,
        2.07738933e-04, 1.78508459e-04, 1.09188900e-04, 8.98662849e-05,
        6.89226175e-05, 1.86486093e-04, 9.30259069e-05, 9.61907610e-05,
        2.23302999e-04, 1.34889554e-04, 1.33594982e-04, 1.07728944e-04,
        9.83618090e-05, 1.14627864e-04, 1.38659835e-04, 1.01243478e-04,
        7.20252386e-05, 2.59377035e-04, 1.21871126e-04, 1.36982328e-04,
        6.07215004e-05, 5.79247052e-05, 7.57264249e-05, 2.08724463e-04,
        2.08510459e-04, 9.15555904e-05, 1.14886450e-04, 8.47384039e-05,
        1.22215568e-04, 9.26178084e-05, 3.66014357e-04, 2.46826390e-04,
        1.90788595e-04, 5.72477911e-05, 1.10054595e-04, 1.19085667e-04,
        1.87445445e-04, 6.06750879e-05, 1.93604906e-04, 1.00533106e-04,
        5.75409497e-05, 1.66692027e-04, 3.14439308e-04, 6.95331589e-05,
        9.29820232e-06, 1.70887421e-05, 2.37313600e-04, 3.54107377e-05,
        2.69722128e-04, 1.72306979e-04, 8.42095501e-05, 2.58593042e-04,
        4.99779938e-05, 2.54768520e-05, 1.53391955e-04, 1.96813230e-04,
        2.05319369e-04, 1.14863985e-04, 8.85865398e-06, 6.22672285e-05,
        8.57326159e-06, 3.82099823e-05, 5.88049457e-05, 3.52442340e-05,
        5.65454485e-05, 1.03163298e-04, 2.17641298e-04, 9.49686295e-05,
        1.85494251e-04, 3.32046472e-05, 1.93771801e-04, 2.58736588e-04,
        1.83642799e-04, 2.25084141e-04, 1.32869063e-04, 1.86765337e-04,
        2.53211756e-05, 7.07338704e-05, 2.10482857e-04, 1.42119921e-04,
        2.17291957e-04, 7.08348620e-05, 1.12650051e-04, 2.18145553e-04,
        2.27117824e-04, 8.06527450e-05, 1.48698868e-04, 1.49357273e-04,
        9.98215513e-05, 9.60120351e-05, 5.61173524e-05, 5.48668237e-05,
        9.72241616e-05, 1.17210473e-04, 1.99016510e-04, 4.54004904e-04,
        3.79263800e-04, 9.01503998e-05, 9.47184655e-05, 2.68019414e-04,
        6.70671760e-05, 2.74481816e-05, 1.01440939e-04, 2.89607824e-05,
        8.24463167e-05, 6.02768926e-04, 6.83997266e-05, 1.88399647e-04,
        1.39464932e-04, 1.29205748e-04, 1.52052177e-04, 2.41638311e-04,
        4.89134992e-05, 2.12651004e-04, 2.42079776e-04, 9.03514691e-05,
        8.43532076e-05, 1.31434815e-04, 1.95600764e-04, 1.24801685e-04,
        1.91947977e-04, 2.14679587e-04, 3.41820743e-05, 2.27213782e-05,
        4.10634044e-06, 1.33729829e-04, 1.12649385e-04, 6.24529265e-06,
        4.48176198e-06, 1.38304199e-04, 5.72753655e-04, 1.16936903e-04,
        5.04291310e-05, 1.19495849e-04, 4.31434609e-05, 2.88931629e-04]), 'param_max_depth': masked_array(data=[No
ne, None, None, None, None, None, None, None, None,
                   None, None, None, None, None, None, None, None, None,
                   None, None, None, None, None, None, None, None, None,
                   None, None, None, None, None, None, None, None, None,
                   None, None, None, None, None, None, None, None, None,
                   None, None, None, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,
                   5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,
                   5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,
                   10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
                   10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
                   10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
                   10, 10, 10, 10, 10, 10, 15, 15, 15, 15, 15, 15, 15, 15,
                   15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15,
                   15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15,
                   15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 20, 20,
                   20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20,
                   20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20,
                   20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20,
                   20, 20, 20, 20],
             mask=[False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
```

```
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False],
       fill_value='?',
            dtype=object), 'param_max_features': masked_array(data=[None, None, None, None, None, None, None, Non
e, None,
                   None, None, None, None, None, None, None, 'sqrt',
                   'sqrt', 'sqrt', 'sqrt', 'sqrt', 'sqrt', 'sqrt', 'sqrt',
                   'sqrt', 'sqrt', 'sqrt', 'sqrt', 'sqrt', 'sqrt', 'sqrt',
                   'sqrt', 'log2', 'log2', 'log2', 'log2', 'log2', 'log2',
                   'log2', 'log2', 'log2', 'log2', 'log2', 'log2', 'log2',
                   'log2', 'log2', 'log2', None, None, None, None, None,
                   None, None, None, None, None, None, None, None, None,
                   None, None, 'sqrt', 'sqrt', 'sqrt', 'sqrt', 'sqrt',
                   'sqrt', 'sqrt', 'sqrt', 'sqrt', 'sqrt', 'sqrt', 'sqrt',
                   'sqrt', 'sqrt', 'sqrt', 'sqrt', 'log2', 'log2', 'log2',
                   'log2', 'log2', 'log2', 'log2', 'log2', 'log2', 'log2',
                   'log2', 'log2', 'log2', 'log2', 'log2', 'log2', None,
                   None, None, None, None, None, None, None, None, None,
                   None, None, None, None, None, None, 'sqrt', 'sqrt',
                   'sqrt', 'sqrt', 'sqrt', 'sqrt', 'sqrt', 'sqrt', 'sqrt',
                   'sqrt', 'sqrt', 'sqrt', 'sqrt', 'sqrt', 'sqrt', 'sqrt',
                   'log2', 'log2', 'log2', 'log2', 'log2', 'log2', 'log2',
                   'log2', 'log2', 'log2', 'log2', 'log2', 'log2', 'log2',
                   'log2', 'log2', None, None, None, None, None, None,
                   None, None, None, None, None, None, None, None, None,
                   None, 'sqrt', 'sqrt', 'sqrt', 'sqrt', 'sqrt', 'sqrt',
                   'sqrt', 'sqrt', 'sqrt', 'sqrt', 'sqrt', 'sqrt', 'sqrt',
                   'sqrt', 'sqrt', 'sqrt', 'log2', 'log2', 'log2', 'log2',
                   'log2', 'log2', 'log2', 'log2', 'log2', 'log2', 'log2',
                   'log2', 'log2', 'log2', 'log2', 'log2', None, None,
                   None, None, None, None, None, None, None, None, None,
                   None, None, None, None, None, 'sqrt', 'sqrt', 'sqrt',
                   'sqrt', 'sqrt', 'sqrt', 'sqrt', 'sqrt', 'sqrt', 'sqrt',
                   'sqrt', 'sqrt', 'sqrt', 'sqrt', 'sqrt', 'sqrt', 'log2',
                   'log2', 'log2', 'log2', 'log2', 'log2', 'log2', 'log2',
                   'log2', 'log2', 'log2', 'log2', 'log2', 'log2', 'log2',
                   'log2'],
             mask=[False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False],
       fill_value='?',
            dtype=object), 'param_min_samples_leaf': masked_array(data=[1, 1, 1, 1, 2, 2, 2, 2, 5, 5, 5, 5, 10, 1
0, 10, 10, 1,
                   1, 1, 1, 2, 2, 2, 2, 5, 5, 5, 5, 10, 10, 10, 10, 1, 1,
                   1, 1, 2, 2, 2, 2, 5, 5, 5, 5, 10, 10, 10, 10, 1, 1, 1,
                   1, 2, 2, 2, 2, 5, 5, 5, 5, 10, 10, 10, 10, 1, 1, 1, 1,
                   2, 2, 2, 2, 5, 5, 5, 5, 10, 10, 10, 10, 1, 1, 1, 1, 2,
                   2, 2, 2, 5, 5, 5, 5, 10, 10, 10, 10, 1, 1, 1, 1, 2, 2,
                   2, 2, 5, 5, 5, 5, 10, 10, 10, 10, 1, 1, 1, 1, 2, 2, 2,
                   2, 5, 5, 5, 5, 10, 10, 10, 10, 1, 1, 1, 1, 2, 2, 2, 2,
                   5, 5, 5, 5, 10, 10, 10, 10, 1, 1, 1, 1, 2, 2, 2, 2, 5,
                   5, 5, 5, 10, 10, 10, 10, 1, 1, 1, 1, 2, 2, 2, 2, 5, 5,
                   5, 5, 10, 10, 10, 10, 1, 1, 1, 1, 2, 2, 2, 2, 5, 5, 5,
                   5, 10, 10, 10, 10, 1, 1, 1, 1, 2, 2, 2, 2, 5, 5, 5, 5,
                   10, 10, 10, 10, 1, 1, 1, 1, 2, 2, 2, 2, 5, 5, 5, 5, 10,
                   10, 10, 10, 1, 1, 1, 1, 2, 2, 2, 2, 5, 5, 5, 5, 10, 10,
                   10, 10],
```

```
             mask=[False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False],
       fill_value='?',
            dtype=object), 'param_min_samples_split': masked_array(data=[2, 5, 10, 20, 2, 5, 10, 20, 2, 5, 10, 2
0, 2, 5, 10, 20,
                   2, 5, 10, 20, 2, 5, 10, 20, 2, 5, 10, 20, 2, 5, 10, 20,
                   2, 5, 10, 20, 2, 5, 10, 20, 2, 5, 10, 20, 2, 5, 10, 20,
                   2, 5, 10, 20, 2, 5, 10, 20, 2, 5, 10, 20, 2, 5, 10, 20,
                   2, 5, 10, 20, 2, 5, 10, 20, 2, 5, 10, 20, 2, 5, 10, 20,
                   2, 5, 10, 20, 2, 5, 10, 20, 2, 5, 10, 20, 2, 5, 10, 20,
                   2, 5, 10, 20, 2, 5, 10, 20, 2, 5, 10, 20, 2, 5, 10, 20,
                   2, 5, 10, 20, 2, 5, 10, 20, 2, 5, 10, 20, 2, 5, 10, 20,
                   2, 5, 10, 20, 2, 5, 10, 20, 2, 5, 10, 20, 2, 5, 10, 20,
                   2, 5, 10, 20, 2, 5, 10, 20, 2, 5, 10, 20, 2, 5, 10, 20,
                   2, 5, 10, 20, 2, 5, 10, 20, 2, 5, 10, 20, 2, 5, 10, 20,
                   2, 5, 10, 20, 2, 5, 10, 20, 2, 5, 10, 20, 2, 5, 10, 20,
                   2, 5, 10, 20, 2, 5, 10, 20, 2, 5, 10, 20, 2, 5, 10, 20,
                   2, 5, 10, 20, 2, 5, 10, 20, 2, 5, 10, 20, 2, 5, 10, 20,
                   2, 5, 10, 20, 2, 5, 10, 20, 2, 5, 10, 20, 2, 5, 10, 20],
             mask=[False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False],
       fill_value='?',
            dtype=object), 'params': [{'max_depth': None, 'max_features': None, 'min_samples_leaf': 1, 'min_sampl
es_split': 2}, {'max_depth': None, 'max_features': None, 'min_samples_leaf': 1, 'min_samples_split': 5}, {'max_de
pth': None, 'max_features': None, 'min_samples_leaf': 1, 'min_samples_split': 10}, {'max_depth': None, 'max_featu
res': None, 'min_samples_leaf': 1, 'min_samples_split': 20}, {'max_depth': None, 'max_features': None, 'min_sampl
es_leaf': 2, 'min_samples_split': 2}, {'max_depth': None, 'max_features': None, 'min_samples_leaf': 2, 'min_sampl
es_split': 5}, {'max_depth': None, 'max_features': None, 'min_samples_leaf': 2, 'min_samples_split': 10}, {'max_d
```

epth': None, 'max_features': None, 'min_samples_leaf': 2, 'min_samples_split': 20}, {'max_depth': None, 'max_features': None, 'min_samples_leaf': 5, 'min_samples_split': 2}, {'max_depth': None, 'max_features': None, 'min_samples_leaf': 5, 'min_samples_split': 5}, {'max_depth': None, 'max_features': None, 'min_samples_leaf': 5, 'min_samples_split': 10}, {'max_depth': None, 'max_features': None, 'min_samples_leaf': 5, 'min_samples_split': 20}, {'max_depth': None, 'max_features': None, 'min_samples_leaf': 10, 'min_samples_split': 2}, {'max_depth': None, 'max_features': None, 'min_samples_leaf': 10, 'min_samples_split': 5}, {'max_depth': None, 'max_features': None, 'min_samples_leaf': 10, 'min_samples_split': 10}, {'max_depth': None, 'max_features': None, 'min_samples_leaf': 10, 'min_samples_split': 20}, {'max_depth': None, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 2}, {'max_depth': None, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 5}, {'max_depth': None, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 10}, {'max_depth': None, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 20}, {'max_depth': None, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 2}, {'max_depth': None, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 5}, {'max_depth': None, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 10}, {'max_depth': None, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 20}, {'max_depth': None, 'max_features': 'sqrt', 'min_samples_leaf': 5, 'min_samples_split': 2}, {'max_depth': None, 'max_features': 'sqrt', 'min_samples_leaf': 5, 'min_samples_split': 5}, {'max_depth': None, 'max_features': 'sqrt', 'min_samples_leaf': 5, 'min_samples_split': 10}, {'max_depth': None, 'max_features': 'sqrt', 'min_samples_leaf': 5, 'min_samples_split': 20}, {'max_depth': None, 'max_features': 'sqrt', 'min_samples_leaf': 10, 'min_samples_split': 2}, {'max_depth': None, 'max_features': 'sqrt', 'min_samples_leaf': 10, 'min_samples_split': 5}, {'max_depth': None, 'max_features': 'sqrt', 'min_samples_leaf': 10, 'min_samples_split': 10}, {'max_depth': None, 'max_features': 'sqrt', 'min_samples_leaf': 10, 'min_samples_split': 20}, {'max_depth': None, 'max_features': 'log2', 'min_samples_leaf': 1, 'min_samples_split': 2}, {'max_depth': None, 'max_features': 'log2', 'min_samples_leaf': 1, 'min_samples_split': 5}, {'max_depth': None, 'max_features': 'log2', 'min_samples_leaf': 1, 'min_samples_split': 10}, {'max_depth': None, 'max_features': 'log2', 'min_samples_leaf': 1, 'min_samples_split': 20}, {'max_depth': None, 'max_features': 'log2', 'min_samples_leaf': 2, 'min_samples_split': 2}, {'max_depth': None, 'max_features': 'log2', 'min_samples_leaf': 2, 'min_samples_split': 5}, {'max_depth': None, 'max_features': 'log2', 'min_samples_leaf': 2, 'min_samples_split': 10}, {'max_depth': None, 'max_features': 'log2', 'min_samples_leaf': 2, 'min_samples_split': 20}, {'max_depth': None, 'max_features': 'log2', 'min_samples_leaf': 5, 'min_samples_split': 2}, {'max_depth': None, 'max_features': 'log2', 'min_samples_leaf': 5, 'min_samples_split': 5}, {'max_depth': None, 'max_features': 'log2', 'min_samples_leaf': 5, 'min_samples_split': 10}, {'max_depth': None, 'max_features': 'log2', 'min_samples_leaf': 5, 'min_samples_split': 20}, {'max_depth': None, 'max_features': 'log2', 'min_samples_leaf': 10, 'min_samples_split': 2}, {'max_depth': None, 'max_features': 'log2', 'min_samples_leaf': 10, 'min_samples_split': 5}, {'max_depth': None, 'max_features': 'log2', 'min_samples_leaf': 10, 'min_samples_split': 10}, {'max_depth': None, 'max_features': 'log2', 'min_samples_leaf': 10, 'min_samples_split': 20}, {'max_depth': 5, 'max_features': None, 'min_samples_leaf': 1, 'min_samples_split': 2}, {'max_depth': 5, 'max_features': None, 'min_samples_leaf': 1, 'min_samples_split': 5}, {'max_depth': 5, 'max_features': None, 'min_samples_leaf': 1, 'min_samples_split': 10}, {'max_depth': 5, 'max_features': None, 'min_samples_leaf': 1, 'min_samples_split': 20}, {'max_depth': 5, 'max_features': None, 'min_samples_leaf': 2, 'min_samples_split': 2}, {'max_depth': 5, 'max_features': None, 'min_samples_leaf': 2, 'min_samples_split': 5}, {'max_depth': 5, 'max_features': None, 'min_samples_leaf': 2, 'min_samples_split': 10}, {'max_depth': 5, 'max_features': None, 'min_samples_leaf': 2, 'min_samples_split': 20}, {'max_depth': 5, 'max_features': None, 'min_samples_leaf': 5, 'min_samples_split': 2}, {'max_depth': 5, 'max_features': None, 'min_samples_leaf': 5, 'min_samples_split': 5}, {'max_depth': 5, 'max_features': None, 'min_samples_leaf': 5, 'min_samples_split': 10}, {'max_depth': 5, 'max_features': None, 'min_samples_leaf': 5, 'min_samples_split': 20}, {'max_depth': 5, 'max_features': None, 'min_samples_leaf': 10, 'min_samples_split': 2}, {'max_depth': 5, 'max_features': None, 'min_samples_leaf': 10, 'min_samples_split': 5}, {'max_depth': 5, 'max_features': None, 'min_samples_leaf': 10, 'min_samples_split': 10}, {'max_depth': 5, 'max_features': None, 'min_samples_leaf': 10, 'min_samples_split': 20}, {'max_depth': 5, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 2}, {'max_depth': 5, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 5}, {'max_depth': 5, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 10}, {'max_depth': 5, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 20}, {'max_depth': 5, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 2}, {'max_depth': 5, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 5}, {'max_depth': 5, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 10}, {'max_depth': 5, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 20}, {'max_depth': 5, 'max_features': 'sqrt', 'min_samples_leaf': 5, 'min_samples_split': 2}, {'max_depth': 5, 'max_features': 'sqrt', 'min_samples_leaf': 5, 'min_samples_split': 5}, {'max_depth': 5, 'max_features': 'sqrt', 'min_samples_leaf': 5, 'min_samples_split': 10}, {'max_depth': 5, 'max_features': 'sqrt', 'min_samples_leaf': 5, 'min_samples_split': 20}, {'max_depth': 5, 'max_features': 'sqrt', 'min_samples_leaf': 10, 'min_samples_split': 2}, {'max_depth': 5, 'max_features': 'sqrt', 'min_samples_leaf': 10, 'min_samples_split': 5}, {'max_depth': 5, 'max_features': 'sqrt', 'min_samples_leaf': 10, 'min_samples_split': 10}, {'max_depth': 5, 'max_features': 'sqrt', 'min_samples_leaf': 10, 'min_samples_split': 20}, {'max_depth': 5, 'max_features': 'log2', 'min_samples_leaf': 1, 'min_samples_split': 2}, {'max_depth': 5, 'max_features': 'log2', 'min_samples_leaf': 1, 'min_samples_split': 5}, {'max_depth': 5, 'max_features': 'log2', 'min_samples_leaf': 1, 'min_samples_split': 10}, {'max_depth': 5, 'max_features': 'log2', 'min_samples_leaf': 1, 'min_samples_split': 20}, {'max_depth': 5, 'max_features': 'log2', 'min_samples_leaf': 2, 'min_samples_split': 2}, {'max_depth': 5, 'max_features': 'log2', 'min_samples_leaf': 2, 'min_samples_split': 5}, {'max_depth': 5, 'max_features': 'log2', 'min_samples_leaf': 2, 'min_samples_split': 10}, {'max_depth': 5, 'max_features': 'log2', 'min_samples_leaf': 2, 'min_samples_split': 20}, {'max_depth': 5, 'max_features': 'log2', 'min_samples_leaf': 5, 'min_samples_split': 2}, {'max_depth': 5, 'max_features': 'log2', 'min_samples_leaf': 5, 'min_samples_split': 5}, {'max_depth': 5, 'max_features': 'log2', 'min_samples_leaf': 5, 'min_samples_split': 10}, {'max_depth': 5, 'max_features': 'log2', 'min_samples_leaf': 5, 'min_samples_split': 20}, {'max_depth': 5, 'max_features': 'log2', 'min_samples_leaf': 10, 'min_samples_split': 2}, {'max_depth': 5, 'max_features': 'log2', 'min_samples_leaf': 10, 'min_samples_split': 5}, {'max_depth': 5, 'max_features': 'log2', 'min_samples_leaf': 10, 'min_samples_split': 10}, {'max_depth': 5, 'max_features': 'log2', 'min_samples_leaf': 10, 'min_samples_split': 20}, {'max_depth': 10, 'max_features': None, 'min_samples_leaf': 1, 'min_samples_split': 2}, {'max_depth': 10, 'max_features': None, 'min_samples_leaf': 1, 'min_samples_split': 5}, {'max_depth': 10, 'max_features': None, 'min_samples_leaf': 1, 'min_samples_split': 10}, {'max_depth': 10, 'max_features': None, 'min_samples_leaf': 1, 'min_samples_split': 20}, {'max_depth': 10, 'max_features': None, 'min_samples_leaf': 2, 'min_samples_split': 2}, {'max_depth': 10, 'max_features': None, 'min_samples_leaf': 2, 'min_samples_split': 5}, {'max_depth': 10, 'max_features': None, 'min_samples_leaf': 2, 'min_samples_split': 10}, {'max_depth': 10, 'max_features': None, 'min_samples_leaf': 2, 'min_samples_split': 20}, {'max_depth': 10, 'max_features': None, 'min_samples_leaf': 5, 'min_samples_split': 2}, {'max_depth': 10, 'max_features': None, 'min_samples_leaf': 5, 'min_samples_split': 5}, {'max_depth': 10, 'max_features': None, 'min_samples_leaf': 5, 'min_samples_split': 10}, {'max_depth': 10, 'max_features': None, 'min_samples_leaf': 5, 'min_samples_split': 20}, {'max_depth': 10, 'max_features': None, 'min_samples_leaf': 10, 'min_samples_split': 2}, {'max_depth': 10, 'max_features': None, 'min_samples_leaf': 10, 'min_samples_split': 5}, {'max_depth': 10, 'max_features': None, 'min_samples_leaf': 10, 'min_samples_split': 10}, {'max_depth': 10, 'max_features': None, 'min_samples_leaf': 10, 'min_samples_split': 20}, {'max

_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 2}, {'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 5}, {'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 10}, {'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 20}, {'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 2}, {'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 5}, {'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 10}, {'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 20}, {'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 5, 'min_samples_split': 2}, {'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 5, 'min_samples_split': 5}, {'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 5, 'min_samples_split': 10}, {'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 5, 'min_samples_split': 20}, {'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 10, 'min_samples_split': 2}, {'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 10, 'min_samples_split': 5}, {'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 10, 'min_samples_split': 10}, {'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 10, 'min_samples_split': 20}, {'max_depth': 10, 'max_features': 'log2', 'min_samples_leaf': 1, 'min_samples_split': 2}, {'max_depth': 10, 'max_features': 'log2', 'min_samples_leaf': 1, 'min_samples_split': 5}, {'max_depth': 10, 'max_features': 'log2', 'min_samples_leaf': 1, 'min_samples_split': 10}, {'max_depth': 10, 'max_features': 'log2', 'min_samples_leaf': 1, 'min_samples_split': 20}, {'max_depth': 10, 'max_features': 'log2', 'min_samples_leaf': 2, 'min_samples_split': 2}, {'max_depth': 10, 'max_features': 'log2', 'min_samples_leaf': 2, 'min_samples_split': 5}, {'max_depth': 10, 'max_features': 'log2', 'min_samples_leaf': 2, 'min_samples_split': 10}, {'max_depth': 10, 'max_features': 'log2', 'min_samples_leaf': 2, 'min_samples_split': 20}, {'max_depth': 10, 'max_features': 'log2', 'min_samples_leaf': 5, 'min_samples_split': 2}, {'max_depth': 10, 'max_features': 'log2', 'min_samples_leaf': 5, 'min_samples_split': 5}, {'max_depth': 10, 'max_features': 'log2', 'min_samples_leaf': 5, 'min_samples_split': 10}, {'max_depth': 10, 'max_features': 'log2', 'min_samples_leaf': 5, 'min_samples_split': 20}, {'max_depth': 10, 'max_features': 'log2', 'min_samples_leaf': 10, 'min_samples_split': 2}, {'max_depth': 10, 'max_features': 'log2', 'min_samples_leaf': 10, 'min_samples_split': 5}, {'max_depth': 10, 'max_features': 'log2', 'min_samples_leaf': 10, 'min_samples_split': 10}, {'max_depth': 10, 'max_features': 'log2', 'min_samples_leaf': 10, 'min_samples_split': 20}, {'max_depth': 15, 'max_features': None, 'min_samples_leaf': 1, 'min_samples_split': 2}, {'max_depth': 15, 'max_features': None, 'min_samples_leaf': 1, 'min_samples_split': 5}, {'max_depth': 15, 'max_features': None, 'min_samples_leaf': 1, 'min_samples_split': 10}, {'max_depth': 15, 'max_features': None, 'min_samples_leaf': 1, 'min_samples_split': 20}, {'max_depth': 15, 'max_features': None, 'min_samples_leaf': 2, 'min_samples_split': 2}, {'max_depth': 15, 'max_features': None, 'min_samples_leaf': 2, 'min_samples_split': 5}, {'max_depth': 15, 'max_features': None, 'min_samples_leaf': 2, 'min_samples_split': 10}, {'max_depth': 15, 'max_features': None, 'min_samples_leaf': 2, 'min_samples_split': 20}, {'max_depth': 15, 'max_features': None, 'min_samples_leaf': 5, 'min_samples_split': 2}, {'max_depth': 15, 'max_features': None, 'min_samples_leaf': 5, 'min_samples_split': 5}, {'max_depth': 15, 'max_features': None, 'min_samples_leaf': 5, 'min_samples_split': 10}, {'max_depth': 15, 'max_features': None, 'min_samples_leaf': 5, 'min_samples_split': 20}, {'max_depth': 15, 'max_features': None, 'min_samples_leaf': 10, 'min_samples_split': 2}, {'max_depth': 15, 'max_features': None, 'min_samples_leaf': 10, 'min_samples_split': 5}, {'max_depth': 15, 'max_features': None, 'min_samples_leaf': 10, 'min_samples_split': 10}, {'max_depth': 15, 'max_features': None, 'min_samples_leaf': 10, 'min_samples_split': 20}, {'max_depth': 15, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 2}, {'max_depth': 15, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 5}, {'max_depth': 15, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 10}, {'max_depth': 15, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 20}, {'max_depth': 15, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 2}, {'max_depth': 15, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 5}, {'max_depth': 15, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 10}, {'max_depth': 15, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 20}, {'max_depth': 15, 'max_features': 'sqrt', 'min_samples_leaf': 5, 'min_samples_split': 2}, {'max_depth': 15, 'max_features': 'sqrt', 'min_samples_leaf': 5, 'min_samples_split': 5}, {'max_depth': 15, 'max_features': 'sqrt', 'min_samples_leaf': 5, 'min_samples_split': 10}, {'max_depth': 15, 'max_features': 'sqrt', 'min_samples_leaf': 5, 'min_samples_split': 20}, {'max_depth': 15, 'max_features': 'sqrt', 'min_samples_leaf': 10, 'min_samples_split': 2}, {'max_depth': 15, 'max_features': 'sqrt', 'min_samples_leaf': 10, 'min_samples_split': 5}, {'max_depth': 15, 'max_features': 'sqrt', 'min_samples_leaf': 10, 'min_samples_split': 10}, {'max_depth': 15, 'max_features': 'sqrt', 'min_samples_leaf': 10, 'min_samples_split': 20}, {'max_depth': 15, 'max_features': 'log2', 'min_samples_leaf': 1, 'min_samples_split': 2}, {'max_depth': 15, 'max_features': 'log2', 'min_samples_leaf': 1, 'min_samples_split': 5}, {'max_depth': 15, 'max_features': 'log2', 'min_samples_leaf': 1, 'min_samples_split': 10}, {'max_depth': 15, 'max_features': 'log2', 'min_samples_leaf': 1, 'min_samples_split': 20}, {'max_depth': 15, 'max_features': 'log2', 'min_samples_leaf': 2, 'min_samples_split': 2}, {'max_depth': 15, 'max_features': 'log2', 'min_samples_leaf': 2, 'min_samples_split': 5}, {'max_depth': 15, 'max_features': 'log2', 'min_samples_leaf': 2, 'min_samples_split': 10}, {'max_depth': 15, 'max_features': 'log2', 'min_samples_leaf': 2, 'min_samples_split': 20}, {'max_depth': 15, 'max_features': 'log2', 'min_samples_leaf': 5, 'min_samples_split': 2}, {'max_depth': 15, 'max_features': 'log2', 'min_samples_leaf': 5, 'min_samples_split': 5}, {'max_depth': 15, 'max_features': 'log2', 'min_samples_leaf': 5, 'min_samples_split': 10}, {'max_depth': 15, 'max_features': 'log2', 'min_samples_leaf': 5, 'min_samples_split': 20}, {'max_depth': 15, 'max_features': 'log2', 'min_samples_leaf': 10, 'min_samples_split': 2}, {'max_depth': 15, 'max_features': 'log2', 'min_samples_leaf': 10, 'min_samples_split': 5}, {'max_depth': 15, 'max_features': 'log2', 'min_samples_leaf': 10, 'min_samples_split': 10}, {'max_depth': 15, 'max_features': 'log2', 'min_samples_leaf': 10, 'min_samples_split': 20}, {'max_depth': 20, 'max_features': None, 'min_samples_leaf': 1, 'min_samples_split': 2}, {'max_depth': 20, 'max_features': None, 'min_samples_leaf': 1, 'min_samples_split': 5}, {'max_depth': 20, 'max_features': None, 'min_samples_leaf': 1, 'min_samples_split': 10}, {'max_depth': 20, 'max_features': None, 'min_samples_leaf': 1, 'min_samples_split': 20}, {'max_depth': 20, 'max_features': None, 'min_samples_leaf': 2, 'min_samples_split': 2}, {'max_depth': 20, 'max_features': None, 'min_samples_leaf': 2, 'min_samples_split': 5}, {'max_depth': 20, 'max_features': None, 'min_samples_leaf': 2, 'min_samples_split': 10}, {'max_depth': 20, 'max_features': None, 'min_samples_leaf': 2, 'min_samples_split': 20}, {'max_depth': 20, 'max_features': None, 'min_samples_leaf': 5, 'min_samples_split': 2}, {'max_depth': 20, 'max_features': None, 'min_samples_leaf': 5, 'min_samples_split': 5}, {'max_depth': 20, 'max_features': None, 'min_samples_leaf': 5, 'min_samples_split': 10}, {'max_depth': 20, 'max_features': None, 'min_samples_leaf': 5, 'min_samples_split': 20}, {'max_depth': 20, 'max_features': None, 'min_samples_leaf': 10, 'min_samples_split': 2}, {'max_depth': 20, 'max_features': None, 'min_samples_leaf': 10, 'min_samples_split': 5}, {'max_depth': 20, 'max_features': None, 'min_samples_leaf': 10, 'min_samples_split': 10}, {'max_depth': 20, 'max_features': None, 'min_samples_leaf': 10, 'min_samples_split': 20}, {'max_depth': 20, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 2}, {'max_depth': 20, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 5}, {'max_depth': 20, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 10}, {'max_depth': 20, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 20}, {'max_depth': 20, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 2}, {'max_depth': 20, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 5}, {'max_depth': 20, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 10}, {'max_depth': 20, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 20}, {'max_depth': 20, 'max_features': 'sqrt', 'min_samples_leaf': 5, 'min_samples_split': 2}, {'max_depth': 20, 'max_

features': 'sqrt', 'min_samples_leaf': 5, 'min_samples_split': 5}, {'max_depth': 20, 'max_features': 'sqrt', 'min
_samples_leaf': 5, 'min_samples_split': 10}, {'max_depth': 20, 'max_features': 'sqrt', 'min_samples_leaf': 5, 'mi
n_samples_split': 20}, {'max_depth': 20, 'max_features': 'sqrt', 'min_samples_leaf': 10, 'min_samples_split': 2},
{'max_depth': 20, 'max_features': 'sqrt', 'min_samples_leaf': 10, 'min_samples_split': 5}, {'max_depth': 20, 'max
_features': 'sqrt', 'min_samples_leaf': 10, 'min_samples_split': 10}, {'max_depth': 20, 'max_features': 'sqrt',
'min_samples_leaf': 10, 'min_samples_split': 20}, {'max_depth': 20, 'max_features': 'log2', 'min_samples_leaf':
1, 'min_samples_split': 2}, {'max_depth': 20, 'max_features': 'log2', 'min_samples_leaf': 1, 'min_samples_split':
5}, {'max_depth': 20, 'max_features': 'log2', 'min_samples_leaf': 1, 'min_samples_split': 10}, {'max_depth': 20,
'max_features': 'log2', 'min_samples_leaf': 1, 'min_samples_split': 20}, {'max_depth': 20, 'max_features': 'log
2', 'min_samples_leaf': 2, 'min_samples_split': 2}, {'max_depth': 20, 'max_features': 'log2', 'min_samples_leaf':
2, 'min_samples_split': 5}, {'max_depth': 20, 'max_features': 'log2', 'min_samples_leaf': 2, 'min_samples_split':
10}, {'max_depth': 20, 'max_features': 'log2', 'min_samples_leaf': 2, 'min_samples_split': 20}, {'max_depth': 20,
'max_features': 'log2', 'min_samples_leaf': 5, 'min_samples_split': 2}, {'max_depth': 20, 'max_features': 'log2',
'min_samples_leaf': 5, 'min_samples_split': 5}, {'max_depth': 20, 'max_features': 'log2', 'min_samples_leaf': 5,
'min_samples_split': 10}, {'max_depth': 20, 'max_features': 'log2', 'min_samples_leaf': 5, 'min_samples_split': 2
0}, {'max_depth': 20, 'max_features': 'log2', 'min_samples_leaf': 10, 'min_samples_split': 2}, {'max_depth': 20,
'max_features': 'log2', 'min_samples_leaf': 10, 'min_samples_split': 5}, {'max_depth': 20, 'max_features': 'log
2', 'min_samples_leaf': 10, 'min_samples_split': 10}, {'max_depth': 20, 'max_features': 'log2', 'min_samples_lea
f': 10, 'min_samples_split': 20}], 'split0_test_score': array([0.8707552 , 0.87886467, 0.87734415, 0.88139888, 0.
87987836,
       0.87531678, 0.87582362, 0.88190573, 0.87582362, 0.87531678,
       0.87379625, 0.87835783, 0.88443994, 0.88443994, 0.88443994,
       0.88443994, 0.86568677, 0.86568677, 0.8803852 , 0.88190573,
       0.87126204, 0.8839331 , 0.87734415, 0.8839331 , 0.88241257,
       0.88545362, 0.89153573, 0.88291941, 0.88241257, 0.89508363,
       0.88798784, 0.88849468, 0.86568677, 0.8636594 , 0.87633046,
       0.90217942, 0.8768373 , 0.86670046, 0.87126204, 0.88596047,
       0.88494678, 0.88241257, 0.88646731, 0.89457679, 0.90116574,
       0.88950836, 0.89254942, 0.88443994, 0.89964521, 0.89913837,
       0.89913837, 0.89913837, 0.89913837, 0.89964521, 0.89913837,
       0.89913837, 0.89913837, 0.89913837, 0.89913837, 0.89913837,
       0.89863153, 0.89863153, 0.89863153, 0.89863153, 0.90724785,
       0.897111 , 0.89913837, 0.89913837, 0.89812468, 0.90268626,
       0.897111 , 0.86822098, 0.89913837, 0.89001521, 0.89964521,
       0.89406994, 0.87785099, 0.8672073 , 0.89508363, 0.90167258,
       0.89001521, 0.86822098, 0.89406994, 0.89863153, 0.88190573,
       0.89660416, 0.89305626, 0.89508363, 0.89863153, 0.89457679,
       0.88798784, 0.90015205, 0.90116574, 0.90217942, 0.89508363,
       0.89102889, 0.88748099, 0.88798784, 0.88950836, 0.88748099,
       0.89254942, 0.89153573, 0.88950836, 0.88849468, 0.8839331 ,
       0.88291941, 0.88697415, 0.88545362, 0.88443994, 0.88494678,
       0.8839331 , 0.8839331 , 0.89761784, 0.89102889, 0.89204257,
       0.89204257, 0.88342625, 0.89609731, 0.8803852 , 0.87987836,
       0.88900152, 0.89254942, 0.87886467, 0.8935631 , 0.89863153,
       0.89660416, 0.8935631 , 0.89204257, 0.89457679, 0.88190573,
       0.89305626, 0.88494678, 0.8935631 , 0.89508363, 0.87987836,
       0.89001521, 0.86011151, 0.89254942, 0.87126204, 0.89204257,
       0.89254942, 0.89254942, 0.89964521, 0.89406994, 0.87379625,
       0.8803852 , 0.87582362, 0.88139888, 0.8803852 , 0.87785099,
       0.87734415, 0.88241257, 0.87531678, 0.87886467, 0.87785099,
       0.87987836, 0.88443994, 0.8839331 , 0.88494678, 0.88443994,
       0.87886467, 0.88139888, 0.88190573, 0.88291941, 0.87531678,
       0.88190573, 0.88443994, 0.89457679, 0.89102889, 0.89913837,
       0.89052205, 0.87633046, 0.89305626, 0.89508363, 0.89660416,
       0.89913837, 0.87126204, 0.88545362, 0.88697415, 0.88697415,
       0.87278256, 0.87024835, 0.88291941, 0.88596047, 0.88089204,
       0.87886467, 0.87734415, 0.89153573, 0.89305626, 0.897111 ,
       0.88342625, 0.89406994, 0.86619361, 0.87886467, 0.87227572,
       0.88139888, 0.87886467, 0.87480993, 0.8768373 , 0.88241257,
       0.87785099, 0.8768373 , 0.87531678, 0.87987836, 0.88443994,
       0.8839331 , 0.88494678, 0.88494678, 0.86568677, 0.87278256,
       0.87531678, 0.88089204, 0.87886467, 0.87987836, 0.87937152,
       0.88494678, 0.88443994, 0.88596047, 0.89406994, 0.89559047,
       0.88697415, 0.89508363, 0.88139888, 0.89406994, 0.86213887,
       0.8803852 , 0.88190573, 0.87633046, 0.87785099, 0.87937152,
       0.89052205, 0.88646731, 0.88494678, 0.88596047, 0.87582362,
       0.89204257, 0.89001521, 0.90167258, 0.88798784, 0.89609731]), 'split1_test_score': array([0.85859098, 0.86
416624, 0.86568677, 0.87328941, 0.87024835,
       0.8672073 , 0.86923467, 0.8768373 , 0.87480993, 0.87582362,
       0.87379625, 0.87785099, 0.89102889, 0.89102889, 0.89102889,
       0.89102889, 0.86670046, 0.88596047, 0.87227572, 0.8768373 ,
       0.87734415, 0.86670046, 0.8768373 , 0.88494678, 0.8935631 ,
       0.88545362, 0.87886467, 0.88900152, 0.88849468, 0.88545362,
       0.88697415, 0.8768373 , 0.85301571, 0.87227572, 0.88494678,
       0.88139888, 0.8768373 , 0.87480993, 0.87227572, 0.88697415,
       0.89406994, 0.87126204, 0.89254942, 0.87886467, 0.89204257,
       0.88900152, 0.88900152, 0.88849468, 0.89457679, 0.89457679,
       0.89457679, 0.89457679, 0.89457679, 0.89457679, 0.89457679,
       0.89457679, 0.89660416, 0.89660416, 0.89660416, 0.89660416,
       0.89913837, 0.89913837, 0.89913837, 0.89913837, 0.8839331 ,
       0.86213887, 0.89863153, 0.89559047, 0.87734415, 0.8707552 ,
       0.86670046, 0.87480993, 0.89812468, 0.88950836, 0.87785099,
       0.88089204, 0.88646731, 0.88494678, 0.88190573, 0.88342625,
       0.9006589 , 0.88443994, 0.89863153, 0.89457679, 0.89153573,

```
       0.85605677, 0.89102889, 0.89508363, 0.8935631 , 0.89964521,
       0.87024835, 0.897111  , 0.88241257, 0.88849468, 0.86923467,
       0.89052205, 0.87785099, 0.87835783, 0.87886467, 0.88190573,
       0.88190573, 0.88139888, 0.87937152, 0.88291941, 0.88494678,
       0.88342625, 0.88494678, 0.88291941, 0.89204257, 0.89204257,
       0.89305626, 0.89204257, 0.89001521, 0.87379625, 0.88646731,
       0.88950836, 0.88900152, 0.89153573, 0.88443994, 0.88900152,
       0.88849468, 0.89052205, 0.89001521, 0.89457679, 0.87582362,
       0.8839331 , 0.87937152, 0.89052205, 0.89052205, 0.88849468,
       0.88900152, 0.89254942, 0.89001521, 0.89964521, 0.89812468,
       0.8768373 , 0.89660416, 0.88241257, 0.88748099, 0.8935631 ,
       0.88900152, 0.8839331 , 0.9006589 , 0.89457679, 0.86315256,
       0.86974151, 0.86923467, 0.87734415, 0.86872783, 0.87126204,
       0.87176888, 0.87835783, 0.87734415, 0.87886467, 0.87734415,
       0.88190573, 0.89102889, 0.89001521, 0.89102889, 0.89001521,
       0.85504308, 0.87024835, 0.86872783, 0.8768373 , 0.88089204,
       0.87328941, 0.88190573, 0.88798784, 0.87987836, 0.87430309,
       0.88443994, 0.89457679, 0.87480993, 0.88900152, 0.89406994,
       0.88950836, 0.87582362, 0.87430309, 0.86974151, 0.88190573,
       0.87176888, 0.89102889, 0.88089204, 0.9006589 , 0.88443994,
       0.87937152, 0.87937152, 0.89254942, 0.8839331 , 0.88748099,
       0.88241257, 0.8935631 , 0.86467309, 0.86619361, 0.8672073 ,
       0.87531678, 0.86923467, 0.86974151, 0.87024835, 0.8768373 ,
       0.87480993, 0.87430309, 0.87328941, 0.87886467, 0.89204257,
       0.89204257, 0.89102889, 0.89102889, 0.87024835, 0.8636594 ,
       0.86467309, 0.88545362, 0.87126204, 0.88089204, 0.87734415,
       0.8839331 , 0.88748099, 0.8839331 , 0.88697415, 0.88950836,
       0.89660416, 0.89305626, 0.87633046, 0.89052205, 0.85250887,
       0.86923467, 0.87430309, 0.87835783, 0.87227572, 0.87633046,
       0.88342625, 0.89052205, 0.87278256, 0.88443994, 0.8768373 ,
       0.87176888, 0.88900152, 0.88798784, 0.88900152, 0.89254942]), 'split2_test_score': array([0.8672073 , 0.87
126204, 0.88139888, 0.8803852 , 0.88494678,
       0.88190573, 0.89305626, 0.88950836, 0.89913837, 0.89761784,
       0.89964521, 0.90268626, 0.90015205, 0.90015205, 0.89913837,
       0.89913837, 0.87430309, 0.86061835, 0.88190573, 0.89406994,
       0.86974151, 0.88241257, 0.87582362, 0.89508363, 0.89660416,
       0.88646731, 0.87835783, 0.89863153, 0.90522048, 0.89964521,
       0.89964521, 0.90319311, 0.87480993, 0.87734415, 0.88545362,
       0.88849468, 0.88190573, 0.88241257, 0.88596047, 0.89254942,
       0.89508363, 0.89153573, 0.90217942, 0.89153573, 0.89913837,
       0.90369995, 0.906741  , 0.89913837, 0.91180943, 0.91180943,
       0.91180943, 0.91180943, 0.91180943, 0.91180943, 0.91180943,
       0.91180943, 0.91231627, 0.91231627, 0.91231627, 0.91231627,
       0.91332995, 0.91332995, 0.91332995, 0.91332995, 0.87024835,
       0.89204257, 0.88494678, 0.89305626, 0.88545362, 0.90369995,
       0.90927522, 0.90572732, 0.90319311, 0.88900152, 0.87987836,
       0.8803852 , 0.87987836, 0.88646731, 0.897111  , 0.90319311,
       0.90369995, 0.89559047, 0.89305626, 0.91535732, 0.8803852 ,
       0.87734415, 0.89761784, 0.90826153, 0.90775469, 0.89964521,
       0.85909782, 0.86923467, 0.90116574, 0.90927522, 0.89305626,
       0.91332995, 0.89305626, 0.89305626, 0.8935631 , 0.89305626,
       0.89559047, 0.89660416, 0.897111  , 0.897111  , 0.90522048,
       0.89964521, 0.89913837, 0.9006589 , 0.897111  , 0.89863153,
       0.89660416, 0.89660416, 0.89508363, 0.88443994, 0.90420679,
       0.89964521, 0.897111  , 0.90319311, 0.8935631 , 0.88849468,
       0.90116574, 0.90268626, 0.90319311, 0.90420679, 0.91130258,
       0.89964521, 0.89001521, 0.90724785, 0.88849468, 0.90116574,
       0.89761784, 0.89660416, 0.88900152, 0.90217942, 0.88798784,
       0.897111  , 0.90167258, 0.90522048, 0.90116574, 0.90319311,
       0.90167258, 0.90319311, 0.90420679, 0.89102889, 0.87328941,
       0.87785099, 0.88342625, 0.88241257, 0.88798784, 0.89052205,
       0.89102889, 0.89001521, 0.90268626, 0.89609731, 0.89863153,
       0.90268626, 0.9006589 , 0.9006589 , 0.89863153, 0.89913837,
       0.88190573, 0.88545362, 0.87785099, 0.89102889, 0.87176888,
       0.8803852 , 0.88190573, 0.8935631 , 0.88291941, 0.89406994,
       0.88596047, 0.88798784, 0.88596047, 0.90369995, 0.89609731,
       0.9006589 , 0.88291941, 0.8839331 , 0.88443994, 0.89559047,
       0.87734415, 0.88545362, 0.89102889, 0.8935631 , 0.89559047,
       0.88748099, 0.89254942, 0.88089204, 0.90015205, 0.90116574,
       0.90623416, 0.89863153, 0.86619361, 0.87227572, 0.8803852 ,
       0.87987836, 0.88596047, 0.88443994, 0.88950836, 0.88849468,
       0.90167258, 0.89913837, 0.897111  , 0.90268626, 0.90015205,
       0.9006589 , 0.90015205, 0.9006589 , 0.86517993, 0.86670046,
       0.8803852 , 0.9006589 , 0.8935631 , 0.87987836, 0.88900152,
       0.88697415, 0.88900152, 0.89305626, 0.89660416, 0.89153573,
       0.89102889, 0.90724785, 0.89153573, 0.90623416, 0.86517993,
       0.8768373 , 0.88190573, 0.8839331 , 0.88596047, 0.89001521,
       0.88291941, 0.88342625, 0.89254942, 0.88545362, 0.88900152,
       0.8935631 , 0.91130258, 0.86923467, 0.90471363, 0.89609731]), 'split3_test_score': array([0.84591992, 0.84
591992, 0.8540294 , 0.86568677, 0.85554992,
       0.85605677, 0.86163203, 0.86974151, 0.86974151, 0.86974151,
       0.87278256, 0.87227572, 0.87126204, 0.87328941, 0.87278256,
       0.87278256, 0.86163203, 0.85909782, 0.85757729, 0.86923467,
       0.86315256, 0.8672073 , 0.87227572, 0.86315256, 0.88291941,
```

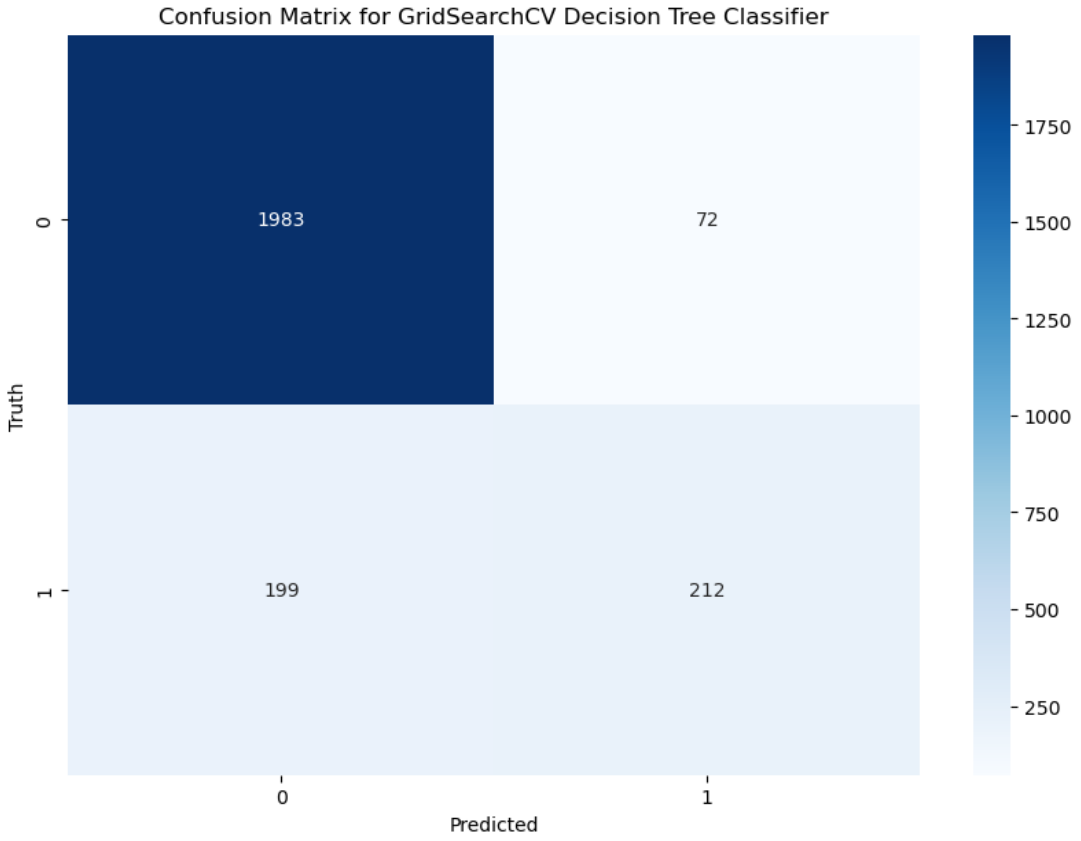
0.86213887, 0.87886467, 0.8768373 , 0.88494678, 0.87480993,
0.89153573, 0.88139888, 0.85707045, 0.86771414, 0.86213887,
0.87835783, 0.87430309, 0.87937152, 0.87379625, 0.86416624,
0.87328941, 0.8839331 , 0.8707552 , 0.88748099, 0.87886467,
0.89153573, 0.88291941, 0.88139888, 0.89609731, 0.89609731,
0.89609731, 0.89609731, 0.89609731, 0.89609731, 0.89609731,
0.89609731, 0.89609731, 0.89609731, 0.89609731, 0.89609731,
0.89609731, 0.89609731, 0.89609731, 0.89609731, 0.88748099,
0.87582362, 0.89863153, 0.88900152, 0.88443994, 0.88342625,
0.89001521, 0.87328941, 0.85149518, 0.87785099, 0.86670046,
0.87379625, 0.87987836, 0.88089204, 0.89305626, 0.897111 ,
0.88900152, 0.87886467, 0.88241257, 0.87531678, 0.87937152,
0.89305626, 0.89863153, 0.89559047, 0.87126204, 0.88342625,
0.86517993, 0.88798784, 0.89153573, 0.86619361, 0.86467309,
0.87937152, 0.87734415, 0.87734415, 0.87176888, 0.87582362,
0.87886467, 0.87480993, 0.87430309, 0.87480993, 0.87785099,
0.87937152, 0.87785099, 0.87785099, 0.87937152, 0.87734415,
0.87734415, 0.87734415, 0.8803852 , 0.87582362, 0.8768373 ,
0.88443994, 0.88139888, 0.89204257, 0.87126204, 0.88190573,
0.87886467, 0.87480993, 0.88494678, 0.89305626, 0.89508363,
0.87937152, 0.88697415, 0.87987836, 0.87886467, 0.88545362,
0.88646731, 0.88494678, 0.88545362, 0.87176888, 0.87937152,
0.88443994, 0.88139888, 0.88089204, 0.88241257, 0.88646731,
0.8768373 , 0.89254942, 0.86771414, 0.87937152, 0.85757729,
0.86163203, 0.8672073 , 0.86568677, 0.8636594 , 0.87024835,
0.86517993, 0.87024835, 0.87430309, 0.87278256, 0.87328941,
0.87328941, 0.87328941, 0.87278256, 0.87278256, 0.87278256,
0.86872783, 0.8672073 , 0.86670046, 0.87835783, 0.87785099,
0.86670046, 0.8803852 , 0.87582362, 0.88494678, 0.87328941,
0.87734415, 0.8839331 , 0.88748099, 0.89052205, 0.88900152,
0.88291941, 0.86416624, 0.86517993, 0.86416624, 0.86619361,
0.88089204, 0.86568677, 0.87328941, 0.87633046, 0.8707552 ,
0.87582362, 0.87126204, 0.87379625, 0.89204257, 0.8768373 ,
0.88494678, 0.87987836, 0.84845413, 0.84997466, 0.85352255,
0.86264572, 0.86061835, 0.86163203, 0.86011151, 0.8672073 ,
0.86771414, 0.87126204, 0.86923467, 0.87227572, 0.87328941,
0.87278256, 0.87328941, 0.87126204, 0.85301571, 0.8636594 ,
0.87379625, 0.86467309, 0.87430309, 0.86974151, 0.86872783,
0.86923467, 0.86163203, 0.86670046, 0.87024835, 0.87937152,
0.89153573, 0.8839331 , 0.88291941, 0.88494678, 0.86011151,
0.86517993, 0.86670046, 0.87227572, 0.85960466, 0.87126204,
0.86923467, 0.88342625, 0.87024835, 0.86061835, 0.87024835,
0.88646731, 0.87582362, 0.86923467, 0.89761784, 0.88089204]), 'split4_test_score': array([0.84787018, 0.84
939148, 0.85547667, 0.87423935, 0.85649087,
0.85496957, 0.86206897, 0.87829615, 0.87170385, 0.86916836,
0.86866126, 0.87829615, 0.88640974, 0.88742394, 0.88742394,
0.88742394, 0.87525355, 0.88032454, 0.88133874, 0.88235294,
0.87373225, 0.86511156, 0.87728195, 0.88843813, 0.89604462,
0.88488844, 0.87068966, 0.88590264, 0.89503043, 0.89655172,
0.89350913, 0.89959432, 0.86206897, 0.87423935, 0.87525355,
0.87474645, 0.87525355, 0.87474645, 0.88945233, 0.89300203,
0.88539554, 0.88742394, 0.89300203, 0.89300203, 0.89198783,
0.89350913, 0.90010142, 0.89503043, 0.90010142, 0.90010142,
0.90010142, 0.90010142, 0.90010142, 0.90010142, 0.90010142,
0.90010142, 0.90162272, 0.90162272, 0.90263692, 0.90162272,
0.90466531, 0.90466531, 0.90466531, 0.90466531, 0.89452333,
0.88640974, 0.89858012, 0.86004057, 0.90212982, 0.86308316,
0.88438134, 0.86460446, 0.86054767, 0.86511156, 0.89097363,
0.89807302, 0.88995943, 0.89807302, 0.86815416, 0.89807302,
0.87170385, 0.89198783, 0.89655172, 0.89807302, 0.90212982,
0.90212982, 0.89756592, 0.87677485, 0.86206897, 0.89705882,
0.90466531, 0.87373225, 0.88184584, 0.86663286, 0.89807302,
0.89097363, 0.87525355, 0.88032454, 0.87677485, 0.88438134,
0.87829615, 0.87728195, 0.87880325, 0.88742394, 0.88286004,
0.88133874, 0.88438134, 0.88894523, 0.89198783, 0.89198783,
0.89097363, 0.89249493, 0.88438134, 0.88235294, 0.89249493,
0.89858012, 0.90314402, 0.87981744, 0.89198783, 0.89807302,
0.88691684, 0.89705882, 0.89452333, 0.89553753, 0.89756592,
0.89503043, 0.88438134, 0.90212982, 0.89452333, 0.88488844,
0.89655172, 0.90010142, 0.88488844, 0.89198783, 0.88539554,
0.89807302, 0.89908722, 0.88640974, 0.89908722, 0.89655172,
0.89503043, 0.90415822, 0.88691684, 0.89756592, 0.85649087,
0.85649087, 0.85801217, 0.87626775, 0.86409736, 0.86004057,
0.86460446, 0.88133874, 0.87068966, 0.86967546, 0.87119675,
0.87880325, 0.88843813, 0.88742394, 0.88640974, 0.88640974,
0.87170385, 0.87576065, 0.88184584, 0.88488844, 0.88894523,
0.87373225, 0.88691684, 0.89705882, 0.89300203, 0.88235294,
0.88032454, 0.88793103, 0.89553753, 0.89350913, 0.89350913,
0.89655172, 0.87322515, 0.87931034, 0.87626775, 0.88691684,
0.87576065, 0.87778905, 0.88488844, 0.87474645, 0.88235294,
0.87880325, 0.89198783, 0.88894523, 0.89908722, 0.89249493,
0.89148073, 0.89807302, 0.84787018, 0.84888438, 0.85598377,
0.87474645, 0.85496957, 0.85953347, 0.86156187, 0.87931034,
0.86967546, 0.86967546, 0.86866126, 0.87778905, 0.88590264,

```
       0.88640974, 0.88742394, 0.88640974, 0.86764706, 0.87728195,
       0.87728195, 0.87525355, 0.87373225, 0.87728195, 0.88488844,
       0.88235294, 0.88995943, 0.88894523, 0.89503043, 0.90212982,
       0.89807302, 0.89148073, 0.90263692, 0.88793103, 0.87271805,
       0.86916836, 0.87423935, 0.88438134, 0.87423935, 0.88235294,
       0.88793103, 0.88590264, 0.88742394, 0.89046653, 0.88235294,
       0.89249493, 0.89148073, 0.90212982, 0.89503043, 0.89655172]), 'mean_test_score': array([0.85806871, 0.8619
2087, 0.86678717, 0.87499992, 0.86942286,
       0.86709123, 0.87236311, 0.87925781, 0.87824346, 0.87753362,
       0.87773631, 0.88189339, 0.88665853, 0.88726684, 0.88696274,
       0.88696274, 0.86871518, 0.87033759, 0.87469654, 0.88088012,
       0.8710465 , 0.873073  , 0.87591255, 0.88311084, 0.89030877,
       0.88088037, 0.87966251, 0.88665848, 0.89122099, 0.89030882,
       0.89193041, 0.88990366, 0.86253037, 0.87104655, 0.87682466,
       0.88503545, 0.8770274 , 0.87560819, 0.87854936, 0.88453046,
       0.88655706, 0.88331347, 0.88899067, 0.88909204, 0.89263984,
       0.89345094, 0.89426255, 0.88970046, 0.90044603, 0.90034466,
       0.90034466, 0.90034466, 0.90034466, 0.90044603, 0.90034466,
       0.90034466, 0.90115577, 0.90115577, 0.9013586 , 0.90115577,
       0.9023725 , 0.9023725 , 0.9023725 , 0.9023725 , 0.88868672,
       0.88270516, 0.89598566, 0.88736544, 0.88949844, 0.88473017,
       0.88949664, 0.87733042, 0.8824998 , 0.88229753, 0.88300973,
       0.88544329, 0.88280689, 0.88351729, 0.88706215, 0.89669519,
       0.89101588, 0.88382078, 0.8929444 , 0.89639109, 0.8870656 ,
       0.88503823, 0.89558009, 0.89415882, 0.88665606, 0.89487046,
       0.87743585, 0.88564356, 0.89162512, 0.88655516, 0.88402413,
       0.89304521, 0.88219719, 0.88341412, 0.88209597, 0.88452959,
       0.88544129, 0.88432613, 0.88381944, 0.88615179, 0.88696228,
       0.88534023, 0.88665833, 0.88716563, 0.88899057, 0.88899057,
       0.88838226, 0.88848378, 0.88949664, 0.88148833, 0.89040978,
       0.89284324, 0.89081633, 0.89253723, 0.88432762, 0.88747066,
       0.88888869, 0.8915253 , 0.89030862, 0.89618809, 0.89568146,
       0.89091688, 0.88686106, 0.89436413, 0.8893963 , 0.88838164,
       0.89253893, 0.89182971, 0.88858438, 0.89213299, 0.88615159,
       0.88929529, 0.88777487, 0.88949685, 0.88828171, 0.89436356,
       0.89101825, 0.89527665, 0.89182837, 0.89132261, 0.86486128,
       0.86922012, 0.8707408 , 0.87662202, 0.87297153, 0.8739848 ,
       0.87398526, 0.88047454, 0.88006799, 0.87925694, 0.87966256,
       0.8833126 , 0.88757105, 0.88696274, 0.8867599 , 0.88655716,
       0.87124903, 0.87601376, 0.87540617, 0.88280637, 0.87895478,
       0.87520261, 0.88311069, 0.88980203, 0.88635509, 0.88463075,
       0.88371823, 0.88615184, 0.88736904, 0.89436325, 0.89385641,
       0.89375535, 0.87347929, 0.87763602, 0.87631792, 0.88351616,
       0.87570966, 0.87804134, 0.88260364, 0.88625187, 0.88280612,
       0.88006881, 0.88250299, 0.88554373, 0.89365424, 0.89101799,
       0.8897001 , 0.89284319, 0.85867693, 0.86323861, 0.86587491,
       0.87479724, 0.86992955, 0.87003138, 0.87165348, 0.87885244,
       0.87834462, 0.87824325, 0.87672262, 0.88229881, 0.88716532,
       0.88716537, 0.88736821, 0.88686127, 0.86435556, 0.86881675,
       0.87429065, 0.88138624, 0.87834503, 0.87753444, 0.87986669,
       0.88148833, 0.88250278, 0.8837191 , 0.88858541, 0.89162718,
       0.89284319, 0.89416031, 0.88696428, 0.89274079, 0.86253145,
       0.87216109, 0.87581087, 0.87905569, 0.87398624, 0.87986643,
       0.88280668, 0.8859489 , 0.88159021, 0.88138778, 0.87885275,
       0.88726736, 0.89152473, 0.88605191, 0.89487025, 0.89243756]), 'std_test_score': array([0.0099633 , 0.01258
894, 0.01110721, 0.00566016, 0.01192213,
       0.01054359, 0.01158529, 0.00647192, 0.01067044, 0.01041066,
       0.01111705, 0.01064542, 0.009415  , 0.00875861, 0.00863057,
       0.00863057, 0.00524169, 0.01082811, 0.00924533, 0.00811361,
       0.00471036, 0.00828939, 0.00189812, 0.01071548, 0.00632574,
       0.00938458, 0.00670016, 0.00720904, 0.00818394, 0.00908547,
       0.00452271, 0.01015445, 0.00750071, 0.00483817, 0.00847128,
       0.00969083, 0.00262445, 0.00531394, 0.00760072, 0.01057156,
       0.00786281, 0.00679609, 0.01040768, 0.00563093, 0.00781612,
       0.00536759, 0.00835565, 0.00656545, 0.00605386, 0.00607064,
       0.00607064, 0.00607064, 0.00607064, 0.00605386, 0.00607064,
       0.00607064, 0.00591861, 0.00591861, 0.00594844, 0.00591861,
       0.0061507 , 0.0061507 , 0.0061507 , 0.0061507 , 0.01218692,
       0.01248012, 0.0055232 , 0.01405677, 0.00920478, 0.01641869,
       0.01410866, 0.01465605, 0.02187407, 0.00971217, 0.01133643,
       0.0091201 , 0.00461023, 0.00995513, 0.01081992, 0.0070019 ,
       0.01124177, 0.00973166, 0.00561347, 0.01276703, 0.0086875 ,
       0.01666917, 0.00298296, 0.01004837, 0.0171917 , 0.00602398,
       0.01667896, 0.01231767, 0.0085154 , 0.01775146, 0.01410287,
       0.01107397, 0.00687741, 0.00609665, 0.00814669, 0.00572914,
       0.00721577, 0.00838422, 0.00830104, 0.00729791, 0.00944927,
       0.007291  , 0.00695032, 0.00765453, 0.00628449, 0.00725586,
       0.00689632, 0.00692091, 0.0064284 , 0.00619256, 0.00891209,
       0.0056843 , 0.00822423, 0.00760608, 0.00812696, 0.00639396,
       0.00714511, 0.00934501, 0.00828943, 0.00409922, 0.01141021,
       0.00784243, 0.00483986, 0.00955285, 0.00576401, 0.00672546,
       0.00427886, 0.00610711, 0.00317712, 0.01077765, 0.00682087,
       0.00796779, 0.01552409, 0.00883528, 0.01102283, 0.00549777,
       0.00821125, 0.00755122, 0.01340094, 0.00632505, 0.00744187,
```

```
       0.00915462, 0.00852676, 0.00594265, 0.00963252, 0.01004235,
       0.00971816, 0.0063966 , 0.01151318, 0.00913924, 0.00980577,
       0.01009862, 0.0089193 , 0.00902631, 0.00846092, 0.00853802,
       0.00938892, 0.0067671 , 0.00648198, 0.00504731, 0.00582499,
       0.0054754 , 0.0023059 , 0.00759332, 0.0049348 , 0.01039273,
       0.00456041, 0.00598161, 0.00719439, 0.00513664, 0.00269473,
       0.00663189, 0.00610451, 0.00734621, 0.00860741, 0.00971571,
       0.00327502, 0.00934931, 0.00576176, 0.00991877, 0.00793981,
       0.00391152, 0.00841037, 0.00715815, 0.00581712, 0.00843665,
       0.00884996, 0.00679656, 0.00860519, 0.01197123, 0.01003703,
       0.0065929 , 0.01138252, 0.00906951, 0.01079538, 0.00700753,
       0.01220732, 0.01073547, 0.0104929 , 0.01052711, 0.008877  ,
       0.00920593, 0.00872819, 0.00953885, 0.00594311, 0.00538632,
       0.00528935, 0.01187571, 0.00799554, 0.00407566, 0.00691875,
       0.00630764, 0.0106015 , 0.00904653, 0.00974436, 0.00749283,
       0.0040235 , 0.00755157, 0.00924059, 0.00738432, 0.00659157,
       0.0055814 , 0.00569275, 0.00460507, 0.00858065, 0.00626041,
       0.00735367, 0.00260431, 0.00862108, 0.01058819, 0.00636431,
       0.00813154, 0.01136957, 0.01464166, 0.00610372, 0.00595022]), 'rank_test_score': array([240, 238, 231, 20
6, 226, 230, 216, 176, 185, 191, 188, 161, 108,
        94, 101, 101, 229, 223, 208, 168, 221, 214, 200, 145,  63, 167,
       175, 109,  55,  62,  47,  65, 237, 220, 195, 128, 194, 203, 182,
       131, 113, 143,  76,  75,  42,  35,  29,  67,   9,  11,  11,  11,
        11,   9,  11,  11,   6,   6,   5,   6,   1,   1,   1,   1,  80,
       152,  20,  92,  69, 129,  71, 193, 156, 158, 147, 124, 148, 140,
        99,  17,  58, 136,  37,  18,  98, 127,  22,  31, 111,  24, 192,
       122,  51, 114, 135,  36, 159, 142, 160, 132, 125, 134, 137, 118,
       104, 126, 110,  95,  78,  77,  84,  83,  71, 163,  61,  38,  60,
        44, 133,  89,  79,  52,  64,  19,  21,  59, 106,  26,  73,  85,
        43,  48,  82,  46, 119,  74,  87,  70,  86,  27,  56,  23,  49,
        54, 233, 227, 222, 197, 215, 212, 211, 169, 171, 177, 174, 144,
        88, 101, 107, 112, 219, 199, 204, 150, 179, 205, 146,  66, 115,
       130, 139, 117,  90,  28,  32,  33, 213, 189, 198, 141, 202, 187,
       153, 116, 151, 170, 154, 123,  34,  57,  68,  39, 239, 235, 232,
       207, 225, 224, 218, 181, 184, 186, 196, 157,  97,  96,  91, 105,
       234, 228, 209, 166, 183, 190, 172, 163, 155, 138,  81,  50,  39,
        30, 100,  41, 236, 217, 201, 178, 210, 173, 149, 121, 162, 165,
       180,  93,  53, 120,  25,  45], dtype=int32)}
Best parameters:  {'max_depth': 5, 'max_features': None, 'min_samples_leaf': 10, 'min_samples_split': 2}
              precision    recall  f1-score   support

           0       0.91      0.96      0.94      2055
           1       0.75      0.52      0.61       411

    accuracy                           0.89      2466
   macro avg       0.83      0.74      0.77      2466
weighted avg       0.88      0.89      0.88      2466

[[1983   72]
 [ 199  212]]
```

In [21]:

```
cm_grid = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(10,7))
sns.heatmap(cm_grid, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Truth')
plt.title('Confusion Matrix for GridSearchCV Decision Tree Classifier')
plt.show()

```

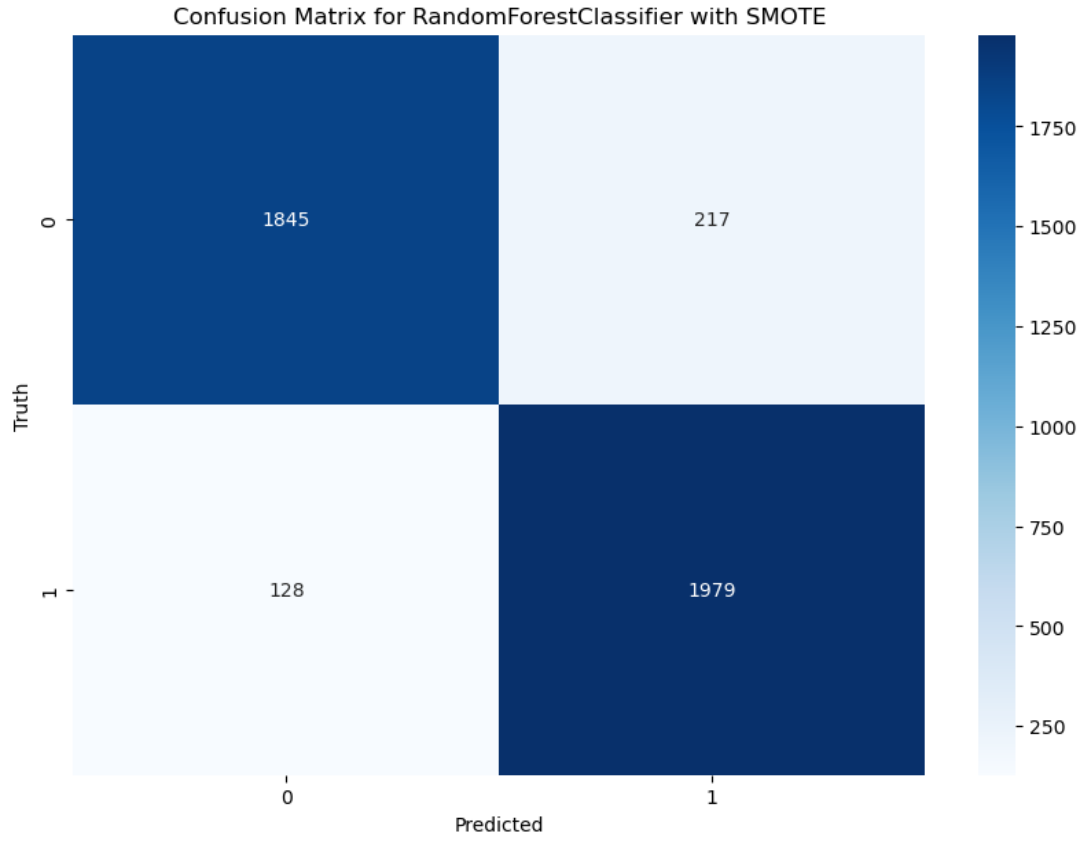


In [22]:

```
#this is inspired by the original paper Sakar 2018
from imblearn.over_sampling import SMOTE
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

# Oversampling the minority class
smote = SMOTE(random_state=42)
X_res, y_res = smote.fit_resample(X, y)

# Splitting the resampled data into training and test sets
X_train_res, X_test_res, y_train_res, y_test_res = train_test_split(X_res, y_res, test_size=0.2, random_state=42

# Creating the classifier
rfc = RandomForestClassifier(n_estimators=100, random_state=42)

rfc.fit(X_train_res, y_train_res)

y_pred_res = rfc.predict(X_test_res)

print(classification_report(y_test_res, y_pred_res))

print(confusion_matrix(y_test_res, y_pred_res))

print("Accuracy: ", accuracy_score(y_test_res, y_pred_res))

```

```
              precision    recall  f1-score   support

           0       0.94      0.89      0.91      2062
           1       0.90      0.94      0.92      2107

    accuracy                           0.92      4169
   macro avg       0.92      0.92      0.92      4169
weighted avg       0.92      0.92      0.92      4169

[[1845  217]
 [ 128 1979]]
Accuracy:  0.9172463420484529
```

In [23]:
```
cm_rfc = confusion_matrix(y_test_res, y_pred_res)

plt.figure(figsize=(10,7))
sns.heatmap(cm_rfc, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Truth')
plt.title('Confusion Matrix for RandomForestClassifier with SMOTE')
plt.show()

```

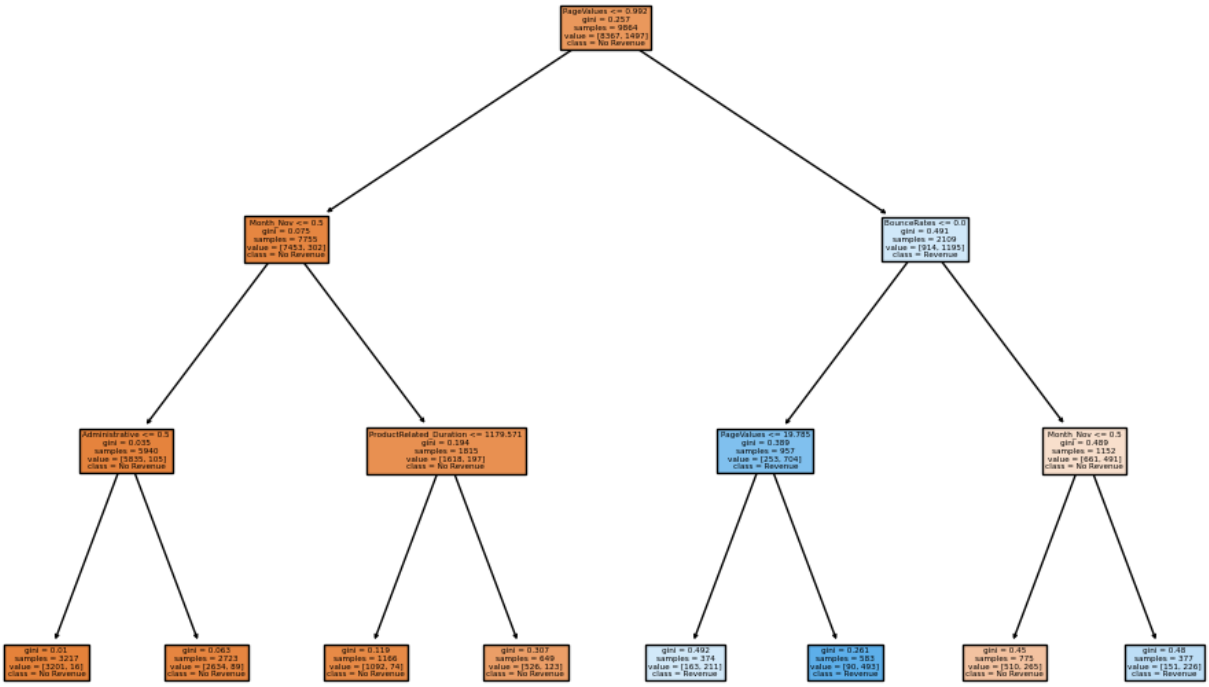


In [24]:
```
!pip install -U imbalanced-learn

```

```
Requirement already satisfied: imbalanced-learn in /Users/amayiyer/Desktop/DatSci_Python/anaconda3/lib/python3.9/
site-packages (0.10.1)
Requirement already satisfied: scikit-learn>=1.0.2 in /Users/amayiyer/Desktop/DatSci_Python/anaconda3/lib/python
3.9/site-packages (from imbalanced-learn) (1.2.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /Users/amayiyer/Desktop/DatSci_Python/anaconda3/lib/python
3.9/site-packages (from imbalanced-learn) (2.2.0)
Requirement already satisfied: numpy>=1.17.3 in /Users/amayiyer/Desktop/DatSci_Python/anaconda3/lib/python3.9/sit
e-packages (from imbalanced-learn) (1.21.5)
Requirement already satisfied: joblib>=1.1.1 in /Users/amayiyer/Desktop/DatSci_Python/anaconda3/lib/python3.9/sit
e-packages (from imbalanced-learn) (1.2.0)
Requirement already satisfied: scipy>=1.3.2 in /Users/amayiyer/Desktop/DatSci_Python/anaconda3/lib/python3.9/site
-packages (from imbalanced-learn) (1.9.1)
```

In [25]:

```
from sklearn.tree import DecisionTreeClassifier, plot_tree

dtree = DecisionTreeClassifier(max_depth=3)
dtree.fit(X_train, y_train)

plt.figure(figsize=(12,8))
plot_tree(dtree, feature_names=X_train.columns, class_names=['No Revenue', 'Revenue'], filled=True)
plt.show()

```

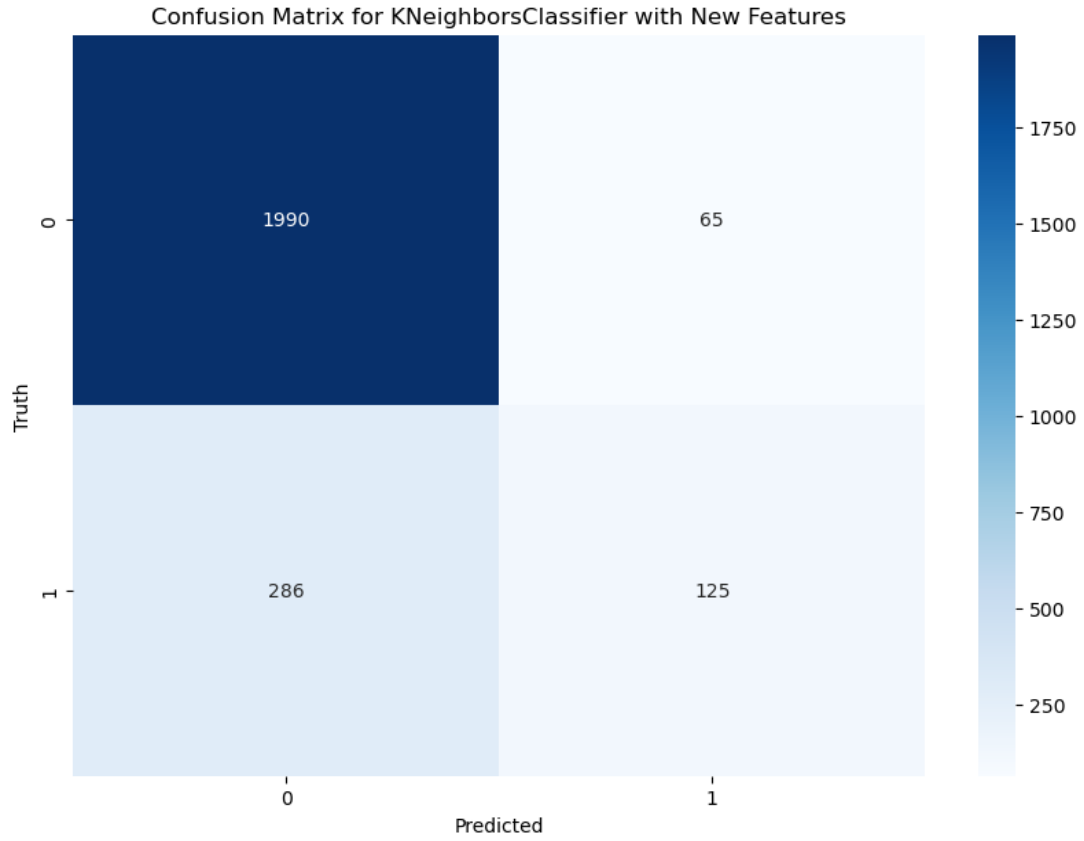


Task 3: Machine Learning Model (KNN) with ~ 12-13 features

In [26]:
```
rn.model_selection import train_test_split
rn.neighbors import KNeighborsClassifier
rn.metrics import classification_report, confusion_matrix

oded[['PageValues', 'ProductRelated', 'ProductRelated_Duration', 'Month_Nov', 'Administrative', 'TrafficType_2', 'Vi

oded['Revenue']

_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

ghborsClassifier(n_neighbors=5)

train, y_train)

n.predict(X_test)

sification_report(y_test, y_pred))

usion_matrix(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.87      0.97      0.92      2055
           1       0.66      0.30      0.42       411

    accuracy                           0.86      2466
   macro avg       0.77      0.64      0.67      2466
weighted avg       0.84      0.86      0.84      2466

[[1990   65]
 [ 286  125]]
```

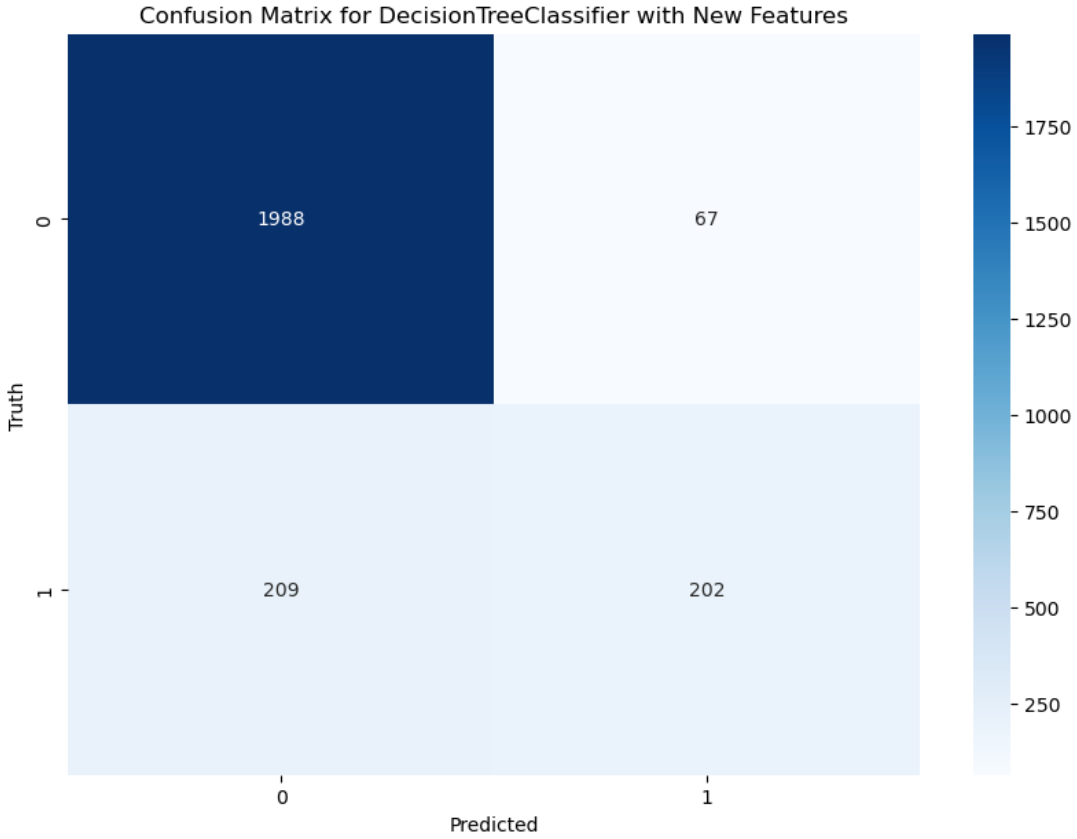
In [27]:
```
cm_knn = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(10,7))
sns.heatmap(cm_knn, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Truth')
plt.title('Confusion Matrix for KNeighborsClassifier with New Features')
plt.show()
```



In [28]:
```
from sklearn.tree import DecisionTreeClassifier

dtree = DecisionTreeClassifier(max_depth=5)

dtree.fit(X_train, y_train)

y_pred = dtree.predict(X_test)

print(classification_report(y_test, y_pred))

print(confusion_matrix(y_test, y_pred))

```

```
              precision    recall  f1-score   support

           0       0.90      0.97      0.94      2055
           1       0.75      0.49      0.59       411

    accuracy                           0.89      2466
   macro avg       0.83      0.73      0.76      2466
weighted avg       0.88      0.89      0.88      2466

[[1988   67]
 [ 209  202]]
```

In [29]:
```
cm_dtree = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(10,7))
sns.heatmap(cm_dtree, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Truth')
plt.title('Confusion Matrix for DecisionTreeClassifier with New Features')
plt.show()

```



In [30]:

```
from sklearn.model_selection import GridSearchCV

param_grid = {
    'max_depth': [None, 5, 10, 15, 20],
    'min_samples_split': [2, 5, 10, 20],
    'min_samples_leaf': [1, 2, 5, 10],
    'max_features': [None, 'sqrt', 'log2']
}

grid = GridSearchCV(DecisionTreeClassifier(), param_grid, cv=5, scoring='accuracy')

grid.fit(X_train, y_train)

print(grid.cv_results_)

print("Best parameters: ", grid.best_params_)

y_pred = grid.predict(X_test)

print(classification_report(y_test, y_pred))

print(confusion_matrix(y_test, y_pred))

```

{'mean_fit_time': array([0.04108605, 0.03782058, 0.04015455, 0.03715405, 0.03812332,
       0.04149613, 0.039218 , 0.03646545, 0.03478584, 0.03548536,
       0.03780618, 0.03594618, 0.03524165, 0.03481078, 0.03482342,
       0.0332262 , 0.01478891, 0.01345282, 0.01276879, 0.01275597,
       0.01331315, 0.01294918, 0.01273012, 0.01250877, 0.01252351,
       0.01217403, 0.01260781, 0.01208949, 0.01192646, 0.01173005,
       0.01158619, 0.01158419, 0.01425633, 0.01346068, 0.01294084,
       0.01255217, 0.01358347, 0.0130465 , 0.01309805, 0.01287141,
       0.01245122, 0.01190195, 0.01249723, 0.01298347, 0.01252508,
       0.01282606, 0.0119597 , 0.01187344, 0.01664395, 0.01478972,
       0.01569242, 0.01447964, 0.01481843, 0.01576777, 0.01530004,
       0.01484909, 0.01499467, 0.0150157 , 0.01604347, 0.01487026,
       0.01564293, 0.02018595, 0.01802397, 0.01414518, 0.00543776,
       0.0052217 , 0.00527949, 0.00532236, 0.00618658, 0.00560408,
       0.0056952 , 0.00521274, 0.00515723, 0.00531483, 0.00557513,
       0.0055234 , 0.00559983, 0.00517001, 0.0055006 , 0.00548205,
       0.00557399, 0.00556026, 0.00533128, 0.00515127, 0.00533543,
       0.00549288, 0.00568643, 0.00699649, 0.00762596, 0.00769825,
       0.00793281, 0.0079854 , 0.00758624, 0.00788441, 0.00732455,
       0.00573077, 0.02518005, 0.02681675, 0.0303997 , 0.02545171,
       0.02393646, 0.02483239, 0.0272747 , 0.02658272, 0.03174338,
       0.02922125, 0.02478132, 0.02433209, 0.02421474, 0.02768979,
       0.02789884, 0.02660871, 0.01080117, 0.00984778, 0.01031537,
       0.01083021, 0.01110272, 0.01093206, 0.01157637, 0.01087217,
       0.0103158 , 0.01065898, 0.01123071, 0.00877376, 0.01052785,
       0.01062503, 0.01035948, 0.01077542, 0.01170869, 0.01065249,
       0.0110302 , 0.01116438, 0.01102166, 0.01071963, 0.01049376,
       0.00795431, 0.00774007, 0.00799685, 0.00807648, 0.00770345,
       0.00754218, 0.00747366, 0.00747962, 0.00742064, 0.0283608 ,
       0.03063207, 0.03451138, 0.03059616, 0.03086252, 0.02850242,
       0.0297286 , 0.02920647, 0.02815418, 0.0277977 , 0.02698064,
       0.0260006 , 0.0262835 , 0.02983451, 0.02553539, 0.02555161,
       0.00979424, 0.01204758, 0.01274776, 0.01251569, 0.01242375,
       0.0121192 , 0.01158581, 0.01197777, 0.011689 , 0.0121222 ,
       0.01164412, 0.00904455, 0.00876279, 0.00841217, 0.00828638,
       0.00867677, 0.01266198, 0.0121954 , 0.01224079, 0.01187248,
       0.00914621, 0.00893159, 0.00913262, 0.00829053, 0.00834389,
       0.00843177, 0.00791297, 0.00877695, 0.00783339, 0.01061487,
       0.01215286, 0.01110716, 0.03654656, 0.03165436, 0.03108506,
       0.03092294, 0.03078346, 0.03513241, 0.03221416, 0.03455434,
       0.03401232, 0.02971263, 0.03396401, 0.03432279, 0.0330843 ,
       0.03274193, 0.03094296, 0.03342333, 0.01304517, 0.01242371,
       0.01326084, 0.01257381, 0.01311679, 0.01327596, 0.01253266,
       0.01210389, 0.01217527, 0.01204853, 0.01226683, 0.01251459,
       0.01183443, 0.01141582, 0.00981445, 0.01212721, 0.01351705,
       0.0121707 , 0.00938087, 0.00930223, 0.00974231, 0.00964885,
       0.01232424, 0.01230645, 0.01206422, 0.0113112 , 0.00883346,
       0.00831046, 0.00861354, 0.00856905, 0.01070681, 0.01299005]), 'std_fit_time': array([3.19323665e-04, 1.453
10789e-03, 9.81421608e-04, 2.30819376e-03,
       2.07674659e-03, 1.48447222e-03, 9.27771404e-04, 1.47616607e-03,
       1.80159874e-03, 2.12305985e-03, 1.02379438e-03, 2.08014655e-03,
       1.22383025e-03, 6.33402399e-04, 1.16373183e-03, 1.30869283e-03,
       5.97987398e-04, 5.02993873e-04, 5.11816077e-04, 6.20621131e-04,
       3.30436439e-04, 2.93262156e-04, 6.48761286e-04, 7.79084351e-04,
       1.11498392e-03, 7.30942692e-04, 2.28279981e-04, 5.43285918e-04,
       3.36778962e-04, 1.02163841e-03, 1.98832576e-04, 5.12903669e-04,
       3.21232761e-04, 6.92374834e-04, 3.98793041e-04, 4.91223481e-04,
       3.02202387e-04, 6.32060739e-04, 8.21185820e-04, 4.83492977e-04,
       1.01858129e-03, 4.04501037e-04, 3.14363351e-04, 7.15638322e-04,
       9.82216224e-04, 3.99391403e-04, 4.22504055e-04, 6.38767138e-04,
       1.85856109e-03, 9.07479619e-05, 1.50144316e-03, 1.13960689e-04,
       4.39145717e-04, 1.06568337e-03, 8.93124755e-04, 2.76828862e-04,
       1.97830445e-04, 1.82085287e-04, 1.50404753e-03, 1.01457972e-04,
       1.80276084e-03, 3.35892773e-04, 2.32862032e-03, 7.46642945e-05,
       3.29369053e-04, 3.30337957e-04, 3.68173915e-04, 3.53302085e-04,
       6.72829981e-04, 2.27647043e-04, 1.63814808e-04, 3.22052280e-04,
       2.54515245e-04, 3.79384571e-04, 3.82872369e-04, 7.63540329e-05,
       4.96193333e-04, 3.57358214e-04, 4.24526100e-04, 6.32279538e-04,
       5.16773048e-04, 5.89454199e-04, 3.24997424e-04, 2.45645845e-04,
       3.39210794e-04, 4.15376771e-04, 6.36801334e-04, 2.93005613e-04,
       3.91855847e-04, 5.70379235e-04, 2.54005143e-04, 7.35608037e-04,
       8.56083958e-04, 6.18799144e-04, 8.11825038e-04, 4.52935807e-04,
       1.03296373e-03, 1.36190143e-03, 4.90997710e-04, 1.38275994e-03,
       2.48330337e-04, 6.78406375e-04, 1.80497677e-03, 1.54521469e-03,
       2.56691111e-03, 3.00996288e-03, 6.53034775e-04, 5.29664534e-04,
       9.15839383e-04, 8.99511897e-04, 4.35254451e-04, 1.16016225e-03,
       8.78157500e-04, 7.56579663e-04, 6.57199201e-04, 7.22513449e-04,
       9.14520209e-04, 5.22232495e-04, 4.70738690e-04, 5.19778518e-04,
       6.70971756e-04, 5.76154250e-04, 4.47047742e-04, 9.48275848e-04,
       6.36125379e-04, 5.20183328e-04, 7.02563953e-04, 3.91087774e-04,
       6.07475557e-04, 7.94186357e-04, 2.78672831e-04, 7.12319553e-04,
       3.29757646e-04, 6.19032463e-04, 5.17745123e-04, 7.90626456e-04,
       6.23647067e-04, 7.06869287e-04, 4.84490380e-04, 3.16177542e-04,

```
       3.46039822e-04, 4.65316414e-04, 5.36345927e-04, 7.98278548e-04,
       9.45863728e-04, 1.51374068e-03, 1.24851822e-03, 5.25913786e-04,
       1.38865566e-03, 8.45615243e-04, 2.50709006e-03, 9.08502665e-04,
       1.53278461e-03, 5.09803998e-04, 1.55028186e-04, 7.88797549e-04,
       7.53940801e-04, 1.46737695e-03, 4.30679506e-04, 1.17201879e-03,
       5.75300464e-04, 7.84491899e-04, 5.14540128e-04, 8.78737253e-04,
       2.82900136e-04, 2.53326606e-04, 5.51459764e-04, 1.29425252e-03,
       8.75906684e-04, 1.01375720e-03, 8.56474749e-04, 6.51337699e-04,
       8.43392564e-04, 3.65113916e-04, 5.42963222e-04, 7.91866400e-04,
       7.62299379e-04, 5.01565453e-04, 3.53404301e-04, 8.42823974e-04,
       5.01456316e-04, 7.05304320e-04, 1.68064299e-04, 3.25411648e-04,
       6.22772703e-04, 4.62202505e-04, 3.26470683e-04, 2.70953559e-04,
       3.79761226e-04, 7.59714298e-04, 5.67051841e-04, 4.08259426e-04,
       2.56637800e-03, 1.46869948e-03, 9.02718803e-04, 2.18718337e-03,
       1.48311509e-03, 3.10472865e-03, 1.15662579e-03, 3.03139303e-03,
       3.34083042e-03, 1.72844326e-03, 2.46094074e-03, 1.40051180e-03,
       1.42586273e-03, 8.08489524e-04, 3.48661204e-03, 8.62628805e-04,
       9.66683323e-04, 2.98805789e-04, 5.40206499e-04, 7.67473289e-04,
       7.55946362e-04, 9.72120192e-04, 9.58659273e-04, 3.02050073e-04,
       2.14279042e-04, 7.19928094e-04, 5.53237691e-04, 7.13175785e-04,
       5.75710215e-04, 1.00795167e-03, 1.36676273e-03, 6.69903607e-04,
       1.83486331e-04, 1.53112700e-03, 3.59099330e-04, 7.17770900e-04,
       4.59823133e-04, 3.06280023e-04, 9.44536233e-04, 8.96463851e-04,
       1.05358539e-03, 1.18165260e-03, 2.35117876e-04, 2.95691547e-04,
       2.54336081e-04, 2.52957389e-04, 1.68034280e-03, 7.80772942e-04]), 'mean_score_time': array([0.00243616, 0.
00217419, 0.00248365, 0.00234199, 0.00227523,
       0.00262113, 0.00225701, 0.0022718 , 0.00207615, 0.00220385,
       0.00245113, 0.00226684, 0.00253143, 0.002455  , 0.00243258,
       0.00235558, 0.00244646, 0.00243182, 0.00236845, 0.00236683,
       0.0023881 , 0.00251174, 0.0023973 , 0.00233946, 0.00244122,
       0.00244493, 0.00248747, 0.00237813, 0.00235376, 0.00240273,
       0.0023809 , 0.00227566, 0.00231156, 0.00244622, 0.002316  ,
       0.00235386, 0.002319  , 0.00242352, 0.0024292 , 0.0023735 ,
       0.00234036, 0.00232811, 0.00263476, 0.00248895, 0.00259109,
       0.00271883, 0.00243258, 0.00233512, 0.00177345, 0.00169368,
       0.00169187, 0.00158877, 0.00165372, 0.00177398, 0.00168219,
       0.00184703, 0.00171957, 0.00167756, 0.00173721, 0.00161333,
       0.00180717, 0.00226831, 0.00177441, 0.00137725, 0.00135169,
       0.00136571, 0.00137963, 0.00153561, 0.00169635, 0.00142798,
       0.00142841, 0.00138464, 0.0013875 , 0.00141048, 0.00134854,
       0.00135145, 0.0013495 , 0.00135345, 0.00137181, 0.00135884,
       0.00137205, 0.00134258, 0.00135427, 0.00135298, 0.00134554,
       0.00149508, 0.0014627 , 0.00218816, 0.00211606, 0.00206437,
       0.00211086, 0.00227499, 0.00206146, 0.00224423, 0.0018939 ,
       0.00143743, 0.00170951, 0.00195141, 0.00223002, 0.00160189,
       0.00142097, 0.00168314, 0.00186739, 0.00185914, 0.00228143,
       0.0021512 , 0.00185928, 0.00155072, 0.00170112, 0.0021112 ,
       0.00203214, 0.00191092, 0.00215912, 0.00200696, 0.00216327,
       0.0022428 , 0.00223985, 0.00222235, 0.00218778, 0.002139  ,
       0.00219593, 0.00217586, 0.00220461, 0.00185337, 0.00222025,
       0.00235324, 0.00230494, 0.00226164, 0.00226417, 0.00222707,
       0.00233512, 0.00226498, 0.00233617, 0.00217643, 0.00203629,
       0.00147204, 0.00143194, 0.00146961, 0.00142651, 0.00139489,
       0.00139012, 0.00136976, 0.00137615, 0.00140762, 0.00160456,
       0.0018703 , 0.00198374, 0.00186601, 0.00185747, 0.00162129,
       0.00185218, 0.00178857, 0.00158057, 0.00176516, 0.00169253,
       0.00168018, 0.00185895, 0.00203633, 0.00156865, 0.00158811,
       0.00187559, 0.00219173, 0.00227079, 0.00238323, 0.0023973 ,
       0.0022646 , 0.0020452 , 0.00226097, 0.00232868, 0.00231919,
       0.00204363, 0.001579  , 0.00163321, 0.00147362, 0.00158672,
       0.00186844, 0.00224433, 0.00234599, 0.00230899, 0.00214553,
       0.00151377, 0.00148921, 0.00148234, 0.00145936, 0.0014545 ,
       0.00146418, 0.00142722, 0.0015707 , 0.00171351, 0.00206389,
       0.00228992, 0.00231462, 0.00205836, 0.00187902, 0.00184393,
       0.00183625, 0.00181055, 0.00213323, 0.00190463, 0.00215764,
       0.00212126, 0.00195699, 0.00213275, 0.00210776, 0.00222487,
       0.00215607, 0.00218325, 0.00228038, 0.00222969, 0.00216432,
       0.00229669, 0.00239072, 0.00229459, 0.00218477, 0.00240097,
       0.00240335, 0.00231376, 0.00236211, 0.00230451, 0.00244341,
       0.00225801, 0.00209718, 0.00203938, 0.00242062, 0.00240707,
       0.00196843, 0.001543  , 0.00150795, 0.00158095, 0.0019074 ,
       0.00228128, 0.00240197, 0.00229139, 0.00188503, 0.00148315,
       0.0015028 , 0.00145459, 0.00161052, 0.00222464, 0.00274615]), 'std_score_time': array([9.04229611e-05, 2.8
8554625e-04, 2.19138992e-04, 3.16249174e-04,
       3.47492768e-04, 2.96580807e-04, 2.31531816e-05, 2.30803696e-04,
       1.35692055e-04, 2.73102029e-04, 1.15083381e-04, 1.90056812e-04,
       1.53002147e-04, 2.07709084e-04, 1.53494318e-04, 2.43175590e-04,
       1.40751264e-04, 8.46549676e-05, 9.74946016e-05, 2.03284545e-04,
       4.06081086e-05, 4.65308537e-05, 1.02583400e-04, 8.07044038e-05,
       1.18600090e-04, 8.75424714e-05, 1.64123540e-04, 9.06480606e-05,
       1.00685676e-04, 1.31444329e-04, 1.45241452e-04, 7.63919916e-05,
       3.38475796e-05, 9.88932446e-05, 8.96016227e-05, 9.55062571e-05,
       3.32398939e-05, 6.15788621e-05, 1.98745556e-04, 1.05475840e-04,
       8.45618504e-05, 1.24978955e-04, 1.76865131e-04, 2.33643911e-04,
```

```
       1.44723762e-04, 3.07226466e-04, 1.40514695e-04, 8.57028274e-05,
       2.09026919e-04, 7.27365657e-05, 1.06448852e-04, 1.82432438e-05,
       5.12788206e-05, 2.00071439e-04, 1.00103509e-04, 1.49357242e-04,
       6.59254920e-05, 3.32261418e-05, 1.88988234e-04, 3.21284937e-05,
       2.38844177e-04, 3.51969137e-05, 3.50173009e-04, 4.84006393e-05,
       1.05472672e-05, 1.98792491e-05, 2.61863929e-05, 3.37418727e-04,
       2.30865771e-04, 2.17829527e-05, 1.67643359e-05, 1.70549127e-05,
       1.68483512e-05, 6.84931390e-05, 1.21872712e-05, 5.26597507e-06,
       1.02285739e-05, 2.97744239e-05, 1.21171068e-05, 1.73208103e-05,
       3.88645394e-05, 1.19498855e-05, 3.14171644e-05, 2.53156974e-05,
       7.20576523e-06, 9.99642674e-05, 1.44387047e-04, 1.98642623e-04,
       1.42855479e-04, 1.04335697e-04, 1.49074320e-04, 1.61842564e-04,
       1.17233749e-04, 1.84068346e-04, 2.29817237e-04, 4.47015444e-05,
       1.10767476e-04, 1.73456842e-04, 1.05331763e-04, 1.20270904e-04,
       9.31603628e-06, 1.25303388e-04, 8.85662093e-05, 1.12251802e-04,
       1.84335958e-04, 3.78325931e-04, 8.97037793e-05, 1.10662635e-04,
       9.82575003e-05, 1.16443400e-04, 4.38979302e-05, 8.25875901e-05,
       8.90898485e-05, 1.37741608e-04, 9.09985468e-05, 5.33047641e-05,
       1.11647793e-04, 1.09497781e-04, 1.49100232e-04, 1.05879068e-04,
       1.53230313e-04, 1.20954743e-04, 1.76474939e-04, 2.23721435e-04,
       1.09542126e-04, 1.23039086e-04, 9.91281950e-05, 7.07819750e-05,
       8.34285989e-05, 1.14841852e-04, 1.09058131e-04, 8.43199926e-05,
       1.24173340e-04, 1.13117956e-04, 1.00107916e-04, 7.26737061e-05,
       7.19534474e-06, 2.29244144e-05, 1.28643560e-05, 1.44045868e-05,
       1.66371758e-05, 1.07234305e-05, 6.89850058e-06, 4.52053195e-05,
       1.33709339e-04, 1.52496117e-04, 1.29829685e-04, 3.11712940e-05,
       4.74953268e-05, 1.27740341e-04, 2.08928724e-04, 2.95752088e-04,
       9.65879069e-05, 7.03276777e-05, 1.27993521e-04, 2.49070101e-04,
       9.93897255e-05, 1.55180058e-04, 1.47434826e-04, 1.32299499e-04,
       2.40742204e-04, 9.54009712e-05, 5.04115437e-05, 1.74244675e-04,
       7.73169065e-05, 1.27756110e-04, 1.46189314e-04, 8.68075989e-05,
       1.54404290e-04, 1.31491130e-04, 2.91157120e-04, 1.15499705e-04,
       1.78267942e-04, 1.16307220e-05, 3.74474766e-05, 2.41307419e-04,
       1.21326584e-04, 6.25370433e-05, 1.11466905e-04, 2.68559369e-04,
       1.36064963e-05, 3.57727673e-05, 2.73234765e-05, 7.61298561e-06,
       4.18457141e-05, 6.47499212e-05, 1.53785713e-05, 1.18562853e-04,
       2.08863820e-04, 5.85447683e-05, 2.78746294e-05, 1.34621053e-04,
       1.42125361e-04, 1.32301853e-04, 1.09600375e-04, 2.15462942e-04,
       5.14462878e-05, 1.67139729e-04, 8.84928833e-05, 1.47795937e-04,
       3.57105236e-04, 1.55941309e-04, 2.52964544e-04, 1.49453393e-04,
       1.85446342e-04, 9.30882619e-05, 3.77033528e-04, 5.98206611e-05,
       7.93910241e-05, 1.58526835e-04, 4.45986791e-05, 8.06951059e-05,
       4.93560781e-05, 1.08473128e-04, 1.46901340e-04, 1.52358882e-04,
       8.97780158e-05, 1.28152024e-04, 7.50568649e-05, 1.19652553e-04,
       7.27062373e-05, 2.23165159e-04, 2.80220463e-04, 1.55606541e-04,
       4.97414528e-05, 3.35439585e-04, 7.42384156e-05, 3.39075815e-05,
       9.62639630e-05, 2.82929969e-04, 1.19783032e-04, 6.78151630e-05,
       1.20413647e-04, 2.22778228e-04, 8.45683841e-06, 1.00160255e-04,
       2.69596493e-05, 1.63780896e-04, 2.80725597e-04, 2.47098074e-04]), 'param_max_depth': masked_array(data=[No
ne, None, None, None, None, None, None, None, None,
                   None, None, None, None, None, None, None, None, None,
                   None, None, None, None, None, None, None, None, None,
                   None, None, None, None, None, None, None, None, None,
                   None, None, None, None, None, None, None, None, None,
                   None, None, None, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,
                   5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,
                   5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,
                   10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
                   10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
                   10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
                   10, 10, 10, 10, 10, 10, 15, 15, 15, 15, 15, 15, 15, 15,
                   15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15,
                   15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15,
                   15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 20, 20,
                   20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20,
                   20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20,
                   20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20,
                   20, 20, 20, 20],
             mask=[False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
```

```
                 False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False],
       fill_value='?',
            dtype=object), 'param_max_features': masked_array(data=[None, None, None, None, None, None, None, Non
e, None,
                 None, None, None, None, None, None, None, 'sqrt',
                 'sqrt', 'sqrt', 'sqrt', 'sqrt', 'sqrt', 'sqrt', 'sqrt',
                 'sqrt', 'sqrt', 'sqrt', 'sqrt', 'sqrt', 'sqrt', 'sqrt',
                 'sqrt', 'log2', 'log2', 'log2', 'log2', 'log2', 'log2',
                 'log2', 'log2', 'log2', 'log2', 'log2', 'log2', 'log2',
                 'log2', 'log2', 'log2', None, None, None, None, None,
                 None, None, None, None, None, None, None, None, None,
                 None, None, 'sqrt', 'sqrt', 'sqrt', 'sqrt', 'sqrt',
                 'sqrt', 'sqrt', 'sqrt', 'sqrt', 'sqrt', 'sqrt', 'sqrt',
                 'sqrt', 'sqrt', 'sqrt', 'sqrt', 'log2', 'log2', 'log2',
                 'log2', 'log2', 'log2', 'log2', 'log2', 'log2', 'log2',
                 'log2', 'log2', 'log2', 'log2', 'log2', 'log2', None,
                 None, None, None, None, None, None, None, None, None,
                 None, None, None, None, None, None, 'sqrt', 'sqrt',
                 'sqrt', 'sqrt', 'sqrt', 'sqrt', 'sqrt', 'sqrt', 'sqrt',
                 'sqrt', 'sqrt', 'sqrt', 'sqrt', 'sqrt', 'sqrt', 'sqrt',
                 'log2', 'log2', 'log2', 'log2', 'log2', 'log2', 'log2',
                 'log2', 'log2', 'log2', 'log2', 'log2', 'log2', 'log2',
                 'log2', 'log2', None, None, None, None, None, None,
                 None, None, None, None, None, None, None, None, None,
                 None, 'sqrt', 'sqrt', 'sqrt', 'sqrt', 'sqrt', 'sqrt',
                 'sqrt', 'sqrt', 'sqrt', 'sqrt', 'sqrt', 'sqrt', 'sqrt',
                 'sqrt', 'sqrt', 'sqrt', 'log2', 'log2', 'log2', 'log2',
                 'log2', 'log2', 'log2', 'log2', 'log2', 'log2', 'log2',
                 'log2', 'log2', 'log2', 'log2', 'log2', None, None,
                 None, None, None, None, None, None, None, None, None,
                 None, None, None, None, 'sqrt', 'sqrt', 'sqrt',
                 'sqrt', 'sqrt', 'sqrt', 'sqrt', 'sqrt', 'sqrt', 'sqrt',
                 'sqrt', 'sqrt', 'sqrt', 'sqrt', 'sqrt', 'sqrt', 'log2',
                 'log2', 'log2', 'log2', 'log2', 'log2', 'log2', 'log2',
                 'log2', 'log2', 'log2', 'log2', 'log2', 'log2', 'log2',
                 'log2'],
             mask=[False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False],
       fill_value='?',
            dtype=object), 'param_min_samples_leaf': masked_array(data=[1, 1, 1, 1, 2, 2, 2, 2, 5, 5, 5, 5, 10, 1
0, 10, 10, 1,
                 1, 1, 1, 2, 2, 2, 2, 5, 5, 5, 5, 10, 10, 10, 10, 1, 1,
                 1, 1, 2, 2, 2, 2, 5, 5, 5, 5, 10, 10, 10, 10, 1, 1, 1,
```

```
                    1, 2, 2, 2, 2, 5, 5, 5, 5, 10, 10, 10, 10, 1, 1, 1, 1,
                    2, 2, 2, 2, 5, 5, 5, 5, 10, 10, 10, 10, 1, 1, 1, 1, 2,
                    2, 2, 2, 5, 5, 5, 5, 10, 10, 10, 10, 1, 1, 1, 1, 2, 2,
                    2, 2, 5, 5, 5, 5, 10, 10, 10, 10, 1, 1, 1, 1, 2, 2, 2,
                    2, 5, 5, 5, 5, 10, 10, 10, 10, 1, 1, 1, 1, 2, 2, 2, 2,
                    5, 5, 5, 5, 10, 10, 10, 10, 1, 1, 1, 1, 2, 2, 2, 2, 5,
                    5, 5, 5, 10, 10, 10, 10, 1, 1, 1, 1, 2, 2, 2, 2, 5, 5,
                    5, 5, 10, 10, 10, 10, 1, 1, 1, 1, 2, 2, 2, 2, 5, 5, 5,
                    5, 10, 10, 10, 10, 1, 1, 1, 1, 2, 2, 2, 2, 5, 5, 5, 5,
                    10, 10, 10, 10, 1, 1, 1, 1, 2, 2, 2, 2, 5, 5, 5, 5, 10,
                    10, 10, 10, 1, 1, 1, 1, 2, 2, 2, 2, 5, 5, 5, 5, 10, 10,
                    10, 10],
              mask=[False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False],
        fill_value='?',
             dtype=object), 'param_min_samples_split': masked_array(data=[2, 5, 10, 20, 2, 5, 10, 20, 2, 5, 10, 2
0, 2, 5, 10, 20,
                    2, 5, 10, 20, 2, 5, 10, 20, 2, 5, 10, 20, 2, 5, 10, 20,
                    2, 5, 10, 20, 2, 5, 10, 20, 2, 5, 10, 20, 2, 5, 10, 20,
                    2, 5, 10, 20, 2, 5, 10, 20, 2, 5, 10, 20, 2, 5, 10, 20,
                    2, 5, 10, 20, 2, 5, 10, 20, 2, 5, 10, 20, 2, 5, 10, 20,
                    2, 5, 10, 20, 2, 5, 10, 20, 2, 5, 10, 20, 2, 5, 10, 20,
                    2, 5, 10, 20, 2, 5, 10, 20, 2, 5, 10, 20, 2, 5, 10, 20,
                    2, 5, 10, 20, 2, 5, 10, 20, 2, 5, 10, 20, 2, 5, 10, 20,
                    2, 5, 10, 20, 2, 5, 10, 20, 2, 5, 10, 20, 2, 5, 10, 20,
                    2, 5, 10, 20, 2, 5, 10, 20, 2, 5, 10, 20, 2, 5, 10, 20,
                    2, 5, 10, 20, 2, 5, 10, 20, 2, 5, 10, 20, 2, 5, 10, 20,
                    2, 5, 10, 20, 2, 5, 10, 20, 2, 5, 10, 20, 2, 5, 10, 20,
                    2, 5, 10, 20, 2, 5, 10, 20, 2, 5, 10, 20, 2, 5, 10, 20,
                    2, 5, 10, 20, 2, 5, 10, 20, 2, 5, 10, 20, 2, 5, 10, 20,
                    2, 5, 10, 20, 2, 5, 10, 20, 2, 5, 10, 20, 2, 5, 10, 20],
              mask=[False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
```

```
                  False, False, False, False, False, False, False, False,
                  False, False, False, False, False, False, False, False,
                  False, False, False, False, False, False, False, False,
                  False, False, False, False, False, False, False, False,
                  False, False, False, False, False, False, False, False],
       fill_value='?',
            dtype=object), 'params': [{'max_depth': None, 'max_features': None, 'min_samples_leaf': 1, 'min_sampl
es_split': 2}, {'max_depth': None, 'max_features': None, 'min_samples_leaf': 1, 'min_samples_split': 5}, {'max_de
pth': None, 'max_features': None, 'min_samples_leaf': 1, 'min_samples_split': 10}, {'max_depth': None, 'max_featu
res': None, 'min_samples_leaf': 1, 'min_samples_split': 20}, {'max_depth': None, 'max_features': None, 'min_sampl
es_leaf': 2, 'min_samples_split': 2}, {'max_depth': None, 'max_features': None, 'min_samples_leaf': 2, 'min_sampl
es_split': 5}, {'max_depth': None, 'max_features': None, 'min_samples_leaf': 2, 'min_samples_split': 10}, {'max_d
epth': None, 'max_features': None, 'min_samples_leaf': 2, 'min_samples_split': 20}, {'max_depth': None, 'max_feat
ures': None, 'min_samples_leaf': 5, 'min_samples_split': 2}, {'max_depth': None, 'max_features': None, 'min_sampl
es_leaf': 5, 'min_samples_split': 5}, {'max_depth': None, 'max_features': None, 'min_samples_leaf': 5, 'min_sampl
es_split': 10}, {'max_depth': None, 'max_features': None, 'min_samples_leaf': 5, 'min_samples_split': 20}, {'max_
depth': None, 'max_features': None, 'min_samples_leaf': 10, 'min_samples_split': 2}, {'max_depth': None, 'max_fea
tures': None, 'min_samples_leaf': 10, 'min_samples_split': 5}, {'max_depth': None, 'max_features': None, 'min_sam
ples_leaf': 10, 'min_samples_split': 10}, {'max_depth': None, 'max_features': None, 'min_samples_leaf': 10, 'min_
samples_split': 20}, {'max_depth': None, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 2},
{'max_depth': None, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 5}, {'max_depth': None,
'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 10}, {'max_depth': None, 'max_features': 'sqr
t', 'min_samples_leaf': 1, 'min_samples_split': 20}, {'max_depth': None, 'max_features': 'sqrt', 'min_samples_lea
f': 2, 'min_samples_split': 2}, {'max_depth': None, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_s
plit': 5}, {'max_depth': None, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 10}, {'max_dep
th': None, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 20}, {'max_depth': None, 'max_feat
ures': 'sqrt', 'min_samples_leaf': 5, 'min_samples_split': 2}, {'max_depth': None, 'max_features': 'sqrt', 'min_s
amples_leaf': 5, 'min_samples_split': 5}, {'max_depth': None, 'max_features': 'sqrt', 'min_samples_leaf': 5, 'min
_samples_split': 10}, {'max_depth': None, 'max_features': 'sqrt', 'min_samples_leaf': 5, 'min_samples_split': 2
0}, {'max_depth': None, 'max_features': 'sqrt', 'min_samples_leaf': 10, 'min_samples_split': 2}, {'max_depth': No
ne, 'max_features': 'sqrt', 'min_samples_leaf': 10, 'min_samples_split': 5}, {'max_depth': None, 'max_features':
'sqrt', 'min_samples_leaf': 10, 'min_samples_split': 10}, {'max_depth': None, 'max_features': 'sqrt', 'min_sample
s_leaf': 10, 'min_samples_split': 20}, {'max_depth': None, 'max_features': 'log2', 'min_samples_leaf': 1, 'min_sa
mples_split': 2}, {'max_depth': None, 'max_features': 'log2', 'min_samples_leaf': 1, 'min_samples_split': 5}, {'m
ax_depth': None, 'max_features': 'log2', 'min_samples_leaf': 1, 'min_samples_split': 10}, {'max_depth': None, 'ma
x_features': 'log2', 'min_samples_leaf': 1, 'min_samples_split': 20}, {'max_depth': None, 'max_features': 'log2',
'min_samples_leaf': 2, 'min_samples_split': 2}, {'max_depth': None, 'max_features': 'log2', 'min_samples_leaf':
2, 'min_samples_split': 5}, {'max_depth': None, 'max_features': 'log2', 'min_samples_leaf': 2, 'min_samples_spli
t': 10}, {'max_depth': None, 'max_features': 'log2', 'min_samples_leaf': 2, 'min_samples_split': 20}, {'max_dept
h': None, 'max_features': 'log2', 'min_samples_leaf': 5, 'min_samples_split': 2}, {'max_depth': None, 'max_featur
es': 'log2', 'min_samples_leaf': 5, 'min_samples_split': 5}, {'max_depth': None, 'max_features': 'log2', 'min_sam
ples_leaf': 5, 'min_samples_split': 10}, {'max_depth': None, 'max_features': 'log2', 'min_samples_leaf': 5, 'min_
samples_split': 20}, {'max_depth': None, 'max_features': 'log2', 'min_samples_leaf': 10, 'min_samples_split': 2},
{'max_depth': None, 'max_features': 'log2', 'min_samples_leaf': 10, 'min_samples_split': 5}, {'max_depth': None,
'max_features': 'log2', 'min_samples_leaf': 10, 'min_samples_split': 10}, {'max_depth': None, 'max_features': 'lo
g2', 'min_samples_leaf': 10, 'min_samples_split': 20}, {'max_depth': 5, 'max_features': None, 'min_samples_leaf':
1, 'min_samples_split': 2}, {'max_depth': 5, 'max_features': None, 'min_samples_leaf': 1, 'min_samples_split':
5}, {'max_depth': 5, 'max_features': None, 'min_samples_leaf': 1, 'min_samples_split': 10}, {'max_depth': 5, 'max
_features': None, 'min_samples_leaf': 1, 'min_samples_split': 20}, {'max_depth': 5, 'max_features': None, 'min_sa
mples_leaf': 2, 'min_samples_split': 2}, {'max_depth': 5, 'max_features': None, 'min_samples_leaf': 2, 'min_sampl
es_split': 5}, {'max_depth': 5, 'max_features': None, 'min_samples_leaf': 2, 'min_samples_split': 10}, {'max_dept
h': 5, 'max_features': None, 'min_samples_leaf': 2, 'min_samples_split': 20}, {'max_depth': 5, 'max_features': No
ne, 'min_samples_leaf': 5, 'min_samples_split': 2}, {'max_depth': 5, 'max_features': None, 'min_samples_leaf': 5,
'min_samples_split': 5}, {'max_depth': 5, 'max_features': None, 'min_samples_leaf': 5, 'min_samples_split': 10},
{'max_depth': 5, 'max_features': None, 'min_samples_leaf': 5, 'min_samples_split': 20}, {'max_depth': 5, 'max_fea
tures': None, 'min_samples_leaf': 10, 'min_samples_split': 2}, {'max_depth': 5, 'max_features': None, 'min_sample
s_leaf': 10, 'min_samples_split': 5}, {'max_depth': 5, 'max_features': None, 'min_samples_leaf': 10, 'min_samples
_split': 10}, {'max_depth': 5, 'max_features': None, 'min_samples_leaf': 10, 'min_samples_split': 20}, {'max_dept
h': 5, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 2}, {'max_depth': 5, 'max_features':
'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 5}, {'max_depth': 5, 'max_features': 'sqrt', 'min_samples_lea
f': 1, 'min_samples_split': 10}, {'max_depth': 5, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_spl
it': 20}, {'max_depth': 5, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 2}, {'max_depth':
5, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 5}, {'max_depth': 5, 'max_features': 'sqr
t', 'min_samples_leaf': 2, 'min_samples_split': 10}, {'max_depth': 5, 'max_features': 'sqrt', 'min_samples_leaf':
2, 'min_samples_split': 20}, {'max_depth': 5, 'max_features': 'sqrt', 'min_samples_leaf': 5, 'min_samples_split':
2}, {'max_depth': 5, 'max_features': 'sqrt', 'min_samples_leaf': 5, 'min_samples_split': 5}, {'max_depth': 5, 'ma
x_features': 'sqrt', 'min_samples_leaf': 5, 'min_samples_split': 10}, {'max_depth': 5, 'max_features': 'sqrt', 'm
in_samples_leaf': 5, 'min_samples_split': 20}, {'max_depth': 5, 'max_features': 'sqrt', 'min_samples_leaf': 10,
'min_samples_split': 2}, {'max_depth': 5, 'max_features': 'sqrt', 'min_samples_leaf': 10, 'min_samples_split':
5}, {'max_depth': 5, 'max_features': 'sqrt', 'min_samples_leaf': 10, 'min_samples_split': 10}, {'max_depth': 5,
'max_features': 'sqrt', 'min_samples_leaf': 10, 'min_samples_split': 20}, {'max_depth': 5, 'max_features': 'log
2', 'min_samples_leaf': 1, 'min_samples_split': 2}, {'max_depth': 5, 'max_features': 'log2', 'min_samples_leaf':
1, 'min_samples_split': 5}, {'max_depth': 5, 'max_features': 'log2', 'min_samples_leaf': 1, 'min_samples_split':
10}, {'max_depth': 5, 'max_features': 'log2', 'min_samples_leaf': 1, 'min_samples_split': 20}, {'max_depth': 5,
'max_features': 'log2', 'min_samples_leaf': 2, 'min_samples_split': 2}, {'max_depth': 5, 'max_features': 'log2',
'min_samples_leaf': 2, 'min_samples_split': 5}, {'max_depth': 5, 'max_features': 'log2', 'min_samples_leaf': 2,
'min_samples_split': 10}, {'max_depth': 5, 'max_features': 'log2', 'min_samples_leaf': 2, 'min_samples_split': 2
0}, {'max_depth': 5, 'max_features': 'log2', 'min_samples_leaf': 5, 'min_samples_split': 2}, {'max_depth': 5, 'ma
x_features': 'log2', 'min_samples_leaf': 5, 'min_samples_split': 5}, {'max_depth': 5, 'max_features': 'log2', 'mi
n_samples_leaf': 5, 'min_samples_split': 10}, {'max_depth': 5, 'max_features': 'log2', 'min_samples_leaf': 5, 'mi
n_samples_split': 20}, {'max_depth': 5, 'max_features': 'log2', 'min_samples_leaf': 10, 'min_samples_split': 2},
{'max_depth': 5, 'max_features': 'log2', 'min_samples_leaf': 10, 'min_samples_split': 5}, {'max_depth': 5, 'max_f
eatures': 'log2', 'min_samples_leaf': 10, 'min_samples_split': 10}, {'max_depth': 5, 'max_features': 'log2', 'min
_samples_leaf': 10, 'min_samples_split': 20}, {'max_depth': 10, 'max_features': None, 'min_samples_leaf': 1, 'min
```

_samples_split': 2}, {'max_depth': 10, 'max_features': None, 'min_samples_leaf': 1, 'min_samples_split': 5}, {'ma
x_depth': 10, 'max_features': None, 'min_samples_leaf': 1, 'min_samples_split': 10}, {'max_depth': 10, 'max_featu
res': None, 'min_samples_leaf': 1, 'min_samples_split': 20}, {'max_depth': 10, 'max_features': None, 'min_samples
_leaf': 2, 'min_samples_split': 2}, {'max_depth': 10, 'max_features': None, 'min_samples_leaf': 2, 'min_samples_s
plit': 5}, {'max_depth': 10, 'max_features': None, 'min_samples_leaf': 2, 'min_samples_split': 10}, {'max_depth':
10, 'max_features': None, 'min_samples_leaf': 2, 'min_samples_split': 20}, {'max_depth': 10, 'max_features': Non
e, 'min_samples_leaf': 5, 'min_samples_split': 2}, {'max_depth': 10, 'max_features': None, 'min_samples_leaf': 5,
'min_samples_split': 5}, {'max_depth': 10, 'max_features': None, 'min_samples_leaf': 5, 'min_samples_split': 10},
{'max_depth': 10, 'max_features': None, 'min_samples_leaf': 5, 'min_samples_split': 20}, {'max_depth': 10, 'max_f
eatures': None, 'min_samples_leaf': 10, 'min_samples_split': 2}, {'max_depth': 10, 'max_features': None, 'min_sam
ples_leaf': 10, 'min_samples_split': 5}, {'max_depth': 10, 'max_features': None, 'min_samples_leaf': 10, 'min_sam
ples_split': 10}, {'max_depth': 10, 'max_features': None, 'min_samples_leaf': 10, 'min_samples_split': 20}, {'max
_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 2}, {'max_depth': 10, 'max_featu
res': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 5}, {'max_depth': 10, 'max_features': 'sqrt', 'min_samp
les_leaf': 1, 'min_samples_split': 10}, {'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_sam
ples_split': 20}, {'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 2}, {'max
_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 5}, {'max_depth': 10, 'max_featu
res': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 10}, {'max_depth': 10, 'max_features': 'sqrt', 'min_sam
ples_leaf': 2, 'min_samples_split': 20}, {'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 5, 'min_sa
mples_split': 2}, {'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 5, 'min_samples_split': 5}, {'max
_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 5, 'min_samples_split': 10}, {'max_depth': 10, 'max_feat
ures': 'sqrt', 'min_samples_leaf': 5, 'min_samples_split': 20}, {'max_depth': 10, 'max_features': 'sqrt', 'min_sa
mples_leaf': 10, 'min_samples_split': 2}, {'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 10, 'min_
samples_split': 5}, {'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 10, 'min_samples_split': 10},
{'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 10, 'min_samples_split': 20}, {'max_depth': 10, 'ma
x_features': 'log2', 'min_samples_leaf': 1, 'min_samples_split': 2}, {'max_depth': 10, 'max_features': 'log2', 'm
in_samples_leaf': 1, 'min_samples_split': 5}, {'max_depth': 10, 'max_features': 'log2', 'min_samples_leaf': 1, 'm
in_samples_split': 10}, {'max_depth': 10, 'max_features': 'log2', 'min_samples_leaf': 1, 'min_samples_split': 2
0}, {'max_depth': 10, 'max_features': 'log2', 'min_samples_leaf': 2, 'min_samples_split': 2}, {'max_depth': 10,
'max_features': 'log2', 'min_samples_leaf': 2, 'min_samples_split': 5}, {'max_depth': 10, 'max_features': 'log2',
'min_samples_leaf': 2, 'min_samples_split': 10}, {'max_depth': 10, 'max_features': 'log2', 'min_samples_leaf': 2,
'min_samples_split': 20}, {'max_depth': 10, 'max_features': 'log2', 'min_samples_leaf': 5, 'min_samples_split':
2}, {'max_depth': 10, 'max_features': 'log2', 'min_samples_leaf': 5, 'min_samples_split': 5}, {'max_depth': 10,
'max_features': 'log2', 'min_samples_leaf': 5, 'min_samples_split': 10}, {'max_depth': 10, 'max_features': 'log
2', 'min_samples_leaf': 5, 'min_samples_split': 20}, {'max_depth': 10, 'max_features': 'log2', 'min_samples_lea
f': 10, 'min_samples_split': 2}, {'max_depth': 10, 'max_features': 'log2', 'min_samples_leaf': 10, 'min_samples_s
plit': 5}, {'max_depth': 10, 'max_features': 'log2', 'min_samples_leaf': 10, 'min_samples_split': 10}, {'max_dept
h': 10, 'max_features': 'log2', 'min_samples_leaf': 10, 'min_samples_split': 20}, {'max_depth': 15, 'max_feature
s': None, 'min_samples_leaf': 1, 'min_samples_split': 2}, {'max_depth': 15, 'max_features': None, 'min_samples_le
af': 1, 'min_samples_split': 5}, {'max_depth': 15, 'max_features': None, 'min_samples_leaf': 1, 'min_samples_spli
t': 10}, {'max_depth': 15, 'max_features': None, 'min_samples_leaf': 1, 'min_samples_split': 20}, {'max_depth': 1
5, 'max_features': None, 'min_samples_leaf': 2, 'min_samples_split': 2}, {'max_depth': 15, 'max_features': None,
'min_samples_leaf': 2, 'min_samples_split': 5}, {'max_depth': 15, 'max_features': None, 'min_samples_leaf': 2, 'm
in_samples_split': 10}, {'max_depth': 15, 'max_features': None, 'min_samples_leaf': 2, 'min_samples_split': 20},
{'max_depth': 15, 'max_features': None, 'min_samples_leaf': 5, 'min_samples_split': 2}, {'max_depth': 15, 'max_fe
atures': None, 'min_samples_leaf': 5, 'min_samples_split': 5}, {'max_depth': 15, 'max_features': None, 'min_sampl
es_leaf': 5, 'min_samples_split': 10}, {'max_depth': 15, 'max_features': None, 'min_samples_leaf': 5, 'min_sample
s_split': 20}, {'max_depth': 15, 'max_features': None, 'min_samples_leaf': 10, 'min_samples_split': 2}, {'max_dep
th': 15, 'max_features': None, 'min_samples_leaf': 10, 'min_samples_split': 5}, {'max_depth': 15, 'max_features':
None, 'min_samples_leaf': 10, 'min_samples_split': 10}, {'max_depth': 15, 'max_features': None, 'min_samples_lea
f': 10, 'min_samples_split': 20}, {'max_depth': 15, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_s
plit': 2}, {'max_depth': 15, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 5}, {'max_dept
h': 15, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 10}, {'max_depth': 15, 'max_feature
s': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 20}, {'max_depth': 15, 'max_features': 'sqrt', 'min_sampl
es_leaf': 2, 'min_samples_split': 2}, {'max_depth': 15, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_sampl
es_split': 5}, {'max_depth': 15, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 10}, {'max_d
epth': 15, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 20}, {'max_depth': 15, 'max_featur
es': 'sqrt', 'min_samples_leaf': 5, 'min_samples_split': 2}, {'max_depth': 15, 'max_features': 'sqrt', 'min_sampl
es_leaf': 5, 'min_samples_split': 5}, {'max_depth': 15, 'max_features': 'sqrt', 'min_samples_leaf': 5, 'min_sampl
es_split': 10}, {'max_depth': 15, 'max_features': 'sqrt', 'min_samples_leaf': 5, 'min_samples_split': 20}, {'max_
depth': 15, 'max_features': 'sqrt', 'min_samples_leaf': 10, 'min_samples_split': 2}, {'max_depth': 15, 'max_featu
res': 'sqrt', 'min_samples_leaf': 10, 'min_samples_split': 5}, {'max_depth': 15, 'max_features': 'sqrt', 'min_sam
ples_leaf': 10, 'min_samples_split': 10}, {'max_depth': 15, 'max_features': 'sqrt', 'min_samples_leaf': 10, 'min_
samples_split': 20}, {'max_depth': 15, 'max_features': 'log2', 'min_samples_leaf': 1, 'min_samples_split': 2},
{'max_depth': 15, 'max_features': 'log2', 'min_samples_leaf': 1, 'min_samples_split': 5}, {'max_depth': 15, 'max_
features': 'log2', 'min_samples_leaf': 1, 'min_samples_split': 10}, {'max_depth': 15, 'max_features': 'log2', 'mi
n_samples_leaf': 1, 'min_samples_split': 20}, {'max_depth': 15, 'max_features': 'log2', 'min_samples_leaf': 2, 'm
in_samples_split': 2}, {'max_depth': 15, 'max_features': 'log2', 'min_samples_leaf': 2, 'min_samples_split': 5},
{'max_depth': 15, 'max_features': 'log2', 'min_samples_leaf': 2, 'min_samples_split': 10}, {'max_depth': 15, 'max
_features': 'log2', 'min_samples_leaf': 2, 'min_samples_split': 20}, {'max_depth': 15, 'max_features': 'log2', 'm
in_samples_leaf': 5, 'min_samples_split': 2}, {'max_depth': 15, 'max_features': 'log2', 'min_samples_leaf': 5, 'm
in_samples_split': 5}, {'max_depth': 15, 'max_features': 'log2', 'min_samples_leaf': 5, 'min_samples_split': 10},
{'max_depth': 15, 'max_features': 'log2', 'min_samples_leaf': 5, 'min_samples_split': 20}, {'max_depth': 15, 'max
_features': 'log2', 'min_samples_leaf': 10, 'min_samples_split': 2}, {'max_depth': 15, 'max_features': 'log2', 'm
in_samples_leaf': 10, 'min_samples_split': 5}, {'max_depth': 15, 'max_features': 'log2', 'min_samples_leaf': 10,
'min_samples_split': 10}, {'max_depth': 15, 'max_features': 'log2', 'min_samples_leaf': 10, 'min_samples_split':
20}, {'max_depth': 20, 'max_features': None, 'min_samples_leaf': 1, 'min_samples_split': 2}, {'max_depth': 20, 'm
ax_features': None, 'min_samples_leaf': 1, 'min_samples_split': 5}, {'max_depth': 20, 'max_features': None, 'min_
samples_leaf': 1, 'min_samples_split': 10}, {'max_depth': 20, 'max_features': None, 'min_samples_leaf': 1, 'min_s
amples_split': 20}, {'max_depth': 20, 'max_features': None, 'min_samples_leaf': 2, 'min_samples_split': 2}, {'max
_depth': 20, 'max_features': None, 'min_samples_leaf': 2, 'min_samples_split': 5}, {'max_depth': 20, 'max_feature
s': None, 'min_samples_leaf': 2, 'min_samples_split': 10}, {'max_depth': 20, 'max_features': None, 'min_samples_l
eaf': 2, 'min_samples_split': 20}, {'max_depth': 20, 'max_features': None, 'min_samples_leaf': 5, 'min_samples_sp
lit': 2}, {'max_depth': 20, 'max_features': None, 'min_samples_leaf': 5, 'min_samples_split': 5}, {'max_depth': 2

0, 'max_features': None, 'min_samples_leaf': 5, 'min_samples_split': 10}, {'max_depth': 20, 'max_features': None, 'min_samples_leaf': 5, 'min_samples_split': 20}, {'max_depth': 20, 'max_features': None, 'min_samples_leaf': 10, 'min_samples_split': 2}, {'max_depth': 20, 'max_features': None, 'min_samples_leaf': 10, 'min_samples_split': 5}, {'max_depth': 20, 'max_features': None, 'min_samples_leaf': 10, 'min_samples_split': 10}, {'max_depth': 20, 'max_features': None, 'min_samples_leaf': 10, 'min_samples_split': 20}, {'max_depth': 20, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 2}, {'max_depth': 20, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 5}, {'max_depth': 20, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 10}, {'max_depth': 20, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 20}, {'max_depth': 20, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 2}, {'max_depth': 20, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 5}, {'max_depth': 20, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 10}, {'max_depth': 20, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 20}, {'max_depth': 20, 'max_features': 'sqrt', 'min_samples_leaf': 5, 'min_samples_split': 2}, {'max_depth': 20, 'max_features': 'sqrt', 'min_samples_leaf': 5, 'min_samples_split': 5}, {'max_depth': 20, 'max_features': 'sqrt', 'min_samples_leaf': 5, 'min_samples_split': 10}, {'max_depth': 20, 'max_features': 'sqrt', 'min_samples_leaf': 5, 'min_samples_split': 20}, {'max_depth': 20, 'max_features': 'sqrt', 'min_samples_leaf': 10, 'min_samples_split': 2}, {'max_depth': 20, 'max_features': 'sqrt', 'min_samples_leaf': 10, 'min_samples_split': 5}, {'max_depth': 20, 'max_features': 'sqrt', 'min_samples_leaf': 10, 'min_samples_split': 10}, {'max_depth': 20, 'max_features': 'sqrt', 'min_samples_leaf': 10, 'min_samples_split': 20}, {'max_depth': 20, 'max_features': 'log2', 'min_samples_leaf': 1, 'min_samples_split': 2}, {'max_depth': 20, 'max_features': 'log2', 'min_samples_leaf': 1, 'min_samples_split': 5}, {'max_depth': 20, 'max_features': 'log2', 'min_samples_leaf': 1, 'min_samples_split': 10}, {'max_depth': 20, 'max_features': 'log2', 'min_samples_leaf': 1, 'min_samples_split': 20}, {'max_depth': 20, 'max_features': 'log2', 'min_samples_leaf': 2, 'min_samples_split': 2}, {'max_depth': 20, 'max_features': 'log2', 'min_samples_leaf': 2, 'min_samples_split': 5}, {'max_depth': 20, 'max_features': 'log2', 'min_samples_leaf': 2, 'min_samples_split': 10}, {'max_depth': 20, 'max_features': 'log2', 'min_samples_leaf': 2, 'min_samples_split': 20}, {'max_depth': 20, 'max_features': 'log2', 'min_samples_leaf': 5, 'min_samples_split': 2}, {'max_depth': 20, 'max_features': 'log2', 'min_samples_leaf': 5, 'min_samples_split': 5}, {'max_depth': 20, 'max_features': 'log2', 'min_samples_leaf': 5, 'min_samples_split': 10}, {'max_depth': 20, 'max_features': 'log2', 'min_samples_leaf': 5, 'min_samples_split': 20}, {'max_depth': 20, 'max_features': 'log2', 'min_samples_leaf': 10, 'min_samples_split': 2}, {'max_depth': 20, 'max_features': 'log2', 'min_samples_leaf': 10, 'min_samples_split': 5}, {'max_depth': 20, 'max_features': 'log2', 'min_samples_leaf': 10, 'min_samples_split': 10}, {'max_depth': 20, 'max_features': 'log2', 'min_samples_leaf': 10, 'min_samples_split': 20}], 'split0_test_score': array([0.86213887, 0.86568677, 0.87328941, 0.88190573, 0.87785099,
       0.87278256, 0.87734415, 0.88545362, 0.88545362, 0.88849468,
       0.88849468, 0.88900152, 0.89508363, 0.89508363, 0.89508363,
       0.89457679, 0.8540294 , 0.86771414, 0.8707552 , 0.89508363,
       0.8768373 , 0.87937152, 0.87886467, 0.88494678, 0.8707552 ,
       0.8839331 , 0.87328941, 0.89305626, 0.89001521, 0.90015205,
       0.89204257, 0.89254942, 0.85707045, 0.86974151, 0.86517993,
       0.87430309, 0.86822098, 0.87176888, 0.87328941, 0.88596047,
       0.89052205, 0.87582362, 0.86822098, 0.88900152, 0.89153573,
       0.89254942, 0.8803852 , 0.8839331 , 0.89913837, 0.89913837,
       0.89913837, 0.89913837, 0.89913837, 0.89964521, 0.89913837,
       0.89913837, 0.89913837, 0.89913837, 0.89913837, 0.89913837,
       0.89863153, 0.89863153, 0.89863153, 0.89863153, 0.89660416,
       0.8540294 , 0.88342625, 0.89609731, 0.89913837, 0.85098834,
       0.8504815 , 0.89660416, 0.86771414, 0.88697415, 0.89153573,
       0.88748099, 0.89863153, 0.86061835, 0.89609731, 0.86112519,
       0.88950836, 0.89913837, 0.87227572, 0.87633046, 0.89204257,
       0.89254942, 0.85352255, 0.89153573, 0.89254942, 0.8707552 ,
       0.85605677, 0.89812468, 0.87886467, 0.88241257, 0.86213887,
       0.89660416, 0.88545362, 0.88849468, 0.89102889, 0.88849468,
       0.88950836, 0.88748099, 0.89001521, 0.89001521, 0.88596047,
       0.88596047, 0.88545362, 0.88545362, 0.89153573, 0.89102889,
       0.89102889, 0.89102889, 0.88646731, 0.88900152, 0.86974151,
       0.87937152, 0.89305626, 0.89052205, 0.88494678, 0.88342625,
       0.88291941, 0.88900152, 0.87734415, 0.89964521, 0.88291941,
       0.89761784, 0.89457679, 0.88950836, 0.87227572, 0.88545362,
       0.88596047, 0.88697415, 0.87633046, 0.87227572, 0.8839331 ,
       0.87379625, 0.88900152, 0.87886467, 0.88697415, 0.89559047,
       0.89660416, 0.90167258, 0.897111 , 0.87227572, 0.87227572,
       0.87430309, 0.87582362, 0.8839331 , 0.87582362, 0.88089204,
       0.8839331 , 0.88596047, 0.88646731, 0.88596047, 0.88646731,
       0.88900152, 0.89508363, 0.89508363, 0.89508363, 0.89457679,
       0.86822098, 0.85707045, 0.87987836, 0.88596047, 0.88950836,
       0.88190573, 0.87531678, 0.88748099, 0.88342625, 0.88443994,
       0.87987836, 0.89254942, 0.89153573, 0.88291941, 0.89812468,
       0.88241257, 0.87278256, 0.86872783, 0.87328941, 0.88241257,
       0.88342625, 0.87633046, 0.87835783, 0.88342625, 0.87227572,
       0.89204257, 0.87227572, 0.88139888, 0.88342625, 0.88443994,
       0.88950836, 0.89457679, 0.86771414, 0.8707552 , 0.87633046,
       0.88190573, 0.8768373 , 0.87328941, 0.87835783, 0.88494678,
       0.88697415, 0.88950836, 0.88849468, 0.88900152, 0.89508363,
       0.89457679, 0.89508363, 0.89457679, 0.85200203, 0.86213887,
       0.86923467, 0.88798784, 0.8768373 , 0.87582362, 0.88139888,
       0.8803852 , 0.89305626, 0.87734415, 0.88798784, 0.88443994,
       0.88545362, 0.88596047, 0.89508363, 0.88089204, 0.87835783,
       0.86771414, 0.88241257, 0.89052205, 0.88190573, 0.86568677,
       0.88190573, 0.88443994, 0.88596047, 0.88291941, 0.88241257,
       0.89001521, 0.89102889, 0.87987836, 0.88545362, 0.8935631]), 'split1_test_score': array([0.85352255, 0.86517993, 0.86213887, 0.86822098, 0.87227572,
       0.87328941, 0.86974151, 0.87278256, 0.87987836, 0.87835783,
       0.88139888, 0.88089204, 0.88443994, 0.88443994, 0.88545362,
       0.88646731, 0.86061835, 0.87379625, 0.8768373 , 0.87379625,
       0.88697415, 0.86974151, 0.86872783, 0.88291941, 0.88697415,

```
       0.87480993, 0.89254942, 0.8768373 , 0.87531678, 0.8803852 ,
       0.88443994, 0.8768373 , 0.86315256, 0.87227572, 0.8707552 ,
       0.89102889, 0.87227572, 0.88443994, 0.88241257, 0.89052205,
       0.86619361, 0.87987836, 0.89052205, 0.87785099, 0.8803852 ,
       0.89001521, 0.89102889, 0.8768373 , 0.89559047, 0.89559047,
       0.89559047, 0.89559047, 0.89559047, 0.89559047, 0.89559047,
       0.89559047, 0.89660416, 0.89660416, 0.89660416, 0.89660416,
       0.89913837, 0.89863153, 0.89863153, 0.89863153, 0.86872783,
       0.87937152, 0.85707045, 0.86670046, 0.88950836, 0.86011151,
       0.86467309, 0.88798784, 0.85200203, 0.8672073 , 0.87734415,
       0.86467309, 0.86213887, 0.86568677, 0.87379625, 0.87126204,
       0.88697415, 0.89913837, 0.85960466, 0.88545362, 0.88291941,
       0.89001521, 0.87278256, 0.897111  , 0.89204257, 0.88900152,
       0.87582362, 0.87328941, 0.85453624, 0.88342625, 0.89609731,
       0.88798784, 0.88139888, 0.88241257, 0.88139888, 0.87886467,
       0.88443994, 0.88494678, 0.87937152, 0.87886467, 0.88596047,
       0.88494678, 0.88545362, 0.88545362, 0.88545362, 0.88646731,
       0.88545362, 0.88545362, 0.89052205, 0.87734415, 0.88342625,
       0.88798784, 0.8672073 , 0.88342625, 0.88443994, 0.88950836,
       0.88900152, 0.89812468, 0.89508363, 0.89001521, 0.89153573,
       0.8803852 , 0.8803852 , 0.89406994, 0.89153573, 0.89102889,
       0.88596047, 0.88849468, 0.88596047, 0.88342625, 0.86974151,
       0.89660416, 0.88443994, 0.89254942, 0.88089204, 0.89559047,
       0.88697415, 0.89254942, 0.89102889, 0.87987836, 0.86568677,
       0.86619361, 0.86517993, 0.86923467, 0.87430309, 0.87278256,
       0.86974151, 0.87278256, 0.87835783, 0.88089204, 0.88089204,
       0.88139888, 0.88443994, 0.88342625, 0.88748099, 0.88646731,
       0.86923467, 0.85656361, 0.85757729, 0.88849468, 0.88443994,
       0.8839331 , 0.87582362, 0.88494678, 0.87633046, 0.88241257,
       0.8839331 , 0.87328941, 0.88596047, 0.88646731, 0.88545362,
       0.88545362, 0.86011151, 0.86517993, 0.8803852 , 0.88089204,
       0.87987836, 0.87785099, 0.88241257, 0.88342625, 0.89204257,
       0.89102889, 0.87633046, 0.88849468, 0.89153573, 0.87126204,
       0.88900152, 0.88697415, 0.85554992, 0.86416624, 0.86315256,
       0.86872783, 0.86822098, 0.87024835, 0.86822098, 0.87278256,
       0.87886467, 0.88139888, 0.87987836, 0.88139888, 0.88342625,
       0.88748099, 0.88342625, 0.88342625, 0.85656361, 0.87126204,
       0.88089204, 0.88291941, 0.87582362, 0.87633046, 0.86872783,
       0.88900152, 0.8839331 , 0.86112519, 0.88646731, 0.87987836,
       0.88545362, 0.88646731, 0.88139888, 0.87937152, 0.86771414,
       0.86619361, 0.87633046, 0.8839331 , 0.85554992, 0.87582362,
       0.88190573, 0.88190573, 0.87531678, 0.87785099, 0.87633046,
       0.89153573, 0.88443994, 0.89204257, 0.88697415, 0.89559047]), 'split2_test_score': array([0.86670046, 0.86
72073 , 0.87785099, 0.8803852 , 0.87227572,
       0.87835783, 0.88494678, 0.88646731, 0.88950836, 0.88900152,
       0.88697415, 0.89812468, 0.89863153, 0.89812468, 0.89964521,
       0.90015205, 0.86163203, 0.87835783, 0.88798784, 0.89761784,
       0.88342625, 0.87987836, 0.88596047, 0.89559047, 0.88241257,
       0.8839331 , 0.89761784, 0.90572732, 0.8935631 , 0.89102889,
       0.897111  , 0.90217942, 0.86213887, 0.87430309, 0.88139888,
       0.8935631 , 0.87785099, 0.88646731, 0.89406994, 0.88697415,
       0.88849468, 0.88089204, 0.89204257, 0.89812468, 0.90116574,
       0.88494678, 0.90876837, 0.89457679, 0.90522048, 0.90522048,
       0.90522048, 0.90522048, 0.90522048, 0.90522048, 0.90522048,
       0.90522048, 0.90623416, 0.90623416, 0.90623416, 0.90623416,
       0.90826153, 0.90826153, 0.90826153, 0.90826153, 0.85554992,
       0.90015205, 0.89660416, 0.88798784, 0.87785099, 0.91180943,
       0.87126204, 0.89863153, 0.89559047, 0.90927522, 0.84744045,
       0.91332995, 0.85149518, 0.88697415, 0.89964521, 0.8839331 ,
       0.8768373 , 0.90217942, 0.906741  , 0.90522048, 0.86416624,
       0.88950836, 0.90826153, 0.90927522, 0.89863153, 0.86974151,
       0.87633046, 0.88494678, 0.84845413, 0.90116574, 0.88190573,
       0.87785099, 0.89254942, 0.89102889, 0.89102889, 0.89254942,
       0.89001521, 0.89204257, 0.89812468, 0.89660416, 0.89254942,
       0.89204257, 0.89102889, 0.89913837, 0.89812468, 0.89863153,
       0.89660416, 0.89812468, 0.88291941, 0.88646731, 0.897111  ,
       0.89102889, 0.88190573, 0.87886467, 0.88849468, 0.89812468,
       0.89102889, 0.8935631 , 0.89863153, 0.89964521, 0.88139888,
       0.90572732, 0.89001521, 0.89559047, 0.87430309, 0.8935631 ,
       0.89204257, 0.89102889, 0.89559047, 0.89406994, 0.897111  ,
       0.88900152, 0.89660416, 0.8935631 , 0.88646731, 0.89508363,
       0.88089204, 0.90978206, 0.89457679, 0.90623416, 0.87024835,
       0.87379625, 0.87937152, 0.8839331 , 0.87835783, 0.87126204,
       0.88646731, 0.88443994, 0.88849468, 0.88950836, 0.88798784,
       0.89863153, 0.9006589 , 0.9006589 , 0.90015205, 0.89913837,
       0.87987836, 0.87328941, 0.88139888, 0.89153573, 0.87430309,
       0.87785099, 0.88849468, 0.87379625, 0.89254942, 0.89305626,
       0.89001521, 0.90268626, 0.90876837, 0.90268626, 0.89254942,
       0.89660416, 0.88596047, 0.88139888, 0.87937152, 0.88342625,
       0.88190573, 0.88646731, 0.88494678, 0.90369995, 0.90217942,
       0.89406994, 0.88798784, 0.88900152, 0.89964521, 0.90319311,
       0.89863153, 0.89406994, 0.8707552 , 0.86923467, 0.8803852 ,
       0.88089204, 0.87430309, 0.87734415, 0.88596047, 0.88697415,
       0.88849468, 0.88849468, 0.88646731, 0.89812468, 0.89863153,
```

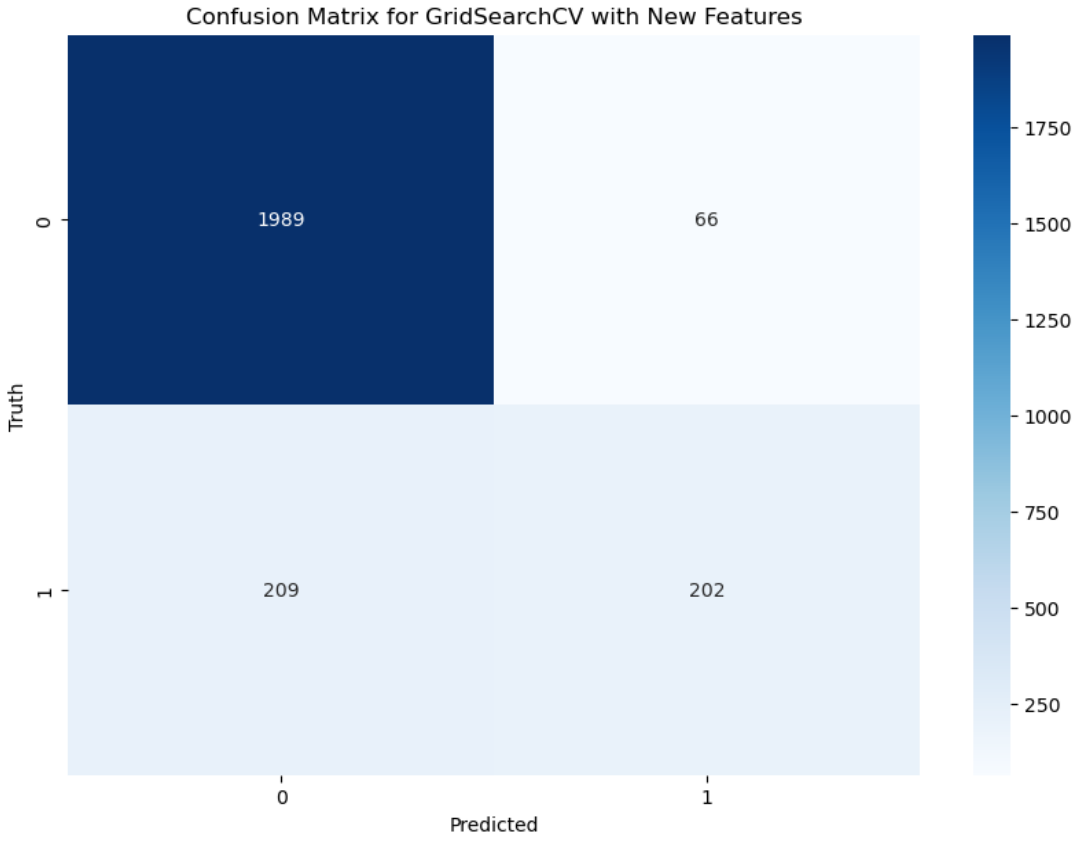
```
       0.89812468, 0.89863153, 0.89812468, 0.86315256, 0.87379625,
       0.88596047, 0.87835783, 0.8768373 , 0.87430309, 0.88089204,
       0.88900152, 0.88646731, 0.90217942, 0.89812468, 0.85960466,
       0.89609731, 0.9006589 , 0.89964521, 0.89508363, 0.87379625,
       0.8803852 , 0.88849468, 0.89052205, 0.8672073 , 0.8839331 ,
       0.88798784, 0.9006589 , 0.89102889, 0.89964521, 0.8839331 ,
       0.89153573, 0.89660416, 0.89913837, 0.90015205, 0.89204257]), 'split3_test_score': array([0.84591992, 0.86
517993, 0.85656361, 0.86619361, 0.86670046,
       0.8707552 , 0.86264572, 0.86923467, 0.87379625, 0.87430309,
       0.87430309, 0.87227572, 0.87531678, 0.87430309, 0.87531678,
       0.87582362, 0.85707045, 0.84591992, 0.85960466, 0.87328941,
       0.86416624, 0.87379625, 0.87531678, 0.88545362, 0.86670046,
       0.88139888, 0.87480993, 0.87024835, 0.86467309, 0.87937152,
       0.88950836, 0.88596047, 0.8469336 , 0.85200203, 0.86568677,
       0.87835783, 0.86213887, 0.86568677, 0.86771414, 0.8707552 ,
       0.87024835, 0.86467309, 0.87379625, 0.87176888, 0.88798784,
       0.8839331 , 0.88849468, 0.87785099, 0.8935631 , 0.8935631 ,
       0.8935631 , 0.8935631 , 0.8935631 , 0.8935631 , 0.8935631 ,
       0.8935631 , 0.89508363, 0.89508363, 0.89508363, 0.89508363,
       0.89508363, 0.89508363, 0.89508363, 0.89508363, 0.88545362,
       0.88139888, 0.87480993, 0.88849468, 0.85554992, 0.86112519,
       0.86315256, 0.88900152, 0.88596047, 0.89609731, 0.8707552 ,
       0.87430309, 0.89254942, 0.84439939, 0.8803852 , 0.86771414,
       0.88748099, 0.88950836, 0.86872783, 0.88241257, 0.87734415,
       0.8636594 , 0.85301571, 0.84997466, 0.84541308, 0.87734415,
       0.88545362, 0.89204257, 0.86771414, 0.85808414, 0.86416624,
       0.8768373 , 0.87379625, 0.87734415, 0.87176888, 0.87328941,
       0.8803852 , 0.87531678, 0.87379625, 0.87430309, 0.87430309,
       0.87430309, 0.87582362, 0.87278256, 0.87480993, 0.87582362,
       0.87531678, 0.87480993, 0.85960466, 0.87430309, 0.88545362,
       0.87785099, 0.87430309, 0.8672073 , 0.88089204, 0.88748099,
       0.87379625, 0.87176888, 0.88798784, 0.88089204, 0.87886467,
       0.86467309, 0.8803852 , 0.88596047, 0.86974151, 0.88798784,
       0.8636594 , 0.87734415, 0.87531678, 0.85453624, 0.8636594 ,
       0.87987836, 0.88545362, 0.86670046, 0.87126204, 0.88545362,
       0.86771414, 0.88089204, 0.87987836, 0.8803852 , 0.86517993,
       0.87176888, 0.87024835, 0.8707552 , 0.87582362, 0.87937152,
       0.86974151, 0.87176888, 0.87582362, 0.87633046, 0.87480993,
       0.87278256, 0.87430309, 0.87582362, 0.87430309, 0.87531678,
       0.86315256, 0.86923467, 0.85808414, 0.86670046, 0.86872783,
       0.88190573, 0.86923467, 0.87379625, 0.87785099, 0.86670046,
       0.88241257, 0.86771414, 0.88494678, 0.8935631 , 0.87633046,
       0.86568677, 0.86517993, 0.85707045, 0.86619361, 0.87835783,
       0.87126204, 0.86822098, 0.87531678, 0.88139888, 0.87785099,
       0.87480993, 0.88342625, 0.87227572, 0.88190573, 0.87734415,
       0.87430309, 0.88900152, 0.85250887, 0.86315256, 0.86264572,
       0.86619361, 0.86670046, 0.86872783, 0.86416624, 0.86923467,
       0.87430309, 0.87278256, 0.87126204, 0.87227572, 0.87531678,
       0.87430309, 0.87430309, 0.87582362, 0.86264572, 0.85909782,
       0.8469336 , 0.85859098, 0.87126204, 0.86467309, 0.87379625,
       0.88241257, 0.88596047, 0.88596047, 0.88291941, 0.87734415,
       0.8803852 , 0.87227572, 0.88697415, 0.88089204, 0.84287886,
       0.85250887, 0.85859098, 0.86568677, 0.85960466, 0.86670046,
       0.86315256, 0.87633046, 0.88190573, 0.85808414, 0.86872783,
       0.86872783, 0.87582362, 0.87937152, 0.87937152, 0.87633046]), 'split4_test_score': array([0.86713996, 0.87
018256, 0.87119675, 0.88438134, 0.86764706,
       0.87068966, 0.87018256, 0.88590264, 0.87322515, 0.87271805,
       0.87271805, 0.88133874, 0.89300203, 0.89198783, 0.89046653,
       0.89300203, 0.85547667, 0.87423935, 0.87677485, 0.88387424,
       0.88184584, 0.86916836, 0.87068966, 0.88894523, 0.89452333,
       0.87981744, 0.88488844, 0.89858012, 0.88945233, 0.89503043,
       0.87829615, 0.89452333, 0.85851927, 0.86916836, 0.87221095,
       0.89350913, 0.87778905, 0.88539554, 0.89249493, 0.89148073,
       0.89350913, 0.88843813, 0.88286004, 0.88894523, 0.89604462,
       0.88387424, 0.89756592, 0.87728195, 0.90212982, 0.90212982,
       0.90212982, 0.90212982, 0.90111562, 0.90212982, 0.90111562,
       0.90111562, 0.90111562, 0.90111562, 0.90212982, 0.90111562,
       0.90314402, 0.90314402, 0.90314402, 0.90314402, 0.86815416,
       0.89452333, 0.86054767, 0.90212982, 0.85953347, 0.88843813,
       0.88793103, 0.85851927, 0.90517241, 0.86561866, 0.87677485,
       0.89705882, 0.87576065, 0.89807302, 0.88336714, 0.85649087,
       0.86308316, 0.88640974, 0.89604462, 0.90212982, 0.88590264,
       0.89452333, 0.88133874, 0.90060852, 0.88995943, 0.90365112,
       0.88286004, 0.85192698, 0.85446247, 0.87322515, 0.88286004,
       0.86663286, 0.88286004, 0.88488844, 0.88336714, 0.88691684,
       0.88488844, 0.88286004, 0.88438134, 0.88894523, 0.88133874,
       0.88083164, 0.88336714, 0.88539554, 0.89350913, 0.89452333,
       0.89350913, 0.89503043, 0.88032454, 0.88488844, 0.89756592,
       0.89350913, 0.89350913, 0.88793103, 0.87778905, 0.89097363,
       0.88640974, 0.89350913, 0.88995943, 0.87322515, 0.87880325,
       0.87829615, 0.87474645, 0.88793103, 0.89655172, 0.88843813,
       0.88438134, 0.89046653, 0.90111562, 0.89198783, 0.86916836,
       0.89148073, 0.87576065, 0.89097363, 0.87525355, 0.88995943,
       0.88438134, 0.89198783, 0.88235294, 0.87931034, 0.87474645,
```

```
       0.87322515, 0.87576065, 0.88539554, 0.87373225, 0.86916836,
       0.87322515, 0.88590264, 0.87018256, 0.87474645, 0.87373225,
       0.88083164, 0.89198783, 0.89300203, 0.89300203, 0.89198783,
       0.85395538, 0.87018256, 0.88184584, 0.87373225, 0.86409736,
       0.88539554, 0.88387424, 0.88336714, 0.89604462, 0.87525355,
       0.89452333, 0.88083164, 0.87423935, 0.88539554, 0.87981744,
       0.89046653, 0.87170385, 0.87221095, 0.88286004, 0.87677485,
       0.86967546, 0.87829615, 0.88032454, 0.88793103, 0.88793103,
       0.87626775, 0.89249493, 0.89908722, 0.88894523, 0.88590264,
       0.89756592, 0.90010142, 0.86511156, 0.87170385, 0.87119675,
       0.88387424, 0.86815416, 0.87271805, 0.87068966, 0.88488844,
       0.87271805, 0.87018256, 0.87271805, 0.88083164, 0.89198783,
       0.89249493, 0.89148073, 0.88995943, 0.87119675, 0.87170385,
       0.86967546, 0.88286004, 0.87880325, 0.88640974, 0.88286004,
       0.88894523, 0.88387424, 0.88742394, 0.88488844, 0.90212982,
       0.89046653, 0.89908722, 0.89452333, 0.88742394, 0.86967546,
       0.87626775, 0.88640974, 0.88843813, 0.87474645, 0.88590264,
       0.86866126, 0.88488844, 0.89756592, 0.88336714, 0.90162272,
       0.87322515, 0.89249493, 0.88286004, 0.89401623, 0.90314402]), 'mean_test_score': array([0.85908435, 0.8666
873 , 0.86820793, 0.87621737, 0.87134999,
       0.87317493, 0.87297214, 0.87996816, 0.88037235, 0.88057503,
       0.88077777, 0.88432654, 0.88929478, 0.88878783, 0.88919315,
       0.89000436, 0.85776538, 0.8680055 , 0.87439197, 0.88473227,
       0.87864996, 0.8743912 , 0.87591188, 0.8875711 , 0.88027314,
       0.88077849, 0.88463101, 0.88888987, 0.8826041 , 0.88919362,
       0.8882796 , 0.89040999, 0.85756295, 0.86749814, 0.87104635,
       0.88615241, 0.87165512, 0.87875169, 0.8819962 , 0.88513852,
       0.88179356, 0.87794105, 0.88148838, 0.88513826, 0.89142383,
       0.88706375, 0.89324861, 0.88209602, 0.89912845, 0.89912845,
       0.89912845, 0.89912845, 0.89892561, 0.89922982, 0.89892561,
       0.89892561, 0.89963519, 0.89963519, 0.89983803, 0.89963519,
       0.90085181, 0.90075045, 0.90075045, 0.90075045, 0.87489794,
       0.88189504, 0.87449169, 0.88828202, 0.87631622, 0.87449452,
       0.86750004, 0.88614886, 0.8812879 , 0.88503453, 0.87277007,
       0.88736919, 0.87611513, 0.87115034, 0.88665822, 0.86810507,
       0.8807768 , 0.89527485, 0.88067877, 0.89030939, 0.880475  ,
       0.88605114, 0.87378422, 0.88970102, 0.88371921, 0.8820987 ,
       0.8753049 , 0.88006608, 0.86080633, 0.87966277, 0.87743364,
       0.88118263, 0.88321164, 0.88483374, 0.88371854, 0.884023  ,
       0.88584743, 0.88452943, 0.8851378 , 0.88574647, 0.88402244,
       0.88361691, 0.88422538, 0.88564474, 0.88868662, 0.88929493,
       0.88838251, 0.88888951, 0.87996759, 0.8824009 , 0.88665966,
       0.88594967, 0.8819963 , 0.88159026, 0.8833125 , 0.88990278,
       0.88463116, 0.88919346, 0.88980131, 0.88868456, 0.88270439,
       0.88533992, 0.88402177, 0.89061206, 0.88088156, 0.88929432,
       0.88240085, 0.88686168, 0.88686276, 0.8792592 , 0.87672267,
       0.8861522 , 0.88625198, 0.88453026, 0.88016982, 0.89233553,
       0.88331317, 0.89537679, 0.88898959, 0.88361676, 0.86962745,
       0.8718574 , 0.87327681, 0.87865032, 0.87560808, 0.8746953 ,
       0.87662172, 0.8801709 , 0.8798652 , 0.88148756, 0.88077787,
       0.88452923, 0.88929468, 0.88959889, 0.89000436, 0.88949741,
       0.86688839, 0.86526814, 0.8717569 , 0.88128472, 0.87621532,
       0.88219822, 0.8785488 , 0.88067748, 0.88524035, 0.88037255,
       0.88615251, 0.88341417, 0.88909014, 0.89020632, 0.88645513,
       0.88412473, 0.87114766, 0.86891761, 0.87641996, 0.88037271,
       0.87722957, 0.87743318, 0.8802717 , 0.88797648, 0.88645595,
       0.88564382, 0.88250304, 0.88605161, 0.88909163, 0.88442837,
       0.88980208, 0.89294476, 0.86232794, 0.8678025 , 0.87074214,
       0.87631869, 0.8708432 , 0.87246556, 0.87347904, 0.87976532,
       0.88027093, 0.88047341, 0.87976409, 0.88432649, 0.8888892 ,
       0.8893961 , 0.88858505, 0.88838216, 0.86111213, 0.86759977,
       0.87053925, 0.87814322, 0.8759127 , 0.875508  , 0.87753501,
       0.88594921, 0.88665827, 0.88280663, 0.88807754, 0.88067938,
       0.88757126, 0.88888992, 0.89152504, 0.88473263, 0.86648451,
       0.86861391, 0.87844768, 0.88382042, 0.86780281, 0.87560932,
       0.87672262, 0.88564469, 0.88635556, 0.88037338, 0.88260533,
       0.88300793, 0.88807831, 0.88665817, 0.88919351, 0.89213413]), 'std_test_score': array([0.00820322, 0.00189
908, 0.00774812, 0.00749395, 0.00398263,
       0.00279504, 0.00758103, 0.00740794, 0.00638526, 0.00692404,
       0.00640644, 0.008697  , 0.00840694, 0.00855296, 0.00839169,
       0.00832434, 0.00292469, 0.01155355, 0.0092563 , 0.01024168,
       0.00794207, 0.0045643 , 0.00614702, 0.00445438, 0.01026993,
       0.00337151, 0.00955522, 0.01332314, 0.0109229 , 0.00814353,
       0.00645098, 0.00853267, 0.00576755, 0.00796347, 0.00586099,
       0.0081729 , 0.00597915, 0.00843184, 0.01035333, 0.00748507,
       0.01126939, 0.00778682, 0.0092744 , 0.00927264, 0.00707003,
       0.00355694, 0.00951311, 0.00675269, 0.00423347, 0.00423347,
       0.00423347, 0.00423347, 0.00410722, 0.00423856, 0.00410722,
       0.00410722, 0.00389586, 0.00389586, 0.00399286, 0.00389586,
       0.00450101, 0.00454396, 0.00454396, 0.00454396, 0.01442551,
       0.01597436, 0.01460701, 0.01198946, 0.01679368, 0.0224721 ,
       0.01223193, 0.01442251, 0.01917694, 0.01678418, 0.01438583,
       0.01705109, 0.01776894, 0.01913954, 0.00973006, 0.00942607,
       0.00988107, 0.00615417, 0.01773642, 0.01134382, 0.0094342 ,
```

```
       0.01134074, 0.02043832, 0.0206813 , 0.01936971, 0.01277604,
       0.01031395, 0.01631627, 0.0110087 , 0.01407965, 0.01271009,
       0.01025487, 0.00607091, 0.00476952, 0.00714396, 0.00696977,
       0.00356397, 0.0055316 , 0.00842498, 0.00805796, 0.0060314 ,
       0.00587805, 0.00491184, 0.00833845, 0.00804321, 0.00783564,
       0.00748562, 0.00821725, 0.01074367, 0.00561136, 0.01025973,
       0.00626033, 0.01032256, 0.00821768, 0.00366508, 0.00483141,
       0.00605838, 0.00917756, 0.00727145, 0.01040922, 0.00468413,
       0.01461284, 0.00720847, 0.00365472, 0.01095904, 0.00277208,
       0.00973107, 0.00497239, 0.0102408 , 0.01455152, 0.0121994 ,
       0.00821932, 0.00676338, 0.01036742, 0.00616437, 0.00404346,
       0.00938494, 0.00976151, 0.00675859, 0.01168871, 0.00371248,
       0.00295627, 0.00499063, 0.00710353, 0.00160473, 0.00460998,
       0.00716384, 0.00647731, 0.00678846, 0.0055988 , 0.00582449,
       0.0087216 , 0.00914578, 0.00885265, 0.00883921, 0.00818807,
       0.00845268, 0.00703131, 0.01139051, 0.0094728 , 0.00949646,
       0.00254174, 0.00681038, 0.00576978, 0.00784136, 0.00888496,
       0.00535292, 0.01273214, 0.01132009, 0.00717272, 0.00801111,
       0.01039818, 0.00871564, 0.00801236, 0.0060048 , 0.00248029,
       0.0056561 , 0.00580716, 0.00330626, 0.00814743, 0.01054266,
       0.00832138, 0.00739557, 0.00889967, 0.00634203, 0.01075014,
       0.00870841, 0.00461182, 0.00707296, 0.0034881 , 0.00703674,
       0.00733969, 0.00397918, 0.00294719, 0.00776846, 0.00727619,
       0.00643742, 0.00789783, 0.00697282, 0.00869772, 0.00844914,
       0.00829575, 0.00874226, 0.00797566, 0.00650996, 0.00584391,
       0.01345116, 0.01024024, 0.002518  , 0.0069061 , 0.00540555,
       0.00377029, 0.00336556, 0.01346473, 0.0052976 , 0.01363944,
       0.00532324, 0.0103245 , 0.00649369, 0.00587585, 0.01235603,
       0.0096248 , 0.01075944, 0.00938071, 0.00964162, 0.00840367,
       0.0092711 , 0.00810166, 0.00760689, 0.01334182, 0.01091256,
       0.00994156, 0.00727073, 0.00772314, 0.0071935 , 0.00877462]), 'rank_test_score': array([238, 232, 223, 18
8, 213, 206, 207, 164, 157, 151, 146, 106,  39,
        53,  45,  29, 239, 225, 201,  99, 173, 202, 192,  65, 158, 144,
       101,  50, 126,  42,  60,  26, 240, 230, 216,  79, 212, 171, 134,
        93, 136, 177, 138,  94,  24,  67,  19, 132,  10,  10,  10,  10,
        14,   9,  14,  14,   6,   6,   5,   6,   1,   2,   2,   2, 197,
       135, 200,  59, 187, 199, 229,  81, 140,  96, 208,  66, 190, 214,
        72, 224, 147,  18, 149,  27, 152,  83, 203,  34, 114, 131, 196,
       163, 237, 169, 179, 142, 121,  97, 115, 110,  86, 103,  95,  87,
       111, 116, 108,  88,  54,  38,  57,  51, 165, 128,  70,  84, 133,
       137, 120,  31, 100,  44,  33,  55, 124,  91, 112,  25, 143,  41,
       129,  69,  68, 170, 182,  80,  77, 102, 162,  21, 119,  17,  48,
       117, 220, 210, 205, 172, 194, 198, 184, 161, 166, 139, 145, 104,
        40,  35,  29,  36, 231, 234, 211, 141, 189, 130, 174, 150,  92,
       156,  78, 118,  47,  28,  75, 109, 215, 221, 185, 155, 181, 180,
       159,  63,  74,  90, 127,  82,  46, 105,  32,  20, 235, 227, 218,
       186, 217, 209, 204, 167, 160, 153, 168, 107,  52,  37,  56,  58,
       236, 228, 219, 176, 191, 195, 178,  85,  71, 123,  62, 148,  64,
        49,  23,  98, 233, 222, 175, 113, 226, 193, 183,  89,  76, 154,
       125, 122,  61,  73,  43,  22], dtype=int32)}
Best parameters:  {'max_depth': 5, 'max_features': None, 'min_samples_leaf': 10, 'min_samples_split': 2}
              precision    recall  f1-score   support

           0       0.90      0.97      0.94      2055
           1       0.75      0.49      0.59       411

    accuracy                           0.89      2466
   macro avg       0.83      0.73      0.77      2466
weighted avg       0.88      0.89      0.88      2466

[[1989   66]
 [ 209  202]]
```

In [31]:

```
cm_grid = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(10,7))
sns.heatmap(cm_grid, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Truth')
plt.title('Confusion Matrix for GridSearchCV with New Features')
plt.show()

```



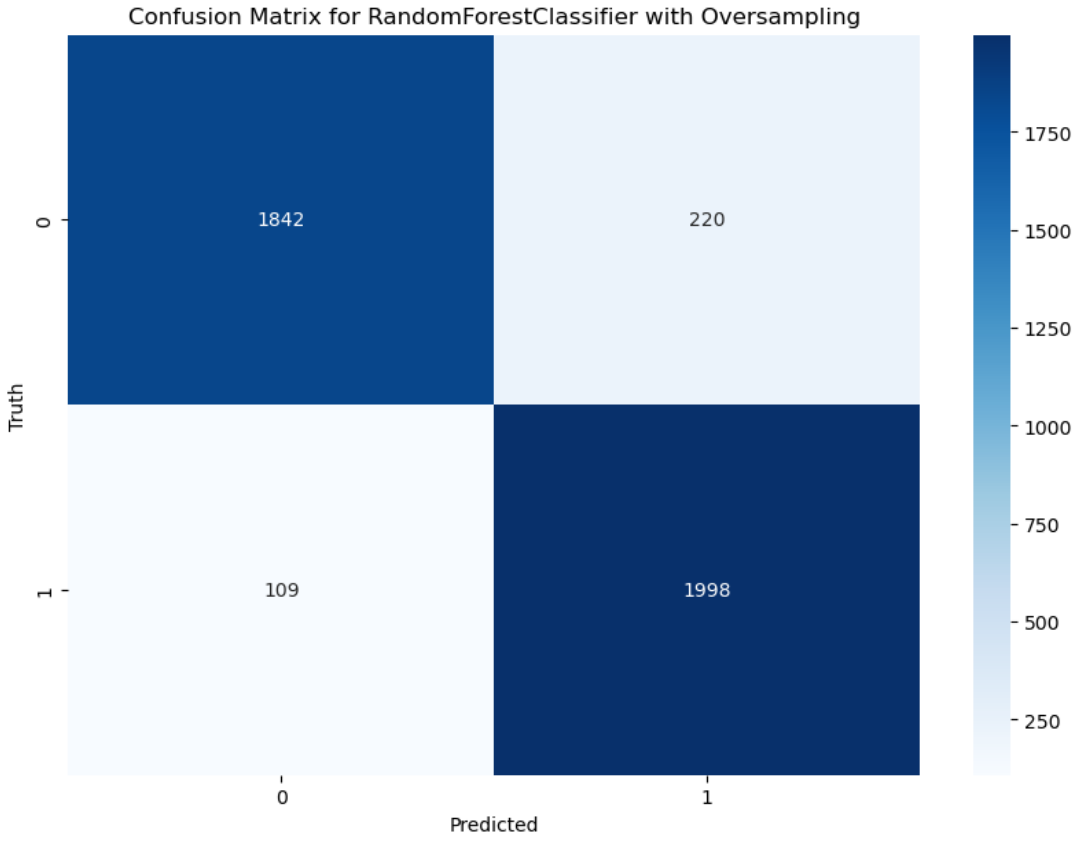
In [32]:

```
from imblearn.over_sampling import SMOTE
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

smote = SMOTE(random_state=42)
X_res, y_res = smote.fit_resample(X, y)

X_train_res, X_test_res, y_train_res, y_test_res = train_test_split(X_res, y_res, test_size=0.2, random_state=42

rfc = RandomForestClassifier(n_estimators=100, random_state=42)

rfc.fit(X_train_res, y_train_res)

y_pred_res = rfc.predict(X_test_res)

print(classification_report(y_test_res, y_pred_res))

print(confusion_matrix(y_test_res, y_pred_res))

print("Accuracy: ", accuracy_score(y_test_res, y_pred_res))

```

```
              precision    recall  f1-score   support

           0       0.94      0.89      0.92      2062
           1       0.90      0.95      0.92      2107

    accuracy                           0.92      4169
   macro avg       0.92      0.92      0.92      4169
weighted avg       0.92      0.92      0.92      4169

[[1842  220]
 [ 109 1998]]
Accuracy:  0.9210841928520028
```

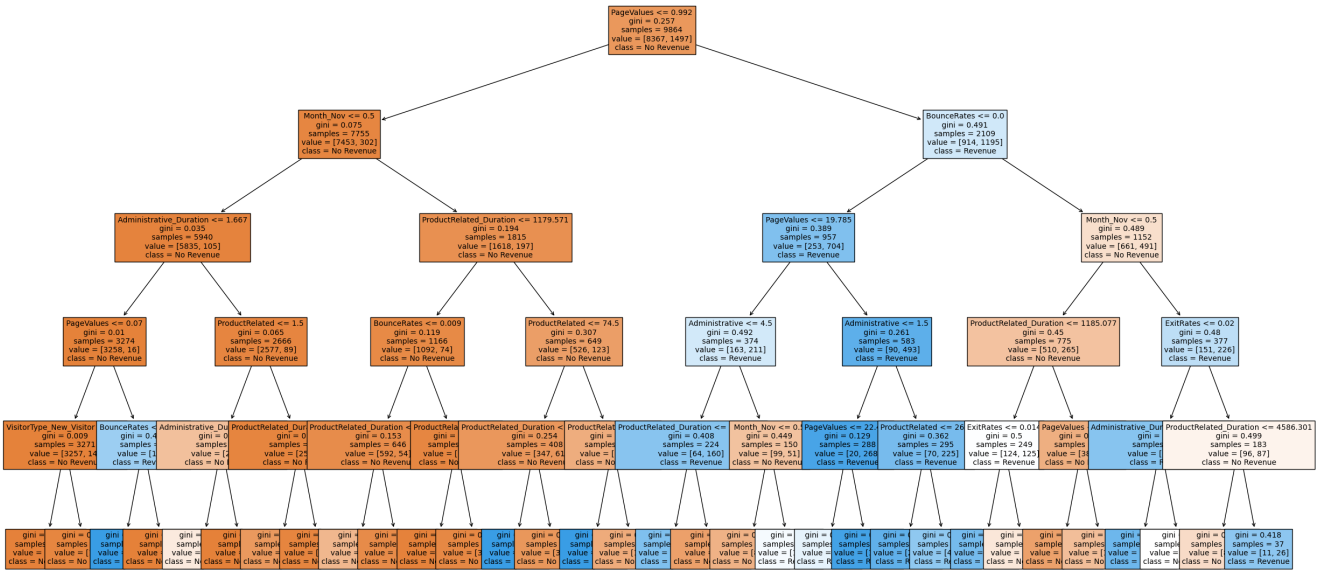
In [33]:

```
cm_rfc = confusion_matrix(y_test_res, y_pred_res)

plt.figure(figsize=(10,7))
sns.heatmap(cm_rfc, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Truth')
plt.title('Confusion Matrix for RandomForestClassifier with Oversampling')
plt.show()

```



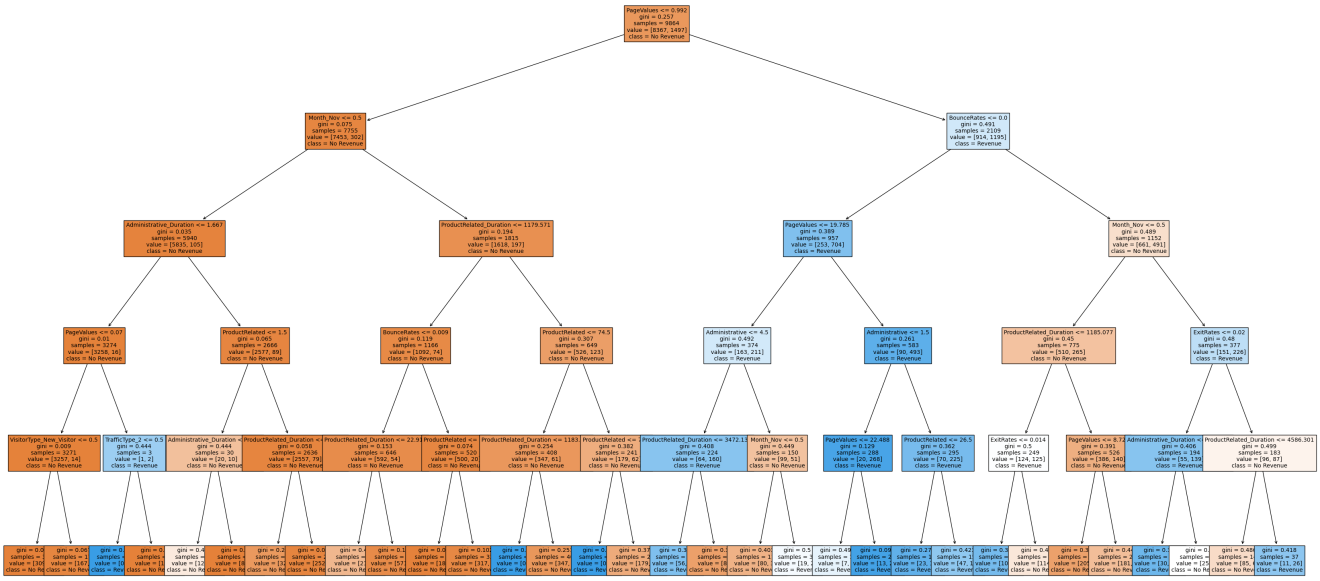
In [34]:

```
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt

dtree = DecisionTreeClassifier(max_depth=5)

dtree.fit(X_train, y_train)

plt.figure(figsize=(30,15))
plot_tree(dtree, feature_names=X_train.columns, class_names=['No Revenue', 'Revenue'], filled=True, fontsize=10
plt.show()

```



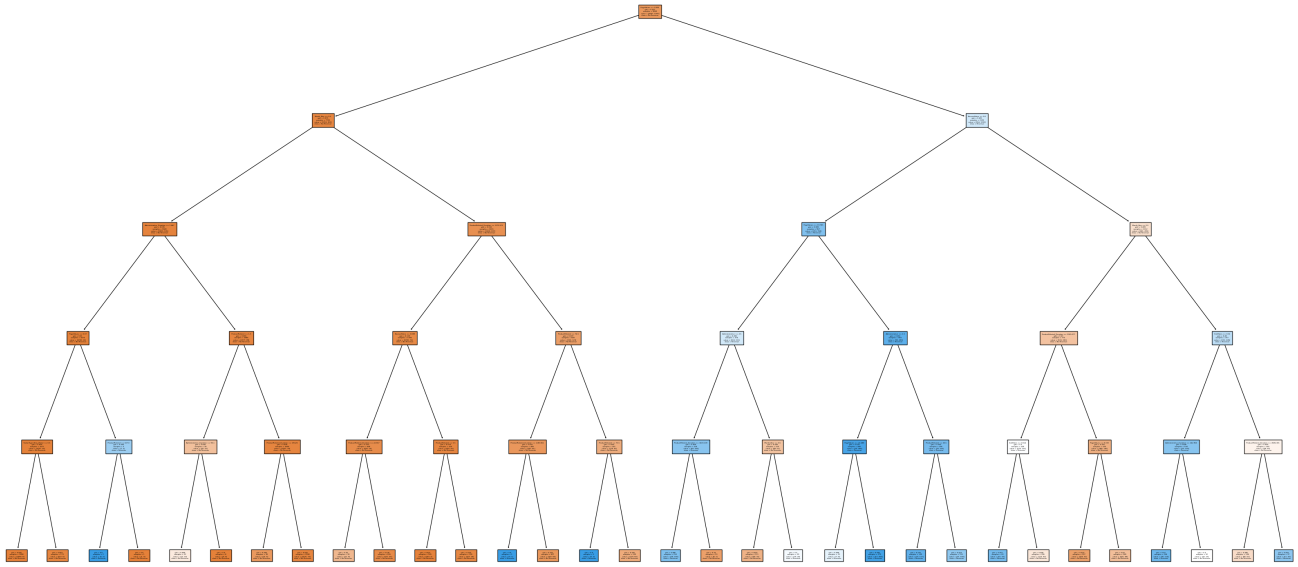
In [35]:

```
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt

dtree = DecisionTreeClassifier(max_depth=5)

dtree.fit(X_train, y_train)

plt.figure(figsize=(40,20))
plot_tree(dtree, feature_names=X_train.columns, class_names=['No Revenue', 'Revenue'], filled=True, fontsize=10
plt.subplots_adjust(left=0.1, right=0.9, top=0.9, bottom=0.1)
plt.show()

```



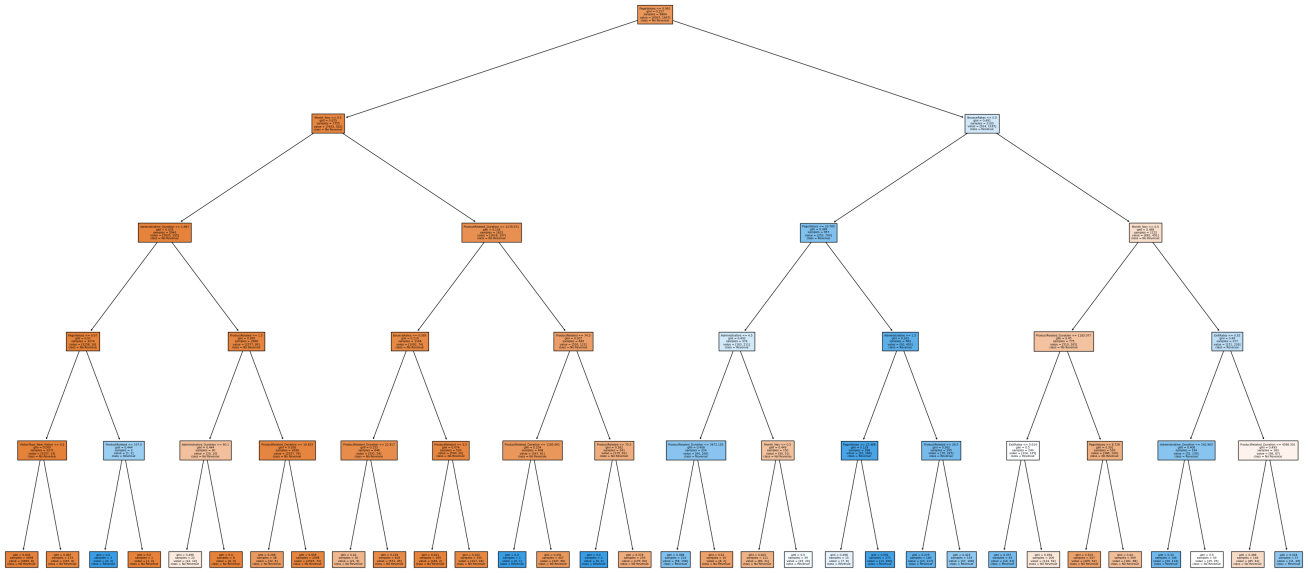
In [36]:

```
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt

dtree = DecisionTreeClassifier(max_depth=5)

dtree.fit(X_train, y_train)

plt.figure(figsize=(40,20))
plot_tree(dtree, feature_names=X_train.columns, class_names=['No Revenue', 'Revenue'], filled=True, fontsize=3)
plt.subplots_adjust(left=0.1, right=0.9, top=0.9, bottom=0.1)
plt.show()

```



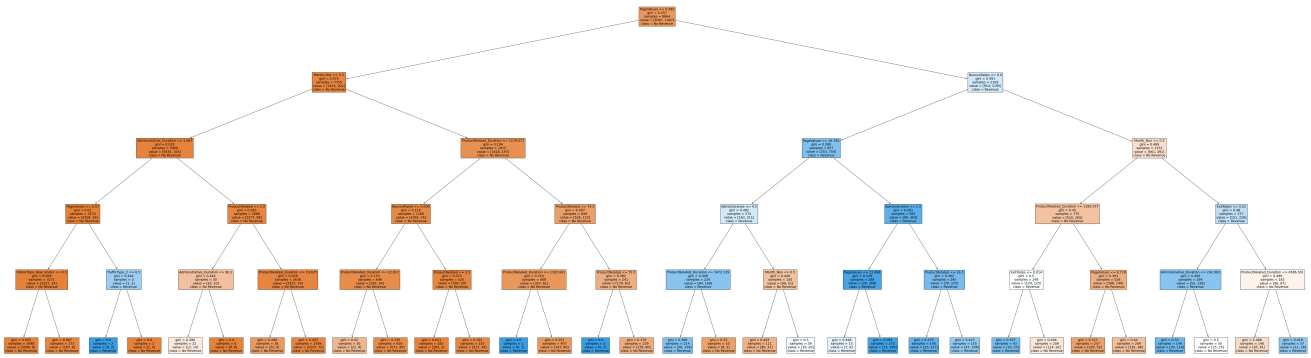
In [37]:

```
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt

dtree = DecisionTreeClassifier(max_depth=5)

dtree.fit(X_train, y_train)

plt.figure(figsize=(40,20))
plot_tree(dtree, feature_names=X_train.columns, class_names=['No Revenue', 'Revenue'], filled=True, fontsize=5)
plt.subplots_adjust(left=0.1, right=0.9, top=0.9, bottom=0.1)
plt.savefig("decision_tree.png", dpi=500)

plt.show()

```



In [38]:

```
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt

dtree = DecisionTreeClassifier(max_depth=5)

dtree.fit(X_train, y_train)

plt.figure(figsize=(100,30))
plot_tree(dtree, feature_names=X_train.columns, class_names=['No Revenue', 'Revenue'], filled=True, fontsize=14
plt.subplots_adjust(left=0.1, right=0.9, top=0.9, bottom=0.1)

plt.savefig("decision_tree.png", dpi=600)

plt.show()

```



In [39]:

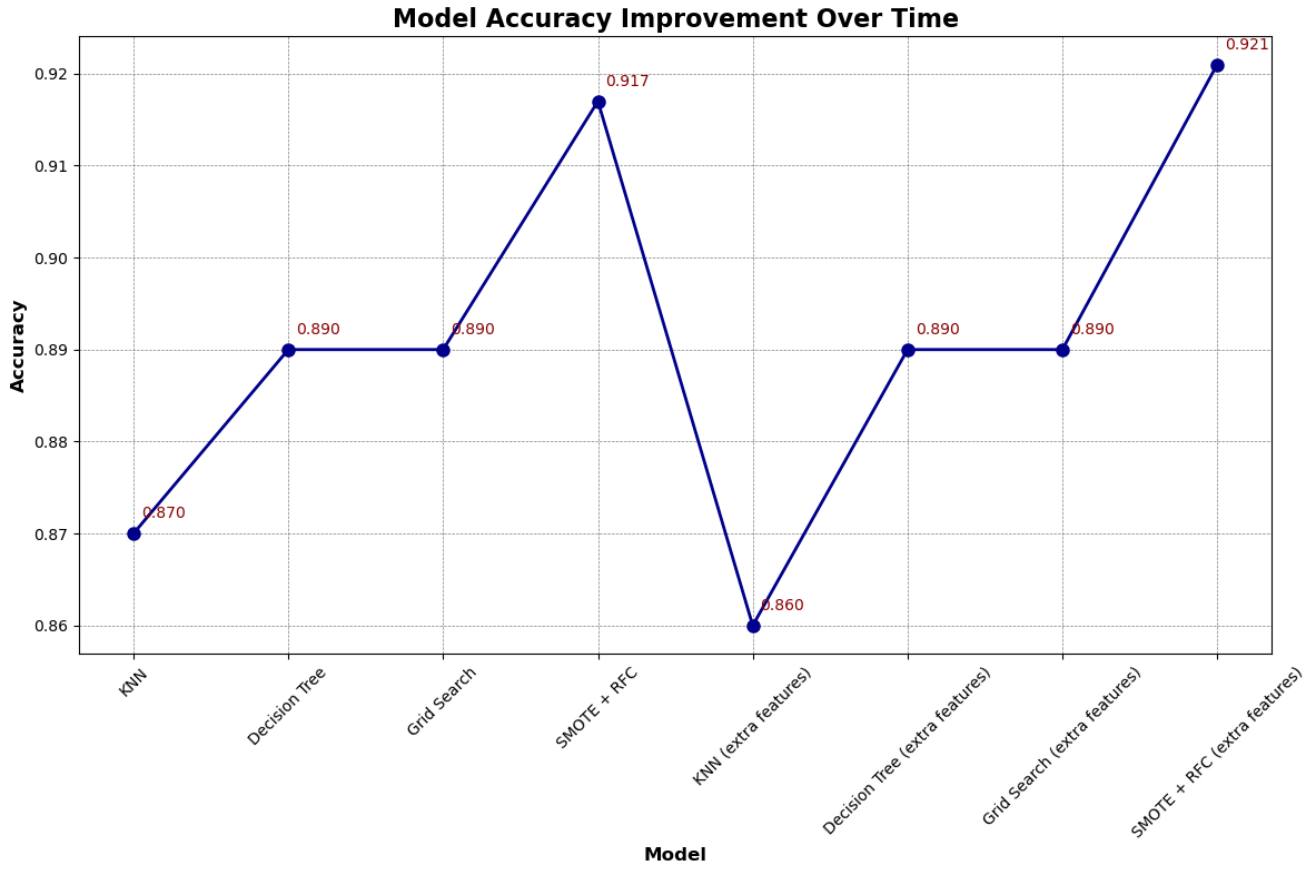
```
from sklearn.tree import DecisionTreeClassifier, export_text

dtree = DecisionTreeClassifier(max_depth=5)

dtree.fit(X_train, y_train)

tree_text = export_text(dtree, feature_names=list(X_train.columns))
print(tree_text)

```

```
|--- PageValues <= 0.99
|   |--- Month_Nov <= 0.50
|   |   |--- Administrative_Duration <= 1.67
|   |   |   |--- PageValues <= 0.07
|   |   |   |   |--- VisitorType_New_Visitor <= 0.50
|   |   |   |   |   |--- class: 0
|   |   |   |   |--- VisitorType_New_Visitor >  0.50
|   |   |   |   |   |--- class: 0
|   |   |   |--- PageValues >  0.07
|   |   |   |   |--- ProductRelated <= 107.00
|   |   |   |   |   |--- class: 1
|   |   |   |   |--- ProductRelated >  107.00
|   |   |   |   |   |--- class: 0
|   |   |--- Administrative_Duration >  1.67
|   |   |   |--- ProductRelated <= 1.50
|   |   |   |   |--- Administrative_Duration <= 90.10
|   |   |   |   |   |--- class: 0
|   |   |   |   |--- Administrative_Duration >  90.10
|   |   |   |   |   |--- class: 0
|   |   |   |--- ProductRelated >  1.50
|   |   |   |   |--- ProductRelated_Duration <= 19.62
|   |   |   |   |   |--- class: 0
|   |   |   |   |--- ProductRelated_Duration >  19.62
|   |   |   |   |   |--- class: 0
|   |--- Month_Nov >  0.50
|   |   |--- ProductRelated_Duration <= 1179.57
|   |   |   |--- BounceRates <= 0.01
|   |   |   |   |--- ProductRelated_Duration <= 22.92
|   |   |   |   |   |--- class: 0
|   |   |   |   |--- ProductRelated_Duration >  22.92
|   |   |   |   |   |--- class: 0
|   |   |   |--- BounceRates >  0.01
|   |   |   |   |--- ProductRelated <= 5.50
|   |   |   |   |   |--- class: 0
|   |   |   |   |--- ProductRelated >  5.50
|   |   |   |   |   |--- class: 0
|   |   |--- ProductRelated_Duration >  1179.57
|   |   |   |--- ProductRelated <= 74.50
|   |   |   |   |--- ProductRelated_Duration <= 1183.06
|   |   |   |   |   |--- class: 1
|   |   |   |   |--- ProductRelated_Duration >  1183.06
|   |   |   |   |   |--- class: 0
|   |   |   |--- ProductRelated >  74.50
|   |   |   |   |--- ProductRelated <= 75.50
|   |   |   |   |   |--- class: 1
|   |   |   |   |--- ProductRelated >  75.50
|   |   |   |   |   |--- class: 0
|--- PageValues >  0.99
|   |--- BounceRates <= 0.00
|   |   |--- PageValues <= 19.78
|   |   |   |--- Administrative <= 4.50
|   |   |   |   |--- ProductRelated_Duration <= 3472.14
|   |   |   |   |   |--- class: 1
|   |   |   |   |--- ProductRelated_Duration >  3472.14
|   |   |   |   |   |--- class: 0
|   |   |   |--- Administrative >  4.50
|   |   |   |   |--- Month_Nov <= 0.50
|   |   |   |   |   |--- class: 0
|   |   |   |   |--- Month_Nov >  0.50
|   |   |   |   |   |--- class: 1
|   |   |--- PageValues >  19.78
|   |   |   |--- Administrative <= 1.50
|   |   |   |   |--- PageValues <= 22.49
|   |   |   |   |   |--- class: 1
|   |   |   |   |--- PageValues >  22.49
|   |   |   |   |   |--- class: 1
|   |   |   |--- Administrative >  1.50
|   |   |   |   |--- ProductRelated <= 26.50
|   |   |   |   |   |--- class: 1
|   |   |   |   |--- ProductRelated >  26.50
|   |   |   |   |   |--- class: 1
|   |--- BounceRates >  0.00
|   |   |--- Month_Nov <= 0.50
|   |   |   |--- ProductRelated_Duration <= 1185.08
|   |   |   |   |--- ExitRates <= 0.01
|   |   |   |   |   |--- class: 1
|   |   |   |   |--- ExitRates >  0.01
|   |   |   |   |   |--- class: 0
|   |   |   |--- ProductRelated_Duration >  1185.08
|   |   |   |   |--- PageValues <= 8.73
|   |   |   |   |   |--- class: 0
|   |   |   |   |--- PageValues >  8.73
|   |   |   |   |   |--- class: 0
```

```
|   |   |--- Month_Nov >  0.50
|   |   |   |--- ExitRates <= 0.02
|   |   |   |   |--- Administrative_Duration <= 242.96
|   |   |   |   |   |--- class: 1
|   |   |   |   |--- Administrative_Duration >  242.96
|   |   |   |   |   |--- class: 0
|   |   |   |--- ExitRates >  0.02
|   |   |   |   |--- ProductRelated_Duration <= 4586.30
|   |   |   |   |   |--- class: 0
|   |   |   |   |--- ProductRelated_Duration >  4586.30
|   |   |   |   |   |--- class: 1
```

In [40]:

```python
import matplotlib.pyplot as plt

accuracy_values = [0.87, 0.89, 0.89, 0.917, 0.86, 0.89, 0.89, 0.921]

models = ['KNN', 'Decision Tree', 'Grid Search', 'SMOTE + RFC', 'KNN (extra features)', 'Decision Tree (extra fe

plt.figure(figsize=(12, 8))
plt.plot(models, accuracy_values, marker='o', linestyle='-', color='darkblue', linewidth=2, markersize=8)

for i, txt in enumerate(accuracy_values):
    plt.annotate(f'{txt:.3f}', (models[i], accuracy_values[i]), xytext=(5,10), textcoords='offset points', fonts

plt.title('Model Accuracy Improvement Over Time', fontsize=16, fontweight='bold')
plt.xlabel('Model', fontsize=12, fontweight='bold')
plt.ylabel('Accuracy', fontsize=12, fontweight='bold')

plt.xticks(rotation=45)

plt.grid(color='gray', linestyle='--', linewidth=0.5)

plt.tight_layout()
plt.show()

```



In [ ]:

```python

```

In [ ]:

```python

```