```
# Importing the required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns


import warnings
warnings.filterwarnings('ignore')


# getting the data from UC Irving machine learning repository
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/00468/online_shoppers_intention.csv"
df = pd.read_csv(url)
```

```
df.head()
```

	Administrative	Administrative_Duration	Informational	Informational_Duration	ProductRelated	ProductRelated_Duration	BounceRa
0	0	0.0	0	0.0	1	0.000000	
1	0	0.0	0	0.0	2	64.000000	
2	0	0.0	0	0.0	1	0.000000	
3	0	0.0	0	0.0	2	2.666667	
4	0	0.0	0	0.0	10	627.500000	

Next steps: Generate code with df | View recommended plots | New interactive sheet

Category	Column Name	Description	Values
Administrative	`Administrative`	Number of administrative pages visited	Integer
	`Administrative_Duration`	Time spent on administrative pages (seconds)	Float
Informational	`Informational`	Number of informational pages visited	Integer
	`Informational_Duration`	Time spent on informational pages (seconds)	Float
Product-Related	`ProductRelated`	Number of product pages visited	Integer
	`ProductRelated_Duration`	Time spent on product pages (seconds)	Float
User Engagement	`BounceRates`	Percentage of visitors who left immediately	Float (0–1)
	`ExitRates`	Percentage of pageviews that were last in session	Float (0–1)
	`PageValues`	Average value of pages viewed before purchase	Float (≥ 0)
Traffic Source	`TrafficType`	Source of traffic (20 categories)	Integer (1–20)
Visitor Type	`VisitorType`	Type of visitor	'New_Visitor', 'Returning_Visitor', 'Other'
Weekend	`Weekend`	Whether session occurred on weekend	Boolean (True/False)
Temporal	`Month`	Month of the year	Abbreviated (e.g., 'Feb', 'Nov')
Special Day	`SpecialDay`	Proximity to special shopping day	Float (0–1, where 1 = day of event)
Target Variable	`Revenue`	Whether purchase occurred	Boolean (True/False)

```
df.info()
```

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

```
df.shape
```

```
(12330, 18)
```

```
df.isnull().sum()
```

	0
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

```
duplicates = df[df.duplicated()]
duplicates.shape
print(duplicates)
```

```
       Administrative  Administrative_Duration  Informational  \
158                 0                      0.0              0   
159                 0                      0.0              0   
178                 0                      0.0              0   
418                 0                      0.0              0   
456                 0                      0.0              0   
...               ...                      ...            ...   
11934               0                      0.0              0   
11938               0                      0.0              0   
12159               0                      0.0              0   
12180               0                      0.0              0   
12185               0                      0.0              0   

       Informational_Duration  ProductRelated  ProductRelated_Duration  \
158                       0.0               1                      0.0   
159                       0.0               1                      0.0   
178                       0.0               1                      0.0   
418                       0.0               1                      0.0   
456                       0.0               1                      0.0   
...                       ...             ...                      ...   
11934                     0.0               1                      0.0   
11938                     0.0               1                      0.0   
12159                     0.0               1                      0.0   
12180                     0.0               1                      0.0   
12185                     0.0               1                      0.0   

       BounceRates  ExitRates  PageValues  SpecialDay Month  OperatingSystems  \
158            0.2        0.2         0.0         0.0   Feb                 1   
159            0.2        0.2         0.0         0.0   Feb                 3   
178            0.2        0.2         0.0         0.0   Feb                 3   
418            0.2        0.2         0.0         0.0   Mar                 1   
456            0.2        0.2         0.0         0.0   Mar                 2   
...            ...        ...         ...         ...   ...               ...   
11934          0.2        0.2         0.0         0.0   Dec                 1   
11938          0.2        0.2         0.0         0.0   Dec                 1   
12159          0.2        0.2         0.0         0.0   Dec                 1   
12180          0.2        0.2         0.0         0.0   Dec                 1   
12185          0.2        0.2         0.0         0.0   Dec                 8   

       Browser  Region  TrafficType        VisitorType  Weekend  Revenue
```

```
158          1       1            3  Returning_Visitor    False    False
159          2       3            3  Returning_Visitor    False    False
178          2       3            3  Returning_Visitor    False    False
418          1       1            1  Returning_Visitor     True    False
456          2       4            1  Returning_Visitor    False    False
...        ...     ...          ...                ...      ...      ...
11934        1       1            2        New_Visitor    False    False
11938        1       4            1  Returning_Visitor     True    False
12159        1       1            3  Returning_Visitor    False    False
12180       13       9           20  Returning_Visitor    False    False
12185       13       9           20              Other    False    False

[125 rows x 18 columns]
```

```
df.describe()
```

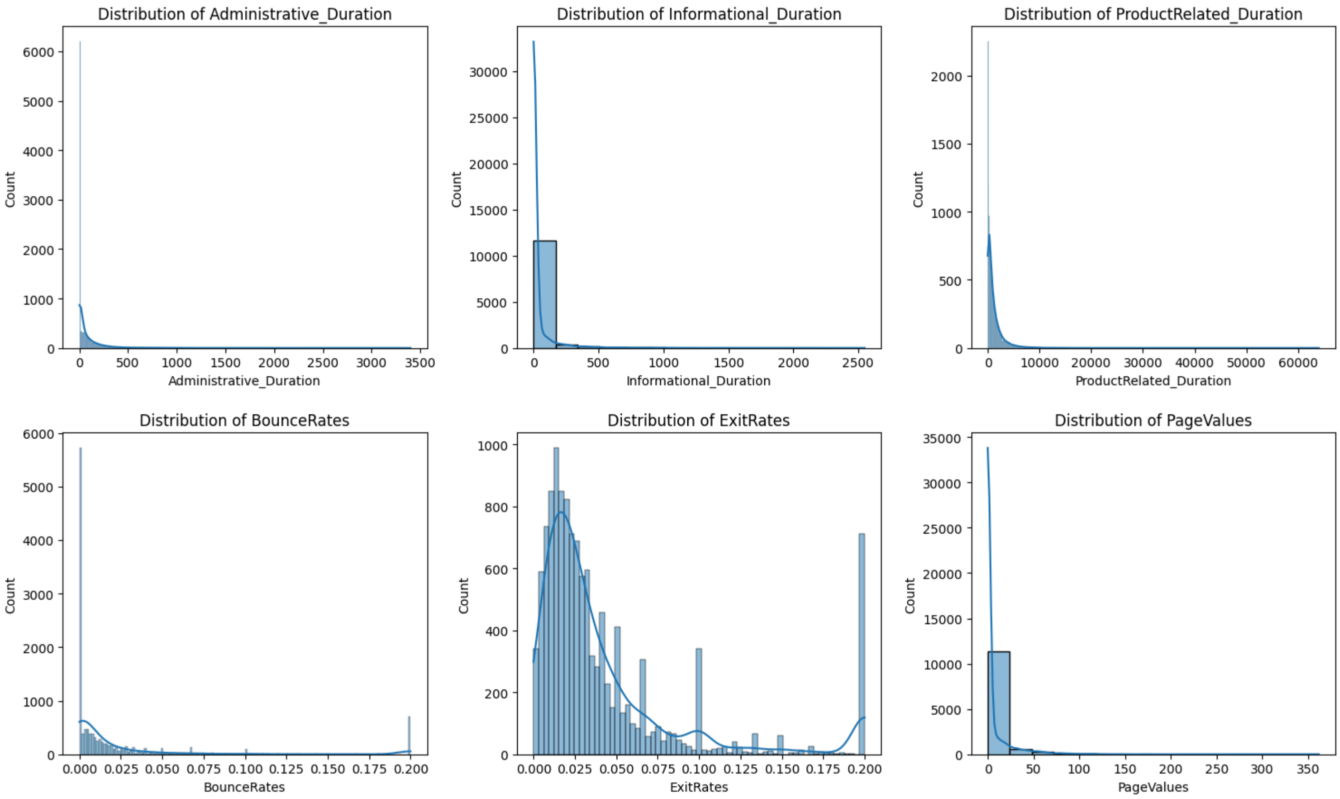
	Administrative	Administrative_Duration	Informational	Informational_Duration	ProductRelated	ProductRelated_Duration	Bou
count	12330.000000	12330.000000	12330.000000	12330.000000	12330.000000	12330.000000	1233
mean	2.315166	80.818611	0.503569	34.472398	31.731468	1194.746220	
std	3.321784	176.779107	1.270156	140.749294	44.475503	1913.669288	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.000000	0.000000	0.000000	0.000000	7.000000	184.137500	
50%	1.000000	7.500000	0.000000	0.000000	18.000000	598.936905	
75%	4.000000	93.256250	0.000000	0.000000	38.000000	1464.157214	
max	27.000000	3398.750000	24.000000	2549.375000	705.000000	63973.522230	

```
df.dtypes
```

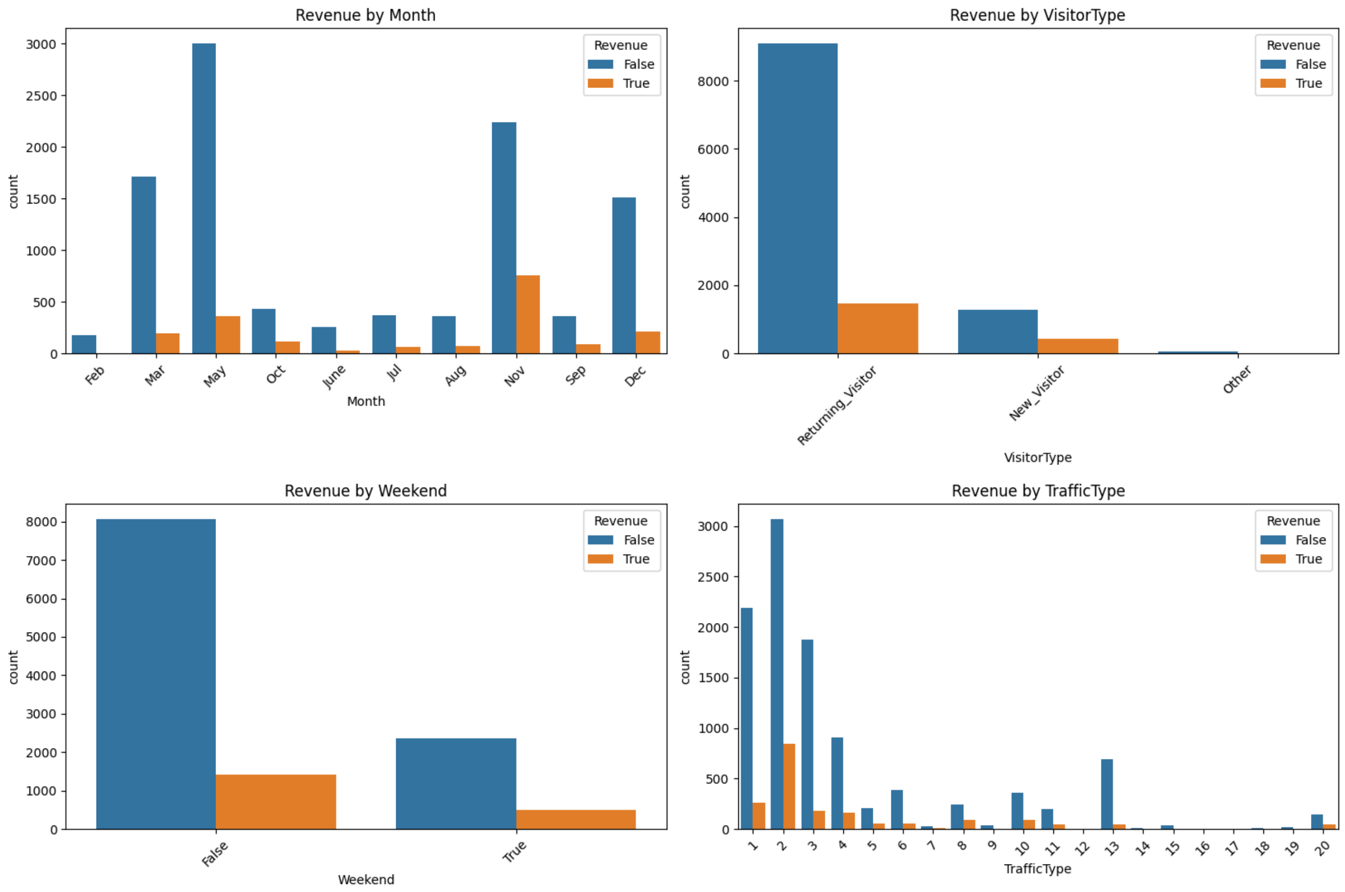
	0
Administrative	int64
Administrative_Duration	float64
Informational	int64
Informational_Duration	float64
ProductRelated	int64
ProductRelated_Duration	float64
BounceRates	float64
ExitRates	float64
PageValues	float64
SpecialDay	float64
Month	object
OperatingSystems	int64
Browser	int64
Region	int64
TrafficType	int64
VisitorType	object
Weekend	bool
Revenue	bool

dtype: object

```
# Numerical fearures
numericalColumns = [
    'Administrative_Duration',
    'Informational_Duration',
    'ProductRelated_Duration',
    'BounceRates',
    'ExitRates',
    'PageValues'
]
plt.figure(figsize=(15, 9))
for i, col in enumerate(numericalColumns, 1):
    plt.subplot(2, 3, i)
    sns.histplot(df[col], kde=True)
    sns.histplot(df[col], kde=True)
    plt.title(f'Distribution of {col}')
plt.tight_layout(h_pad=2)
plt.show()
```



```
# Revenue distribution
categoricalColumns = [
    'Month',
    'VisitorType',
    'Weekend',
    'TrafficType'
]

plt.figure(figsize=(15, 10))
for i, col in enumerate(categoricalColumns, 1):
    plt.subplot(2, 2, i)
    sns.countplot(x=col, hue='Revenue', data=df)
    plt.title(f'Revenue by {col}')
    plt.xticks(rotation=45)
plt.tight_layout(h_pad=1.5)
plt.show()
```

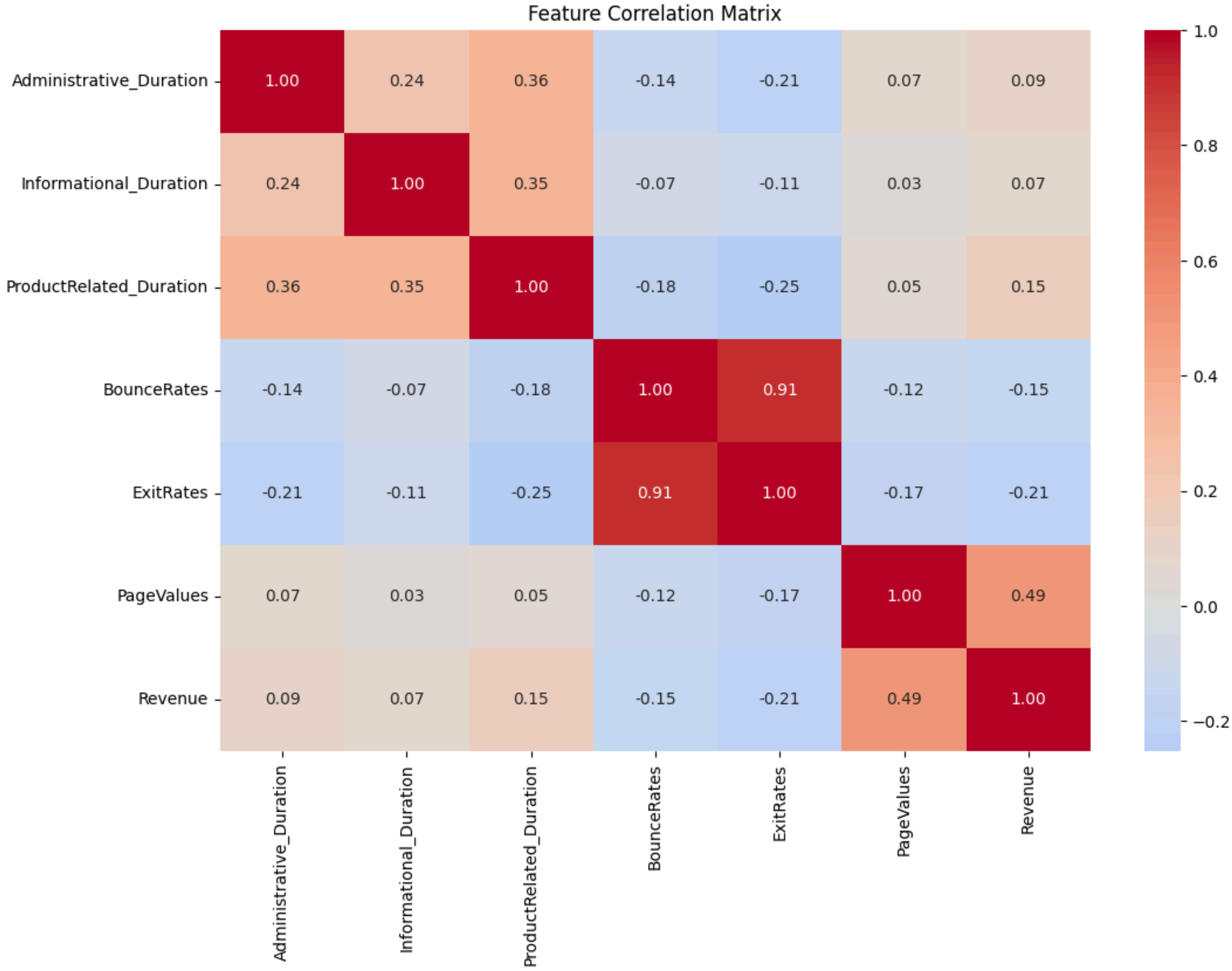


```
# Correlation
corr = df[numericalColumns +['Revenue']].corr()
corr
```

	Administrative_Duration	Informational_Duration	ProductRelated_Duration	BounceRates	ExitRates	PageValues
Administrative_Duration	1.000000	0.238031	0.355422	-0.144170	-0.205798	0.06760
Informational_Duration	0.238031	1.000000	0.347364	-0.074067	-0.105276	0.03086
ProductRelated_Duration	0.355422	0.347364	1.000000	-0.184541	-0.251984	0.05282
BounceRates	-0.144170	-0.074067	-0.184541	1.000000	0.913004	-0.11938
ExitRates	-0.205798	-0.105276	-0.251984	0.913004	1.000000	-0.17449
PageValues	0.067608	0.030861	0.052823	-0.119386	-0.174498	1.00000
Revenue	0.093587	0.070345	0.152373	-0.150673	-0.207071	0.49256

Next steps: Generate code with corr View recommended plots New interactive sheet

```
plt.figure(figsize=(12, 8))
sns.heatmap(corr, annot=True, cmap='coolwarm', center=0, fmt='.2f')
plt.title('Feature Correlation Matrix')
plt.show()
```



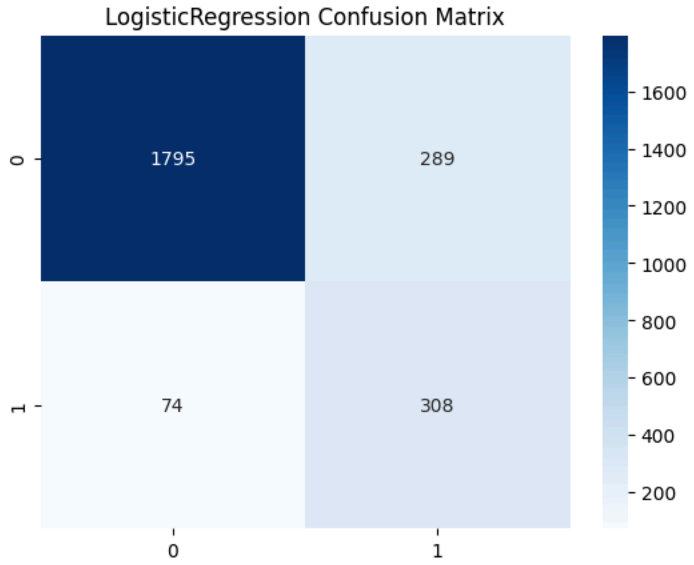
```
# Feature engineering
df['Total_Duration'] = df['Administrative'] + df['Administrative_Duration'] + df['ProductRelated_Duration']
df['PageValue_to_ExitRatio']= np.where(df['ExitRates']>0, df['PageValues']/df['ExitRates'], 0)
df['BounceExit_Interaction'] = df['BounceRates'] * df['ExitRates']
```

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['Revenue'] = le.fit_transform(df['Revenue'])
df
```

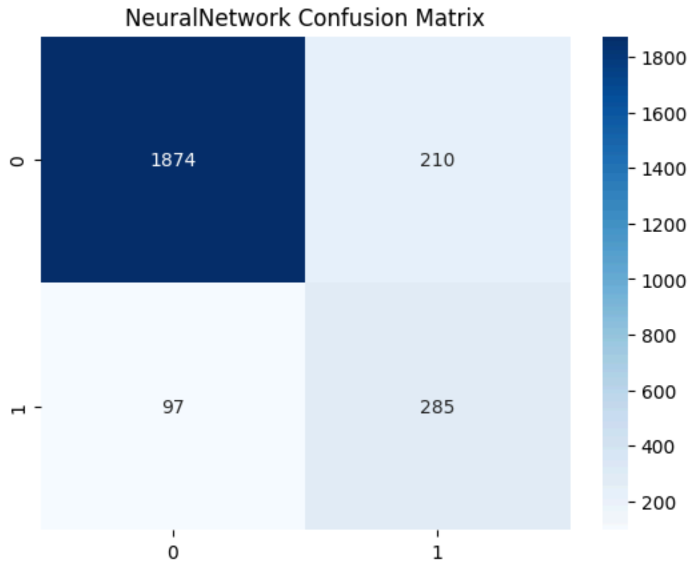
```
       Administrative  Administrative_Duration  Informational  Informational_Duration  ProductRelated  ProductRelated_Duration  Bour
    0               0                      0.0              0                     0.0               1                 0.000000
    1               0                      0.0              0                     0.0               2                64.000000
    2               0                      0.0              0                     0.0               1                 0.000000
    3               0                      0.0              0                     0.0               2                 2.666667
    4               0                      0.0              0                     0.0              10               627.500000
  ...             ...                      ...            ...                     ...             ...                      ...
12325               3                    145.0              0                     0.0              53              1783.791667
12326               0                      0.0              0                     0.0               5               465.750000
12327               0                      0.0              0                     0.0               6               184.250000
12328               4                     75.0              0                     0.0              15               346.000000
12329               0                      0.0              0                     0.0               3                21.250000
12330 rows × 21 columns
```

```
# Machine learning pipeline
from sklearn.model_selection import train_test_split, RandomizedSearchCV, StratifiedKFold
from sklearn.preprocessing import PowerTransformer, OneHotEncoder, StandardScaler
from sklearn.compose import ColumnTransformer
from sklearn.metrics import classification_report, roc_auc_score, precision_recall_curve, average_precision_score, confusion_matrix, acc
import joblib
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
from imblearn.over_sampling import SMOTE
from imblearn.pipeline import Pipeline
from sklearn.tree import DecisionTreeRegressor, DecisionTreeClassifier, plot_tree
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier


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


# Splitting the data
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42, stratify=y)


numericalFeatures = [
    'Administrative_Duration',
    'Informational_Duration',
    'ProductRelated_Duration',
    'BounceRates',
    'ExitRates',
    'PageValues',
    'Total_Duration',
    'PageValue_to_ExitRatio',
    'BounceExit_Interaction'
]

categoricalFeatures = [
    'Month',
    'VisitorType',
    'Weekend',
    'TrafficType',
]


preprocessor = ColumnTransformer(
    transformers=[
        ('num', Pipeline(steps=[
            ('power', PowerTransformer()),
            ('scaler', StandardScaler())
        ]), numericalFeatures),
        ('cat', OneHotEncoder(handle_unknown='ignore'), categoricalFeatures)
    ]
)
smote = SMOTE(sampling_strategy=0.5, random_state=42)


models = {
    'LogisticRegression': {
        'model': LogisticRegression(max_iter=1000, class_weight='balanced'),
        'params': {
            'model__C': [0.1, 1, 10],
            'model__solver': ['lbfgs', 'liblinear']
        }
    },
    'NeuralNetwork': {
        'model': MLPClassifier(early_stopping=True),
        'params':{
            'model__hidden_layer_sizes': [(50,), (100,)],
            'model__alpha': [0.0001, 0.001],
            'model__learning_rate_init': [0.001, 0.01]
        }
    },
    'RandomForest': {
        'model': RandomForestClassifier(random_state=42),
        'params': {
            'model__n_estimators': [100, 200],
            'model__max_depth': [3, 5, 7]
        }
    },
    'GradientBoosting': {
        'model': GradientBoostingClassifier(random_state=42),
        'params': {
            'model__n_estimators': [100, 200],
            'model__learning_rate': [0.01, 0.1],
            'model__max_depth': [3, 5]
        }
    }
}


results = {}

for name, config in models.items():
    print(f"\n====Training {name}====")

    # Create pipeline
    pipeline = Pipeline(steps=[
        ('preprocessor', preprocessor),
        ('smote', smote),
        ('model', config['model'])
    ])

    # Randomized Search
    search = RandomizedSearchCV(
        pipeline,
        config['params'],
        n_iter = 10,
        cv=StratifiedKFold(5),
        scoring='roc_auc',
        n_jobs=-1,
        random_state=42
    )

    search.fit(x_train, y_train)

    # Evaluate
    y_pred = search.best_estimator_.predict(x_test)
    y_proba = search.best_estimator_.predict_proba(x_test)[:,1]

    # Store results
    results[name] = {
        'model': search.best_estimator_,
        'best_params': search.best_params_,
        'accuracy': accuracy_score(y_test, y_pred),
        'roc_auc': roc_auc_score(y_test, y_proba),
        'classification_report': classification_report(y_test, y_pred)
    }

    # Confusion matrix
    plt.figure()
    cm = confusion_matrix(y_test, y_pred)
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
    plt.title(f'{name} Confusion Matrix')
    plt.show()

    print(f"Best parameters: {search.best_params_}")
    print(classification_report(y_test, y_pred))
```

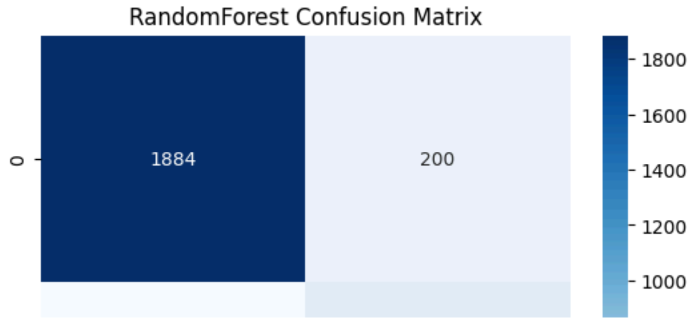
```
====Training LogisticRegression====
```

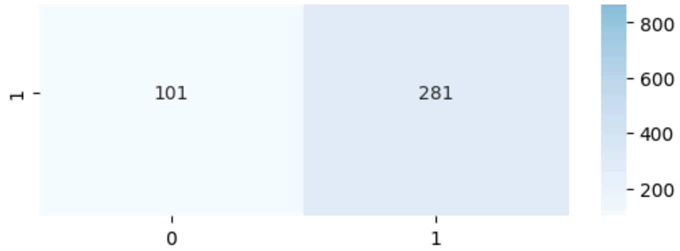


```
Best parameters: {'model__solver': 'liblinear', 'model__C': 0.1}
              precision    recall  f1-score   support

           0       0.96      0.86      0.91      2084
           1       0.52      0.81      0.63       382

    accuracy                           0.85      2466
   macro avg       0.74      0.83      0.77      2466
weighted avg       0.89      0.85      0.86      2466


====Training NeuralNetwork====
```

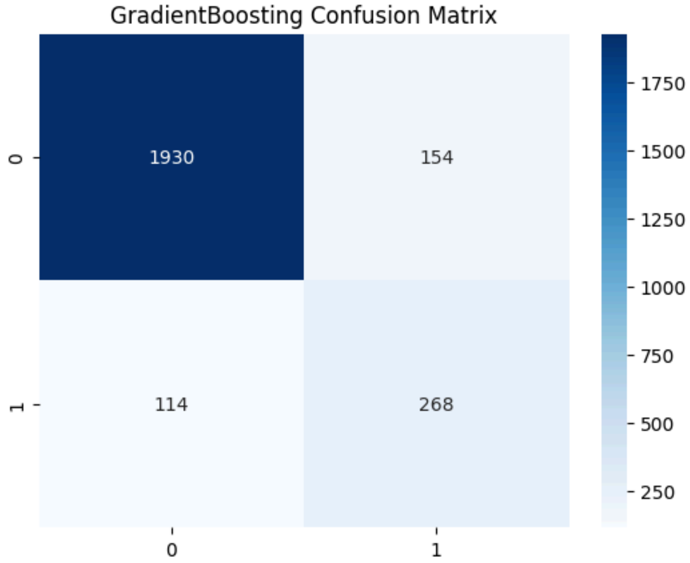


```
Best parameters: {'model__learning_rate_init': 0.001, 'model__hidden_layer_sizes': (100,), 'model__alpha': 0.0001}
              precision    recall  f1-score   support

           0       0.95      0.90      0.92      2084
           1       0.58      0.75      0.65       382

    accuracy                           0.88      2466
   macro avg       0.76      0.82      0.79      2466
weighted avg       0.89      0.88      0.88      2466


====Training RandomForest====
```



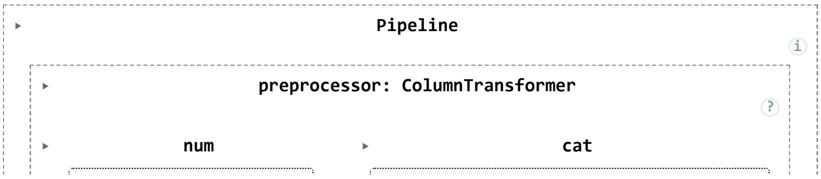


```
Best parameters: {'model__n_estimators': 200, 'model__max_depth': 7}
              precision    recall  f1-score   support

           0       0.95      0.90      0.93      2084
           1       0.58      0.74      0.65       382

    accuracy                           0.88      2466
   macro avg       0.77      0.82      0.79      2466
weighted avg       0.89      0.88      0.88      2466


====Training GradientBoosting====
```



```
Best parameters: {'model__n_estimators': 100, 'model__max_depth': 3, 'model__learning_rate': 0.1}
              precision    recall  f1-score   support

           0       0.94      0.93      0.94      2084
           1       0.64      0.70      0.67       382

    accuracy                           0.89      2466
   macro avg       0.79      0.81      0.80      2466
weighted avg       0.90      0.89      0.89      2466
```

```
best_model = results['GradientBoosting']['model']
best_model
```


Pipeline
preprocessor: ColumnTransformer
num
cat