```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, precision_score,
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.dummy import DummyClassifier
from sklearn.preprocessing import OneHotEncoder
from imblearn.pipeline import Pipeline as ImbPipeline
from imblearn.over_sampling import SMOTE
```

```
pip install ucimlrepo
```

```
Collecting ucimlrepo
  Downloading ucimlrepo-0.0.6-py3-none-any.whl (8.0 kB)
Installing collected packages: ucimlrepo
Successfully installed ucimlrepo-0.0.6
```

```
from ucimlrepo import fetch_ucirepo

# fetch dataset
online_shoppers_purchasing_intention_dataset = fetch_ucirepo(id=468)

# data (as pandas dataframes)
X = online_shoppers_purchasing_intention_dataset.data.features
y = online_shoppers_purchasing_intention_dataset.data.targets

# metadata
print(online_shoppers_purchasing_intention_dataset.metadata)

# variable information
print(online_shoppers_purchasing_intention_dataset.variables)
```

```
{'uci_id': 468, 'name': 'Online Shoppers Purchasing Intention Dataset', 're
                        name     role         type demographic description
0             Administrative  Feature      Integer        None        None
1    Administrative_Duration  Feature      Integer        None        None
2               Informational  Feature      Integer        None        None
3     Informational_Duration  Feature      Integer        None        None
4             ProductRelated  Feature      Integer        None        None
5    ProductRelated_Duration  Feature   Continuous        None        None
6                BounceRates  Feature   Continuous        None        None
7                  ExitRates  Feature   Continuous        None        None
8                 PageValues  Feature      Integer        None        None
9                 SpecialDay  Feature      Integer        None        None
10                     Month  Feature  Categorical        None        None
11         OperatingSystems  Feature      Integer        None        None
12                  Browser  Feature      Integer        None        None
13                   Region  Feature      Integer        None        None
14              TrafficType  Feature      Integer        None        None
15              VisitorType  Feature  Categorical        None        None
```

```
16                 Weekend  Feature       Binary        None        None
17                 Revenue   Target       Binary        None        None

   units missing_values
0   None             no
1   None             no
2   None             no
3   None             no
4   None             no
5   None             no
6   None             no
7   None             no
8   None             no
9   None             no
10  None             no
11  None             no
12  None             no
13  None             no
14  None             no
15  None             no
16  None             no
17  None             no
```

```
sns.set_theme()
```

```
from google.colab import drive
drive.mount("/content/drive")
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, ca
```

```
#Loading the dataset

import pandas as pd
df=pd.read_csv("/content/drive/MyDrive/online_shoppers_intention.csv")

df.head()
```

	Administrative	Administrative_Duration	Informational	Informational_Du
0	0	0.0	0	
1	0	0.0	0	
2	0	0.0	0	
3	0	0.0	0	
4	0	0.0	0	

Next steps: View recommended plots

```
#Handlng missing datas

missing = df.isnull().sum()
```

```
print(missing)
```

```
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
```

```
x = df.iloc[:, [5, 6]].values
x.shape
```

```
(12330, 2)
```

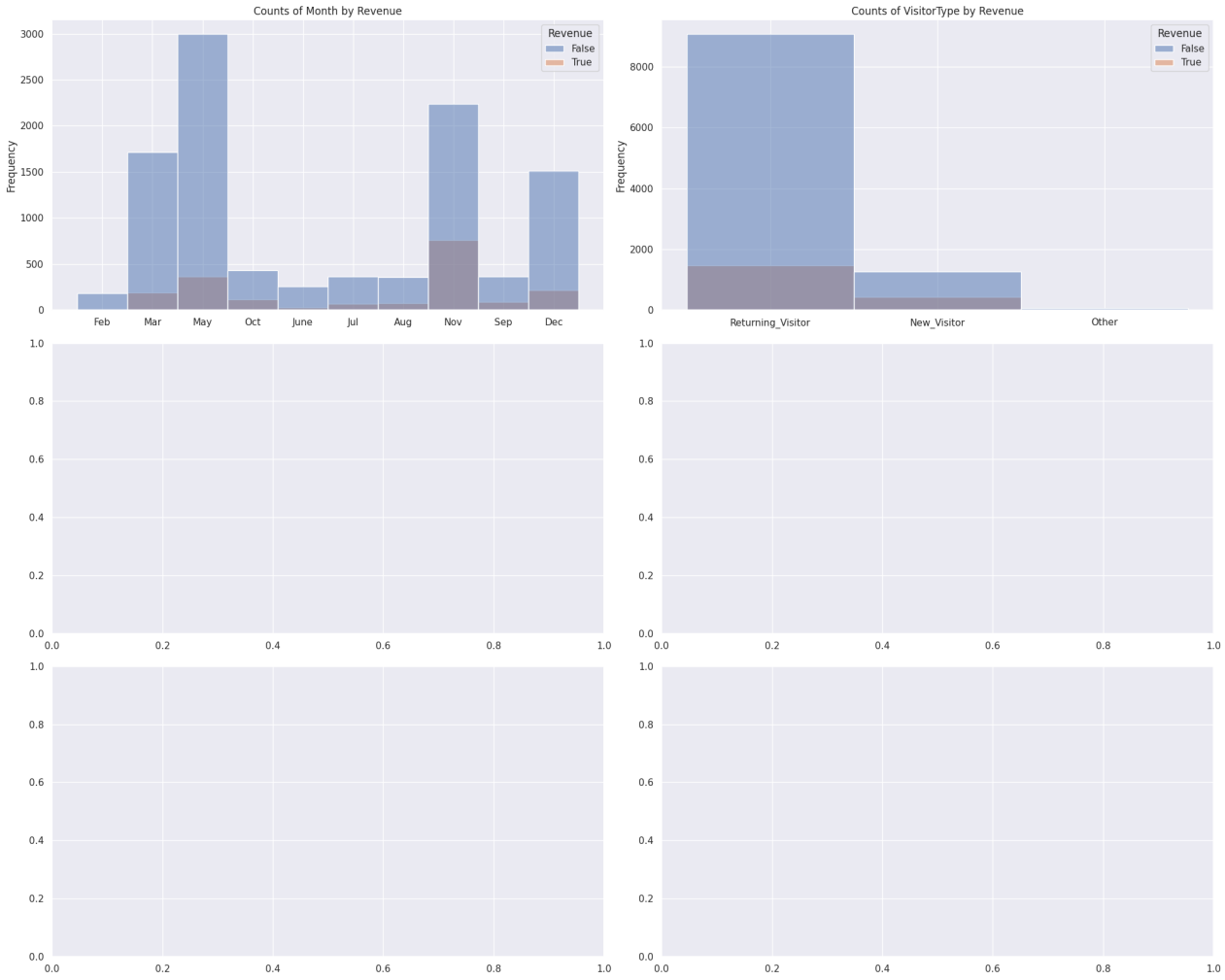
```
df.describe().transpose()
```

	count	mean	std	min	25%
Administrative	12330.0	2.315166	3.321784	0.0	0.000000
Administrative_Duration	12330.0	80.818611	176.779107	0.0	0.000000
Informational	12330.0	0.503569	1.270156	0.0	0.000000
Informational_Duration	12330.0	34.472398	140.749294	0.0	0.000000
ProductRelated	12330.0	31.731468	44.475503	0.0	7.000000
ProductRelated_Duration	12330.0	1194.746220	1913.669288	0.0	184.137500
BounceRates	12330.0	0.022191	0.048488	0.0	0.000000
ExitRates	12330.0	0.043073	0.048597	0.0	0.014286
PageValues	12330.0	5.889258	18.568437	0.0	0.000000
SpecialDay	12330.0	0.061427	0.198917	0.0	0.000000
OperatingSystems	12330.0	2.124006	0.911325	1.0	2.000000
Browser	12330.0	2.357097	1.717277	1.0	2.000000
Region	12330.0	3.147364	2.401591	1.0	1.000000
TrafficType	12330.0	4.069586	4.025169	1.0	2.000000

```
df.duplicated().sum(), df.shape
```

```
(125, (12330, 18))
```

```
#Categorical features

fig, axes = plt.subplots(3, 2, figsize=(20, 16))

# Plot histograms for numerical features based on the value of the target variab
for col, ax in zip(df.select_dtypes(include='object').columns, axes.flatten()):
    sns.histplot(data=df, x=df[col].values, hue="Revenue", ax=ax)
    ax.set_title(f'Counts of {col} by Revenue')
    ax.set_ylabel("Frequency")

plt.tight_layout()
plt.show()
```



```
#OneHot Encoding

encoder = OneHotEncoder(sparse_output=False)

for col in df.select_dtypes(include='object'):
    transformed = encoder.fit_transform(df[[col]])

    encoded_df = pd.DataFrame(transformed, columns=[f"{col}_{category}" for cat

    df = df.join(encoded_df.set_index(df.index)) # Add the new columns to the d


df.drop(columns=df.select_dtypes(include='object').columns, inplace=True)

df.columns
```

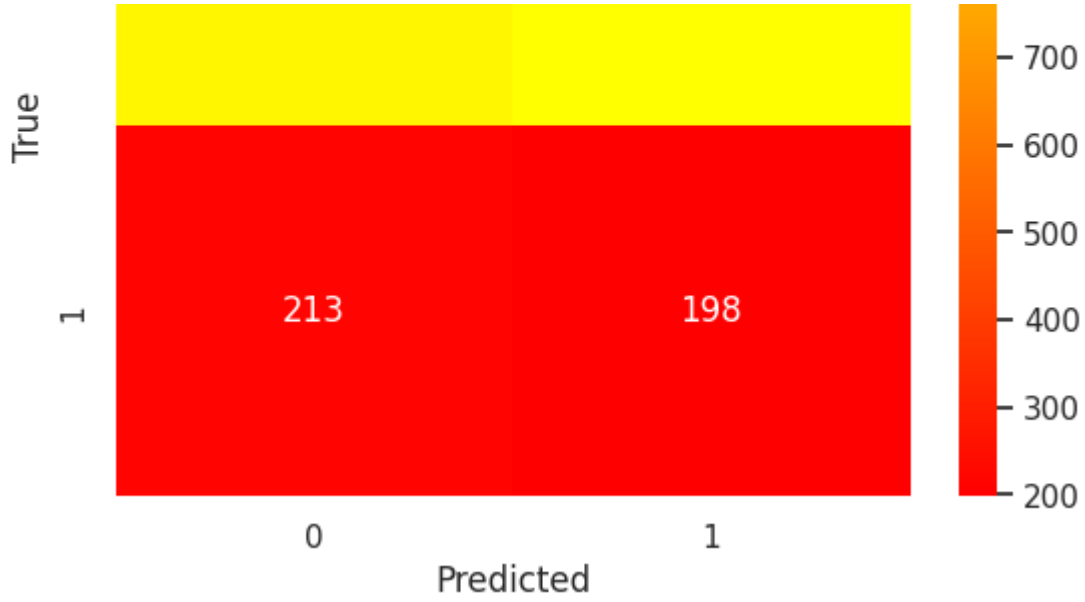
```
df.columns
```

```
Index(['Administrative', 'Administrative_Duration', 'Informational',
       'Informational_Duration', 'ProductRelated',
'ProductRelated_Duration',
       'BounceRates', 'ExitRates', 'PageValues', 'SpecialDay',
       'OperatingSystems', 'Browser', 'Region', 'TrafficType', 'Weekend',
       'Revenue', 'Month_Aug', 'Month_Dec', 'Month_Feb', 'Month_Jul',
       'Month_June', 'Month_Mar', 'Month_May', 'Month_Nov', 'Month_Oct',
       'Month_Sep', 'VisitorType_New_Visitor', 'VisitorType_Other',
       'VisitorType_Returning_Visitor'],
      dtype='object')
```

```
#Train-Test-Split

y = df['Revenue'] # Labels
X = df[df.columns.difference(['Revenue']).to_list()]



X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random
```

```
#Baseline using Random Classifier

majority_class_baseline = DummyClassifier(strategy='uniform')
majority_class_baseline.fit(X_train, y_train)
y_pred_majority = majority_class_baseline.predict(X_test)

print("Random Baseline:")
print(classification_report(y_test, y_pred_majority))

cm = confusion_matrix(y_test, y_pred_majority)
sns.heatmap(cm, annot=True, fmt="d", cmap="autumn")
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```

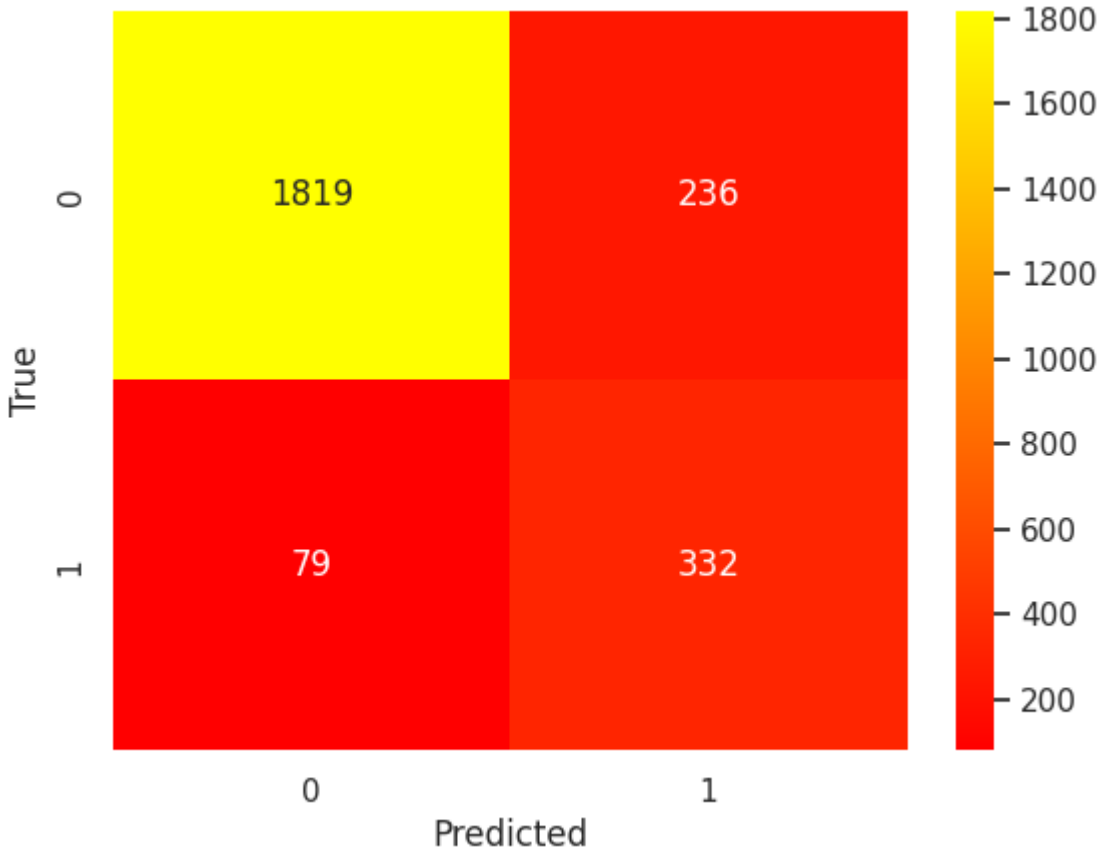
```
Random Baseline:
              precision    recall  f1-score   support

       False       0.83      0.49      0.62      2055
        True       0.16      0.48      0.24       411

    accuracy                           0.49      2466
   macro avg       0.49      0.49      0.43      2466
weighted avg       0.71      0.49      0.55      2466
```

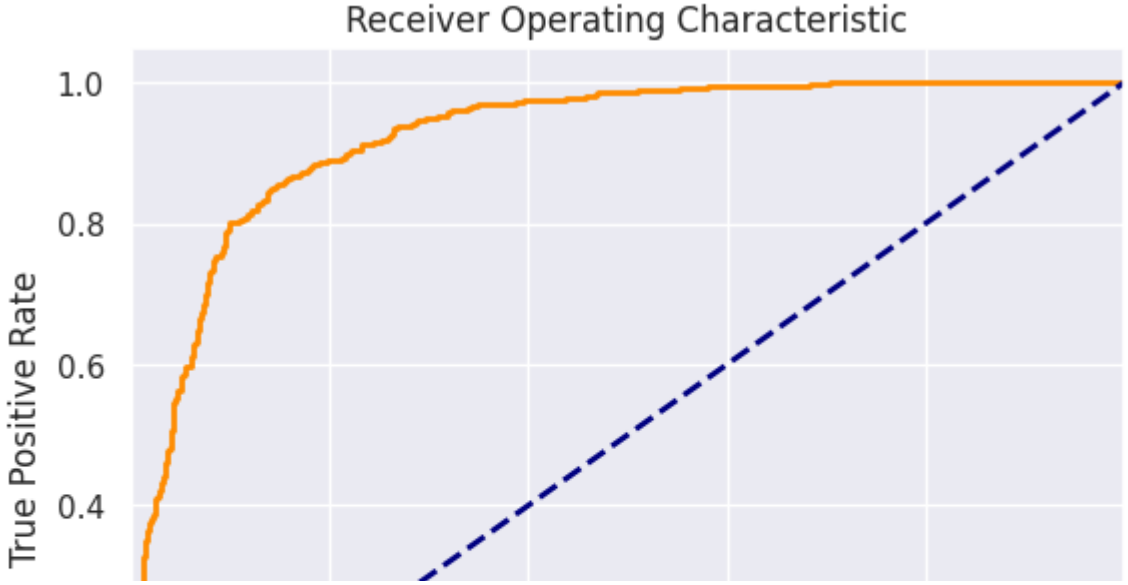


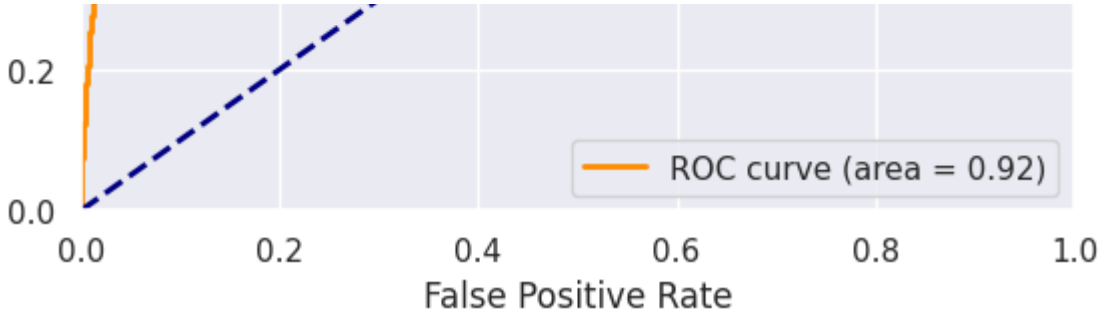

```
from sklearn.ensemble import RandomForestClassifier
from imblearn.pipeline import Pipeline as ImbPipeline
from imblearn.over_sampling import SMOTE

pipe = ImbPipeline([
    ('scaler', StandardScaler()),
    ('smote', SMOTE(random_state=42)),
    ('classifier', RandomForestClassifier(n_estimators=75, max_depth=10, min_sam
])

# Fit and predict
pipe.fit(X_train, y_train)
y_pred = pipe.predict(X_test)

# Predict probabilities for the test data
y_probs = pipe.predict_proba(X_test)[:, 1]  # get the probability of the positiv

# Evaluate the model
conf_matrix = confusion_matrix(y_test, y_pred)
print(classification_report(y_test, y_pred))

# Plot confusion matrix
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="autumn")
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()

# Calculate ROC Curve and AUC
fpr, tpr, thresholds = roc_curve(y_test, y_probs)
roc_auc = auc(fpr, tpr)

# Plot the ROC Curve
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' %
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
```

```
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic')
plt.legend(loc="lower right")
plt.show()
```

	precision	recall	f1-score	support
False	0.96	0.89	0.92	2055
True	0.58	0.81	0.68	411
accuracy			0.87	2466
macro avg	0.77	0.85	0.80	2466
weighted avg	0.90	0.87	0.88	2466








```
#Bivariate Analysis

sns.pairplot(df[df.select_dtypes(include=['number', 'bool']).columns.difference
```

```
<seaborn.axisgrid.PairGrid at 0x7ff109913eb0>
```