**Ex4 - k-Nearest Neighbor algorithm**

**GitHub Link:**

[GitHub Link](https://github.com/Ojus999/Machine-Learning-Sem-6/tree/main)

**Colab Links:**

[Link](https://colab.research.google.com/drive/1hAOsX5a4kraAL_r9A4UMHUBFo3ynYWHP?usp=sharing)

**Aim:**

Develop a python program to predict the Online Shoppers Purchasing Intention using K-Nearest Neighbour algorithm

**Import Dependencies**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from numpy import set\_printoptions

from sklearn.feature\_selection import RFE

from sklearn.linear\_model import LogisticRegression

from sklearn import preprocessing

from sklearn.preprocessing import StandardScaler

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score, roc\_auc\_score, roc\_curve, confusion\_matrix,f1\_score, precision\_score, recall\_score

from imblearn.over\_sampling import SMOTE

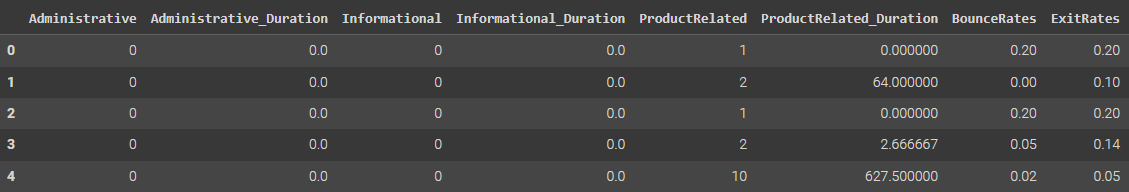
from imblearn.under\_sampling import RandomUnderSampler

**Read Data**

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

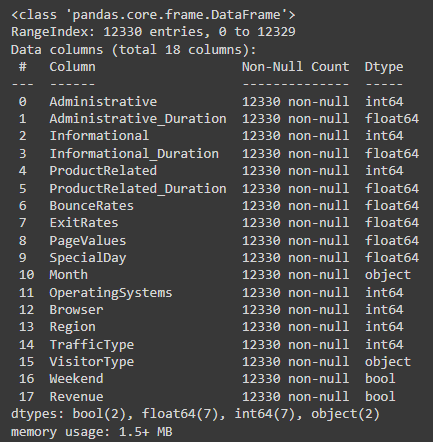
**Preprocessing:**

df.head()

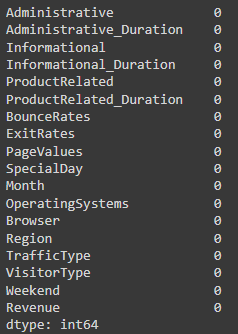


**Null Values**

df.info()



df.isnull().sum()



**Encoding**

label\_encoder = preprocessing.LabelEncoder()

df['Month'] = label\_encoder.fit\_transform(df['Month']);

df['VisitorType'] = label\_encoder.fit\_transform(df['VisitorType'])

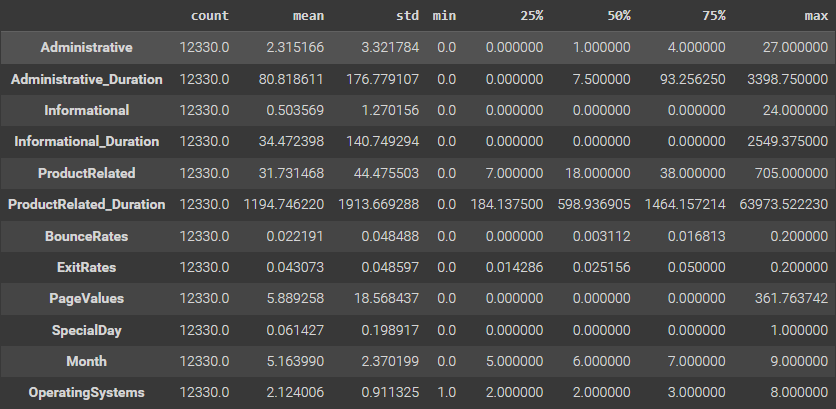
df['Weekend'] = label\_encoder.fit\_transform(df['Weekend'])

df['Revenue'] = label\_encoder.fit\_transform(df['Revenue'])

**Exploratory Data Analysis**

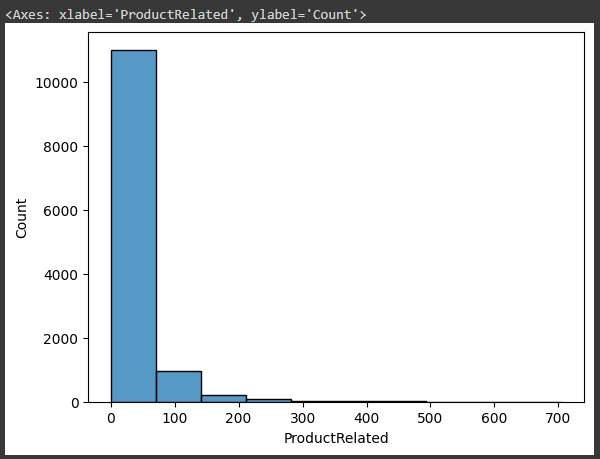
**Common Statistics**

df.describe().transpose()



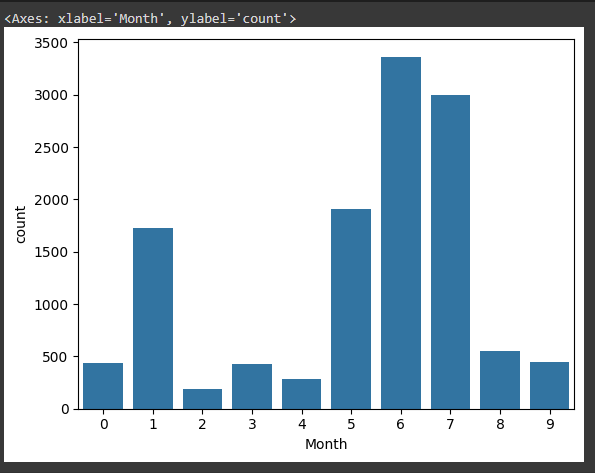
**Histograms**

sns.histplot(data = df, x = 'ProductRelated', bins = 10)

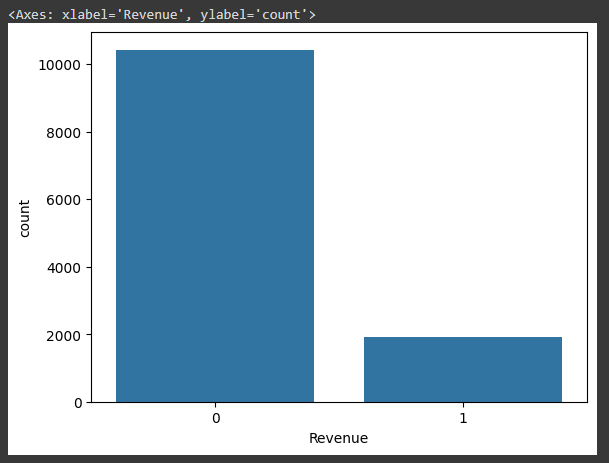


**Count Plot**

sns.countplot(data = df, x = 'Month')

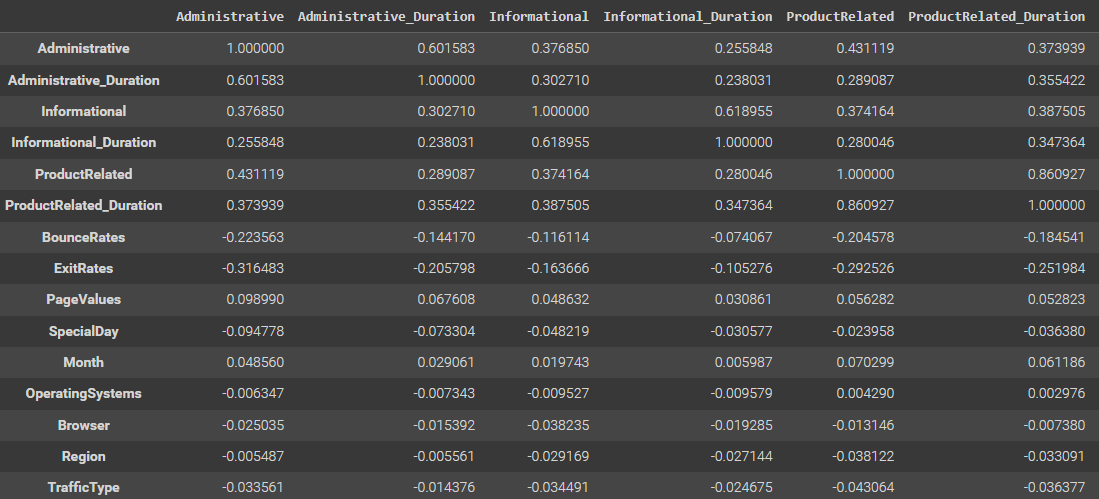


sns.countplot(data = df, x = 'Revenue')



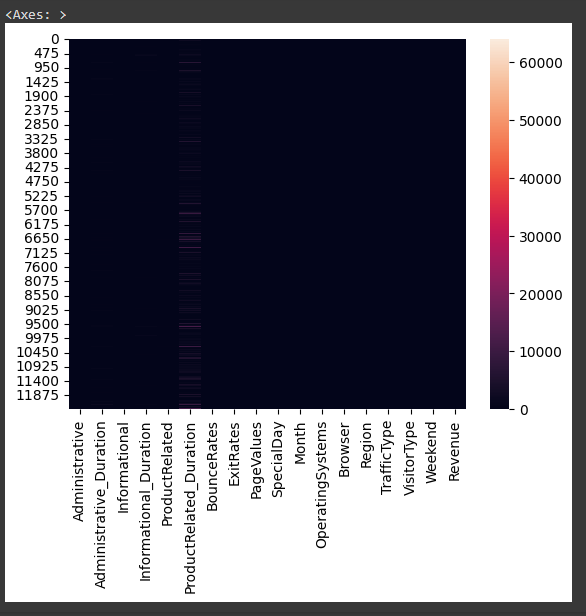
**Correlation**

df.corr()



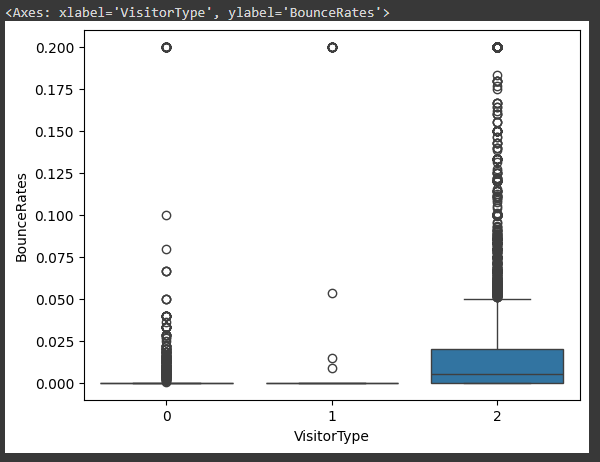
**Heatmap**

sns.heatmap(data = df)



**Box Plot**

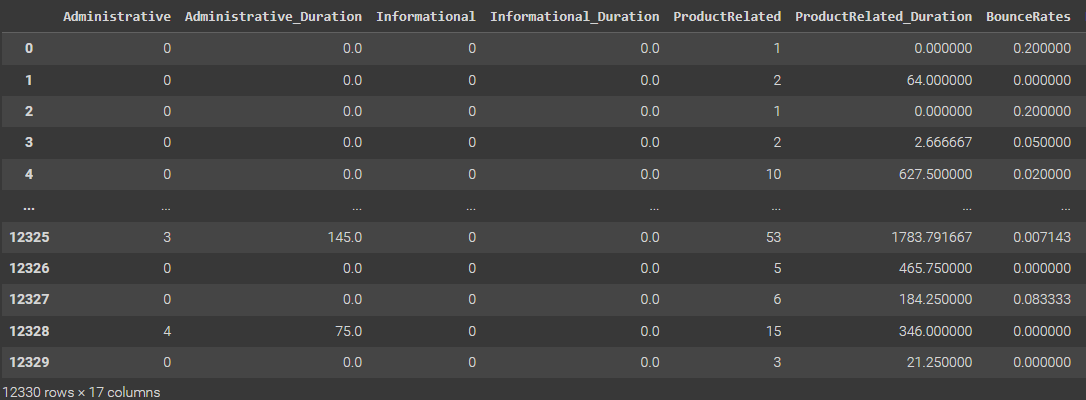
sns.boxplot(data = df, x = 'VisitorType', y = 'BounceRates')



**Feature Engineering**

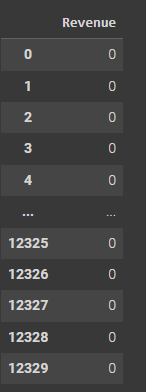
X = df.iloc[:,:17]

X



y = df.iloc[:,-1:]

y



**Split Data**

smote = SMOTE()

rus = RandomUnderSampler(random\_state=42, sampling\_strategy = 'majority')

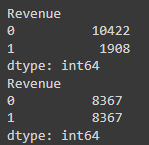
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, random\_state = 42, test\_size = 0.2)

**Sampling**

print((y.value\_counts()))

X\_resampled, y\_resampled = smote.fit\_resample(X\_train, y\_train)

print((y\_resampled.value\_counts()))



**Model Building**

**Normalization**

# data normalization with sklearn

from sklearn.preprocessing import MinMaxScaler

# fit scaler on training data

norm = MinMaxScaler().fit(X\_train)

# transform training data

X\_train\_norm = norm.transform(X\_train)

# transform testing dataabs

X\_test\_norm = norm.transform(X\_test)

# fit scaler on training data

norm = MinMaxScaler().fit(X\_train)

**Fit Model**

from sklearn.neighbors import KNeighborsClassifier

model = KNeighborsClassifier(n\_neighbors=11)

model2 = KNeighborsClassifier(n\_neighbors=11)

model.fit(X\_train, y\_train)

model2.fit(X\_resampled, y\_resampled)

y\_pred = model.predict(X\_test)

y\_pred2 = model2.predict(X\_test)

**Plotting Metrics**

import matplotlib.pyplot as plt

y\_pred\_proba = model.predict\_proba(X\_test)[::,1]

auc = roc\_auc\_score(y\_test, y\_pred\_proba)

fpr, tpr, \_ = roc\_curve(y\_test,  y\_pred\_proba)

plt.plot(fpr,tpr,label="with oversampling, auc="+str(auc), )

x = [0, 1]

y = [0, 1]

y\_pred\_proba2 = model2.predict\_proba(X\_test)[::,1]

auc2 = roc\_auc\_score(y\_test, y\_pred\_proba2)

fpr, tpr, \_ = roc\_curve(y\_test,  y\_pred\_proba2)

plt.plot(fpr,tpr,label="without oversampling, auc="+str(auc2), color='red')

print("Accuracy score without oversampling:",accuracy\_score(y\_test, y\_pred))

print("F1 score without oversampling:",f1\_score(y\_test, y\_pred))

print("Precision without oversampling:",precision\_score(y\_test, y\_pred))

print("Recall without oversampling:",recall\_score(y\_test, y\_pred))

print()

print("Accuracy score with oversampling:",accuracy\_score(y\_test, y\_pred2))

print("F1 score with oversampling:",f1\_score(y\_test, y\_pred2))

print("Precision with oversampling:",precision\_score(y\_test, y\_pred2))

print("Recall with oversampling:",recall\_score(y\_test, y\_pred2))

plt.plot(x,y)

plt.legend(loc=4)

plt.show()

