Ex. No: 5 Y.V.Ojus
21-3-24 3122 21 5001 125

Ex4 - k-Nearest Neighbor algorithm

GitHub Link:

GitHub Link

Colab Links:

Link

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")
```

Ex. No: 5 Y.V.Ojus
21-3-24 3122 21 5001 125

Preprocessing:

df.head()

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

Null Values

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.isnull().sum()

```
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
```

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'])
```

Ex. No: 5 Y.V.Ojus
21-3-24 3122 21 5001 125

Exploratory Data Analysis

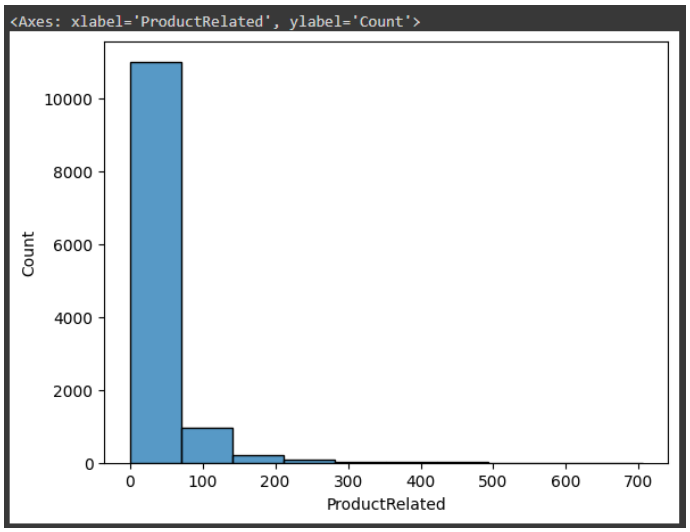
Common Statistics

df.describe().transpose()

	count	mean	std	min	25%	50%	75%	max
Administrative	12330.0	2.315166	3.321784	0.0	0.000000	1.000000	4.000000	27.000000
Administrative_Duration	12330.0	80.818611	176.779107	0.0	0.000000	7.500000	93.256250	3398.750000
Informational	12330.0	0.503569	1.270156	0.0	0.000000	0.000000	0.000000	24.000000
Informational_Duration	12330.0	34.472398	140.749294	0.0	0.000000	0.000000	0.000000	2549.375000
ProductRelated	12330.0	31.731468	44.475503	0.0	7.000000	18.000000	38.000000	705.000000
ProductRelated_Duration	12330.0	1194.746220	1913.669288	0.0	184.137500	598.936905	1464.157214	63973.522230
BounceRates	12330.0	0.022191	0.048488	0.0	0.000000	0.003112	0.016813	0.200000
ExitRates	12330.0	0.043073	0.048597	0.0	0.014286	0.025156	0.050000	0.200000
PageValues	12330.0	5.889258	18.568437	0.0	0.000000	0.000000	0.000000	361.763742
SpecialDay	12330.0	0.061427	0.198917	0.0	0.000000	0.000000	0.000000	1.000000
Month	12330.0	5.163990	2.370199	0.0	5.000000	6.000000	7.000000	9.000000
OperatingSystems	12330.0	2.124006	0.911325	1.0	2.000000	2.000000	3.000000	8.000000

Histograms

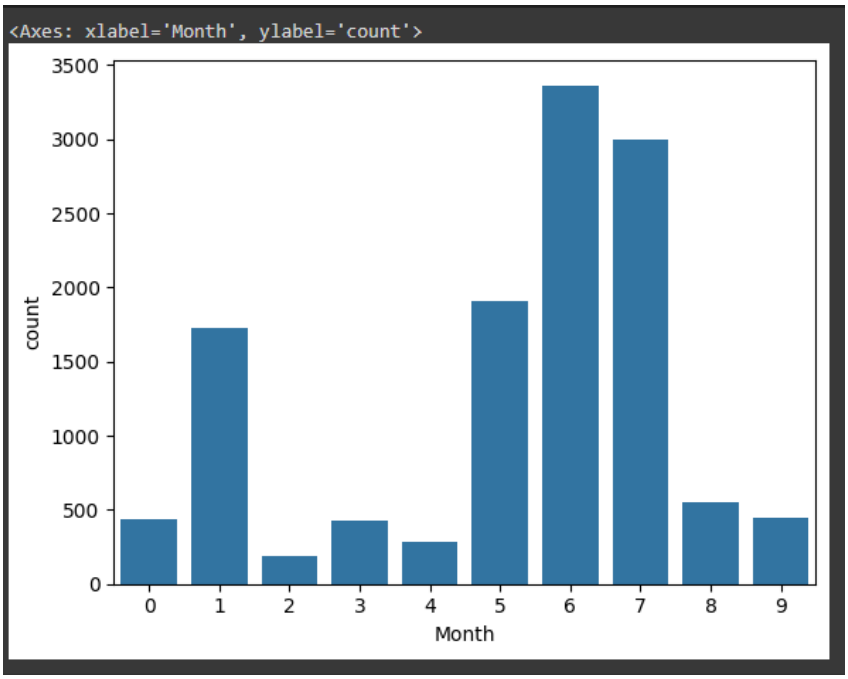
sns.histplot(data = df, x = 'ProductRelated', bins = 10)



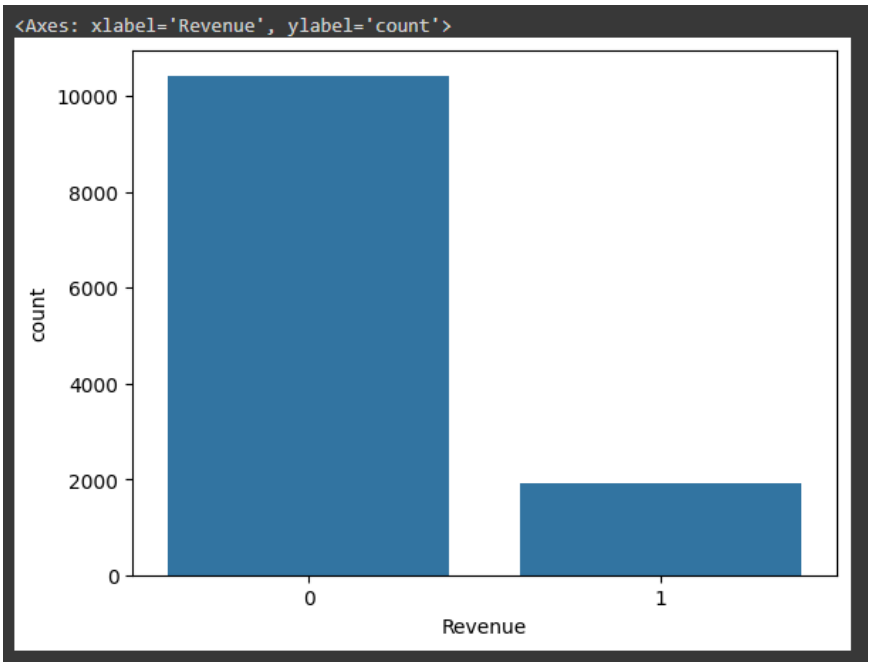
Ex. No: 5 Y.V.Ojus
21-3-24 3122 21 5001 125

Count Plot

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



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



Ex. No: 5 Y.V.Ojus
21-3-24 3122 21 5001 125

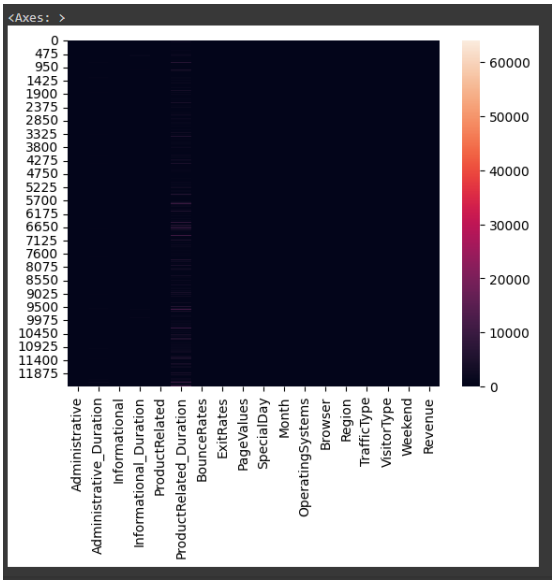
Correlation

df.corr()

	Administrative	Administrative_Duration	Informational	Informational_Duration	ProductRelated	ProductRelated_Duration
Administrative	1.000000	0.601583	0.376850	0.255848	0.431119	0.373939
Administrative_Duration	0.601583	1.000000	0.302710	0.238031	0.289087	0.355422
Informational	0.376850	0.302710	1.000000	0.618955	0.374164	0.387505
Informational_Duration	0.255848	0.238031	0.618955	1.000000	0.280046	0.347364
ProductRelated	0.431119	0.289087	0.374164	0.280046	1.000000	0.860927
ProductRelated_Duration	0.373939	0.355422	0.387505	0.347364	0.860927	1.000000
BounceRates	-0.223563	-0.144170	-0.116114	-0.074067	-0.204578	-0.184541
ExitRates	-0.316483	-0.205798	-0.163666	-0.105276	-0.292526	-0.251984
PageValues	0.098990	0.067608	0.048632	0.030861	0.056282	0.052823
SpecialDay	-0.094778	-0.073304	-0.048219	-0.030577	-0.023958	-0.036380
Month	0.048560	0.029061	0.019743	0.005987	0.070299	0.061186
OperatingSystems	-0.006347	-0.007343	-0.009527	-0.009579	0.004290	0.002976
Browser	-0.025035	-0.015392	-0.038235	-0.019285	-0.013146	-0.007380
Region	-0.005487	-0.005561	-0.029169	-0.027144	-0.038122	-0.033091
TrafficType	-0.033561	-0.014376	-0.034491	-0.024675	-0.043064	-0.036377

Heatmap

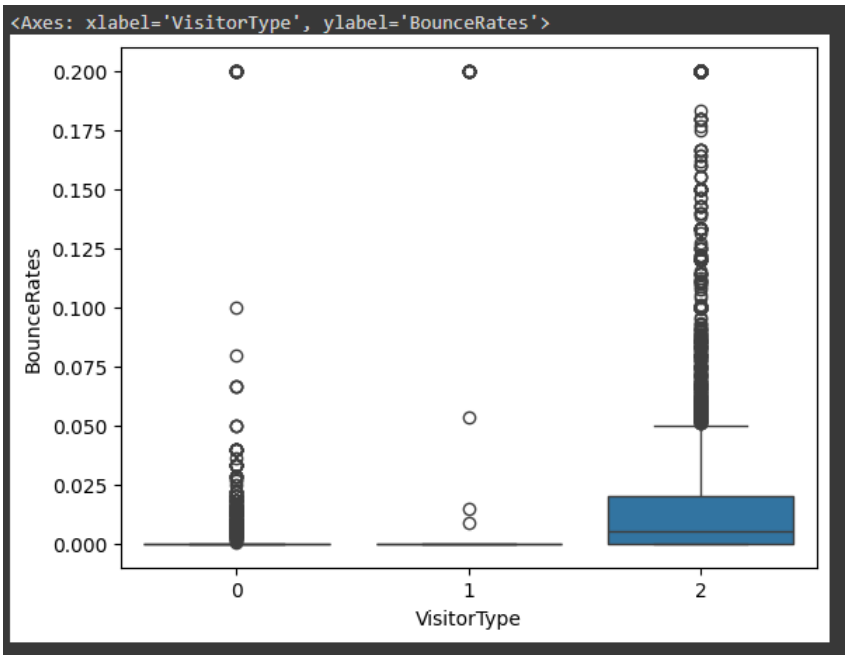
sns.heatmap(data = df)



Ex. No: 5 Y.V.Ojus
21-3-24 3122 21 5001 125

Box Plot

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



Feature Engineering

X = df.iloc[:,:17]

X

	Administrative	Administrative_Duration	Informational	Informational_Duration	ProductRelated	ProductRelated_Duration	BounceRates
0	0	0.0	0	0.0	1	0.000000	0.200000
1	0	0.0	0	0.0	2	64.000000	0.000000
2	0	0.0	0	0.0	1	0.000000	0.200000
3	0	0.0	0	0.0	2	2.666667	0.050000
4	0	0.0	0	0.0	10	627.500000	0.020000
...	...	...	...	...	...	...	...
12325	3	145.0	0	0.0	53	1783.791667	0.007143
12326	0	0.0	0	0.0	5	465.750000	0.000000
12327	0	0.0	0	0.0	6	184.250000	0.083333
12328	4	75.0	0	0.0	15	346.000000	0.000000
12329	0	0.0	0	0.0	3	21.250000	0.000000

12330 rows × 17 columns

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

	Revenue
0	0
1	0
2	0
3	0
4	0
...	...
12325	0
12326	0
12327	0
12328	0
12329	0

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()))
```

```
Revenue
0          10422
1           1908
dtype: int64
Revenue
0          8367
1          8367
dtype: int64
```

Ex. No: 5 Y.V.Ojus
21-3-24 3122 21 5001 125

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)
```

Ex. No: 5 Y.V.Ojus
21-3-24 3122 21 5001 125

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()
```

Ex. No: 5 Y.V.Ojus
21-3-24 3122 21 5001 125

```
Accuracy score without oversampling: 0.8552311435523114
F1 score without oversampling: 0.31477927063339733
Precision without oversampling: 0.7454545454545455
Recall without oversampling: 0.19951338199513383

Accuracy score with oversampling: 0.7591240875912408
F1 score with oversampling: 0.47340425531914887
Precision with oversampling: 0.3723849372384937
Recall with oversampling: 0.6496350364963503
```