LR_OSI_CV_SVG

September 1, 2020

[19]:
```
#Importing relevant libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
import warnings
warnings.filterwarnings("ignore")
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.model_selection import ShuffleSplit
```

[20]:
```
#Importing the OSI dataset:Online Shopper's Intention Data
OSI_df = pd.read_csv("C:/Users/delld/Downloads/online_shoppers_intention.csv")
```

[21]:
```
#checking for missing values
OSI_df.isna().values.any()
```

[21]:
```
False
```

[22]:
```
#Response variable counts
OSI_df.Revenue.value_counts()
```

[22]:
```
False    10422
True      1908
Name: Revenue, dtype: int64
```

[23]:
```
#Predictor Variables
X_features = list(OSI_df.columns)
X_features
```

[23]:
```
['Administrative',
 'Administrative_Duration',
 'Informational',
 'Informational_Duration',
 'ProductRelated',
 'ProductRelated_Duration',
 'BounceRates',
 'ExitRates',
```

```
 'PageValues',
 'SpecialDay',
 'Month',
 'OperatingSystems',
 'Browser',
 'Region',
 'TrafficType',
 'VisitorType',
 'Weekend',
 'Revenue']
```

[24]:
```
#Converting the categorical variables into dummy variables
encoded_OSI_df = pd.get_dummies(OSI_df[X_features])
encoded_OSI_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12330 entries, 0 to 12329
Data columns (total 29 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  OperatingSystems              12330 non-null  int64
 11  Browser                       12330 non-null  int64
 12  Region                        12330 non-null  int64
 13  TrafficType                   12330 non-null  int64
 14  Weekend                       12330 non-null  bool
 15  Revenue                       12330 non-null  bool
 16  Month_Aug                     12330 non-null  uint8
 17  Month_Dec                     12330 non-null  uint8
 18  Month_Feb                     12330 non-null  uint8
 19  Month_Jul                     12330 non-null  uint8
 20  Month_June                    12330 non-null  uint8
 21  Month_Mar                     12330 non-null  uint8
 22  Month_May                     12330 non-null  uint8
 23  Month_Nov                     12330 non-null  uint8
 24  Month_Oct                     12330 non-null  uint8
 25  Month_Sep                     12330 non-null  uint8
 26  VisitorType_New_Visitor       12330 non-null  uint8
```

```
 27  VisitorType_Other              12330 non-null  uint8
 28  VisitorType_Returning_Visitor  12330 non-null  uint8
dtypes: bool(2), float64(7), int64(7), uint8(13)
memory usage: 1.5 MB
```

[25]:
```
#Getting the response variable vector
Y = encoded_OSI_df.Revenue
X = encoded_OSI_df.drop(columns="Revenue")
X.info()
Y["False"]=0
Y["True"]=1
Y=Y.drop(labels='False')
Y=Y.drop(labels='True')
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12330 entries, 0 to 12329
Data columns (total 28 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  OperatingSystems               12330 non-null  int64
 11  Browser                        12330 non-null  int64
 12  Region                         12330 non-null  int64
 13  TrafficType                    12330 non-null  int64
 14  Weekend                        12330 non-null  bool
 15  Month_Aug                      12330 non-null  uint8
 16  Month_Dec                      12330 non-null  uint8
 17  Month_Feb                      12330 non-null  uint8
 18  Month_Jul                      12330 non-null  uint8
 19  Month_June                     12330 non-null  uint8
 20  Month_Mar                      12330 non-null  uint8
 21  Month_May                      12330 non-null  uint8
 22  Month_Nov                      12330 non-null  uint8
 23  Month_Oct                      12330 non-null  uint8
 24  Month_Sep                      12330 non-null  uint8
 25  VisitorType_New_Visitor        12330 non-null  uint8
 26  VisitorType_Other              12330 non-null  uint8
 27  VisitorType_Returning_Visitor  12330 non-null  uint8
dtypes: bool(1), float64(7), int64(7), uint8(13)
```

```
memory usage: 1.5 MB
```

[26]:
```
#Spliting the data into training and testing
train_X,test_X,train_y,test_y = train_test_split(X,Y,train_size=0.
↪8,random_state=42)
```

[27]:
```
#Developing Logistic Regression for classification
logit = LogisticRegression()
OSI_logit = logit.fit(train_X.astype(float),train_y)
OSI_logit.score(test_X,test_y)
```

[27]:
```
0.8682076236820763
```

[28]:
```
#Defining the k-fold cross validation strategy with K=3
kfold = KFold(n_splits=3)
scores = cross_val_score(logit,X,Y,cv=kfold)
scores
scores.mean()
```

[28]:
```
0.8806163828061638
```

[29]:
```
#Defining the k-fold cross validation strategy with K=5
kfold = KFold(n_splits=5)
scores = cross_val_score(logit,X,Y,cv=kfold)
scores
scores.mean()
```

[29]:
```
0.8815896188158963
```

[30]:
```
#Defining the k-fold cross validation strategy with shuffling and K=5
kfold = KFold(n_splits=5,shuffle=True,random_state=0)
scores = cross_val_score(logit,X,Y,cv=kfold)
scores
scores.mean()
```

[30]:
```
0.8811030008110301
```

[31]:
```
#Defining the k-fold cross validation strategy with shuffling split of 0.5 & 0.
↪5 and K=10
shuffle_split = ShuffleSplit(test_size=0.5,train_size=0.5,n_splits=10)
scores = cross_val_score(logit,X,Y,cv=shuffle_split)
scores.mean()
```

[31]:
```
0.8827250608272508
```

[32]:
```
#Defining the k-fold cross validation strategy with shuffling split of 0.5 & 0.
↪2 and K=10
```

```
shuffle_split = ShuffleSplit(test_size=0.5,train_size=0.2,n_splits=10)
scores = cross_val_score(logit,X,Y,cv=shuffle_split)
scores
scores.mean()
```

[32]: 0.8835523114355233

[34]:
```
#Defining the Leave One Out (LOOCV) cross validation strategy on the test␣
↪dataset for ILLLUSTRATION PURPOSE ONLY
from sklearn.model_selection import LeaveOneOut
loo = LeaveOneOut()
scores = cross_val_score(logit,test_X,test_y,cv=loo)
len(scores)
scores.mean()
```

[34]: 0.8665855636658556

[ ]: