In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [4]:

```
df = pd.read_csv('../Machine Learning Project/online_shoppers_intention.csv')
```

In [5]:

```
df
```

Out[5]:

	Administrative	Administrative_Duration	Informational	Informational_Duration	ProductRela
0	0	0.0	0	0.0	
1	0	0.0	0	0.0	
2	0	0.0	0	0.0	
3	0	0.0	0	0.0	
4	0	0.0	0	0.0	
...	...	...	...	...	
12325	3	145.0	0	0.0	
12326	0	0.0	0	0.0	
12327	0	0.0	0	0.0	
12328	4	75.0	0	0.0	
12329	0	0.0	0	0.0	

12330 rows × 18 columns

In [6]:

```
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
```

In [7]:

```
df.describe()
```

Out[7]:

	Administrative	Administrative_Duration	Informational	Informational_Duration	ProductRela
count	12330.000000	12330.000000	12330.000000	12330.000000	12330.000(
mean	2.315166	80.818611	0.503569	34.472398	31.731
std	3.321784	176.779107	1.270156	140.749294	44.475
min	0.000000	0.000000	0.000000	0.000000	0.000(
25%	0.000000	0.000000	0.000000	0.000000	7.000(
50%	1.000000	7.500000	0.000000	0.000000	18.000(
75%	4.000000	93.256250	0.000000	0.000000	38.000(
max	27.000000	3398.750000	24.000000	2549.375000	705.000(

In [8]:

```
df.isnull().sum() #no missing value
```

Out[8]:

```
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
```

In [9]:

```
df['Revenue'] = df['Revenue'].astype(int) #clean data type: bool to int
```

In [10]:

```
df['Weekend'] = df['Weekend'].astype(int) #clean data type: bool to int
```

In [11]:

```
month = {'Feb':2, 'Mar':3,'May':5, 'June':6, 'Jul':7, 'Aug':8, 'Sep':9, 'Oct':10, 'N
df['Month'] = df['Month'].map(month) #clean data type: str to int
```

In [12]:

```
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  int64  
 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  int64  
 17  Revenue                  12330 non-null  int64  
dtypes: float64(7), int64(10), object(1)
memory usage: 1.7+ MB
```

Encoding

In [13]:

```
df['VisitorType'] = df['VisitorType'].map({'Returning_Visitor':2, 'New_Visitor':1, '
```

In [14]:

```
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  int64  
 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  int64  
 16  Weekend                  12330 non-null  int64  
 17  Revenue                  12330 non-null  int64  
dtypes: float64(7), int64(11)
memory usage: 1.7 MB
```

In [88]:

```
plt.figure(figsize=(15,10))
sns.heatmap(df.corr().round(2),
            linewidths=0.1,
            vmax=1.0,
            square=True,
            linecolor='white',
            cmap="YlGnBu",
            annot=True)
```

Out[88]:

```
<AxesSubplot:>
```

In [16]:

```
a = df.corr()
b = a[['Revenue']]
b.sort_values(by='Revenue', ascending=False)
```

Out[16]:

	Revenue
Revenue	1.000000
PageValues	0.492569
ProductRelated	0.158538
ProductRelated_Duration	0.152373
Administrative	0.138917
Month	0.127372
Informational	0.095200
Administrative_Duration	0.093587
Informational_Duration	0.070345
Weekend	0.029295
Browser	0.023984
TrafficType	-0.005113
Region	-0.011595
OperatingSystems	-0.014668
SpecialDay	-0.082305
VisitorType	-0.098485
BounceRates	-0.150673
ExitRates	-0.207071

Outliners

In [17]:

```
print('1º Quartile: ', df['ProductRelated_Duration'].quantile(q = 0.25))
print('2º Quartile: ', df['ProductRelated_Duration'].quantile(q = 0.50))
print('3º Quartile: ', df['ProductRelated_Duration'].quantile(q = 0.75))
print('4º Quartile: ', df['ProductRelated_Duration'].quantile(q = 1.00))
#Calculate the outliers:
  # Interquartile range, IQR = Q3 - Q1
  # lower 1.5*IQR whisker = Q1 - 1.5 * IQR
  # Upper 1.5*IQR whisker = Q3 + 1.5 * IQR

print('Duration above: ', df['ProductRelated_Duration'].quantile(q = 0.75) +
                      1.5*(df['ProductRelated_Duration'].quantile(q = 0.75) - df['Pr
```

```
1º Quartile:  184.1375
2º Quartile:  598.9369047499999
3º Quartile:  1464.1572135000001
4º Quartile:  63973.52223
Duration above:  3384.1867837500004 are outliers
```

In [18]:

```
print('1º Quartile: ', df['Administrative_Duration'].quantile(q = 0.25))
print('2º Quartile: ', df['Administrative_Duration'].quantile(q = 0.50))
print('3º Quartile: ', df['Administrative_Duration'].quantile(q = 0.75))
print('4º Quartile: ', df['Administrative_Duration'].quantile(q = 1.00))
#Calculate the outliers:
  # Interquartile range, IQR = Q3 - Q1
  # lower 1.5*IQR whisker = Q1 - 1.5 * IQR
  # Upper 1.5*IQR whisker = Q3 + 1.5 * IQR

print('Duration above: ', df['Administrative_Duration'].quantile(q = 0.75) +
                      1.5*(df['Administrative_Duration'].quantile(q = 0.75) - df['Ad
```

```
1º Quartile:  0.0
2º Quartile:  7.5
3º Quartile:  93.25625
4º Quartile:  3398.75
Duration above:  233.14062499999997 are outliers
```

In [19]:

```
print('1º Quartile: ', df['Informational_Duration'].quantile(q = 0.25))
print('2º Quartile: ', df['Informational_Duration'].quantile(q = 0.50))
print('3º Quartile: ', df['Informational_Duration'].quantile(q = 0.75))
print('4º Quartile: ', df['Informational_Duration'].quantile(q = 1.00))
#Calculate the outliers:
  # Interquartile range, IQR = Q3 - Q1
  # lower 1.5*IQR whisker = Q1 - 1.5 * IQR
  # Upper 1.5*IQR whisker = Q3 + 1.5 * IQR

print('Duration above: ', df['Informational_Duration'].quantile(q = 0.75) +
                      1.5*(df['Informational_Duration'].quantile(q = 0.75) - df['Inf
```

```
1º Quartile:  0.0
2º Quartile:  0.0
3º Quartile:  0.0
4º Quartile:  2549.375
Duration above:  0.0 are outliers
```

In [20]:

```
df = df[df.ProductRelated_Duration < 3384.18]
df = df[df.Administrative_Duration < 233.14]
```

In [21]:

```
df.describe()
```

Out[21]:

	Administrative	Administrative_Duration	Informational	Informational_Duration	ProductRela
count	10467.000000	10467.000000	10467.000000	10467.000000	10467.000(
mean	1.604662	36.259147	0.337346	20.959281	21.961
std	2.442930	56.192433	0.976302	102.611044	22.490
min	0.000000	0.000000	0.000000	0.000000	0.000(
25%	0.000000	0.000000	0.000000	0.000000	6.000(
50%	0.000000	0.000000	0.000000	0.000000	15.000(
75%	3.000000	58.033333	0.000000	0.000000	30.000(
max	19.000000	233.083333	16.000000	2252.033333	223.000(

In [22]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10467 entries, 0 to 12329
Data columns (total 18 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   Administrative           10467 non-null  int64  
 1   Administrative_Duration  10467 non-null  float64
 2   Informational            10467 non-null  int64  
 3   Informational_Duration   10467 non-null  float64
 4   ProductRelated           10467 non-null  int64  
 5   ProductRelated_Duration  10467 non-null  float64
 6   BounceRates              10467 non-null  float64
 7   ExitRates                10467 non-null  float64
 8   PageValues               10467 non-null  float64
 9   SpecialDay               10467 non-null  float64
 10  Month                    10467 non-null  int64  
 11  OperatingSystems         10467 non-null  int64  
 12  Browser                  10467 non-null  int64  
 13  Region                   10467 non-null  int64  
 14  TrafficType              10467 non-null  int64  
 15  VisitorType              10467 non-null  int64  
 16  Weekend                  10467 non-null  int64  
 17  Revenue                  10467 non-null  int64  
dtypes: float64(7), int64(11)
memory usage: 1.5 MB
```

train_test_split

In [23]:

```
X = df.drop(columns='Revenue', axis=1)
y = df['Revenue']
```

In [24]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,train_size=0.8,random_state
print("Input Training:",X_train.shape)
print("Input Test:",X_test.shape)
print("Output Training:",y_train.shape)
print("Output Test:",y_test.shape)
```

```
Input Training: (8373, 17)
Input Test: (2094, 17)
Output Training: (8373,)
Output Test: (2094,)
```

feature importance

In [25]:

```
import xgboost
fi = xgboost.XGBClassifier()
fi.fit(X,y)
```

```
[16:13:06] WARNING: /Users/runner/miniforge3/conda-bld/xgboost-split_1
643227205751/work/src/learner.cc:1115: Starting in XGBoost 1.3.0, the
default evaluation metric used with the objective 'binary:logistic' wa
s changed from 'error' to 'logloss'. Explicitly set eval_metric if yo
u'd like to restore the old behavior.


/Users/ingrid/opt/anaconda3/lib/python3.9/site-packages/xgboost/sklear
n.py:1224: UserWarning: The use of label encoder in XGBClassifier is d
eprecated and will be removed in a future release. To remove this warn
ing, do the following: 1) Pass option use_label_encoder=False when con
structing XGBClassifier object; and 2) Encode your labels (y) as integ
ers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
```

Out[25]:

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, enable_categoric
al=False,
              gamma=0, gpu_id=-1, importance_type=None,
              interaction_constraints='', learning_rate=0.300000012,
              max_delta_step=0, max_depth=6, min_child_weight=1, missi
ng=nan,
              monotone_constraints='()', n_estimators=100, n_jobs=8,
              num_parallel_tree=1, predictor='auto', random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample
=1,
              tree_method='exact', validate_parameters=1, verbosity=No
ne)
```

In [26]:

```
data = list(zip(X.columns, fi.feature_importances_))
feature_importance = pd.DataFrame(data, columns =['Feature', 'Importance'])
```

In [27]:

```
pd.set_option('display.max_rows', 20)
feature_importance.sort_values(by='Importance', ascending=False)
```

Out[27]:

	Feature	Importance
8	PageValues	0.354704
10	Month	0.086715
15	VisitorType	0.063573
6	BounceRates	0.060466
0	Administrative	0.049524
7	ExitRates	0.040472
2	Informational	0.035623
4	ProductRelated	0.035499
1	Administrative_Duration	0.035209
5	ProductRelated_Duration	0.034034
16	Weekend	0.031666
14	TrafficType	0.030828
9	SpecialDay	0.030021
3	Informational_Duration	0.029965
12	Browser	0.028707
13	Region	0.026991
11	OperatingSystems	0.026004

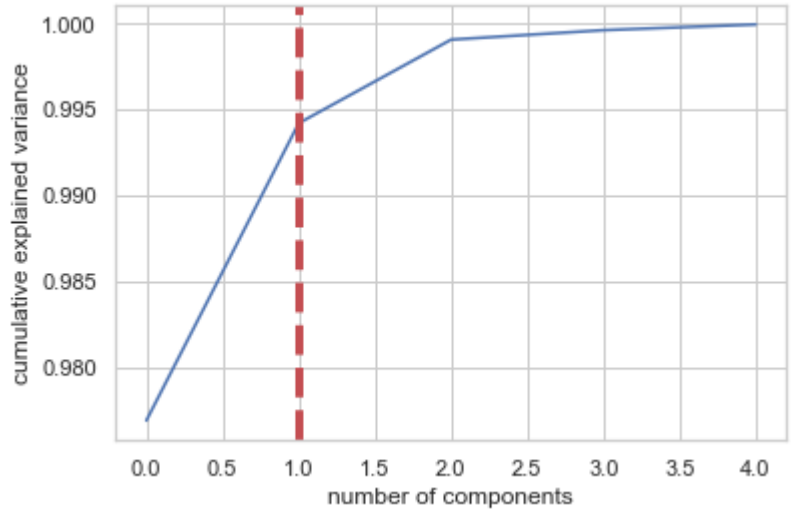
PCA

In [28]:

```
from sklearn.decomposition import PCA
# pca = PCA(n_components = 4)
# X_train_pca = pca.fit_transform(X_train)
# X_test_pca = pca.transform(X_test)
```

In [29]:

```
pca_test = PCA(n_components=5)
pca_test.fit(X_train)
sns.set(style='whitegrid')
plt.plot(np.cumsum(pca_test.explained_variance_ratio_))
plt.xlabel('number of components')
plt.ylabel('cumulative explained variance')
plt.axvline(linewidth=4, color='r', linestyle = '--', x=1, ymin=0, ymax=1)
display(plt.show())
evr = pca_test.explained_variance_ratio_
cvr = np.cumsum(pca_test.explained_variance_ratio_)
```



None

In [30]:

```
pca_test.explained_variance_ratio_
```

Out[30]:

```
array([9.76892057e-01, 1.72962520e-02, 4.86249650e-03, 5.49929960e-04,
       3.32771367e-04])
```

In [31]:

```
pca_df = pd.DataFrame()
pca_df['Cumulative Variance Ratio'] = cvr
pca_df['Explained Variance Ratio'] = evr
display(pca_df.head(10))
```

	Cumulative Variance Ratio	Explained Variance Ratio
0	0.976892	0.976892
1	0.994188	0.017296
2	0.999051	0.004862
3	0.999601	0.000550
4	0.999934	0.000333

In [32]:

```
explained_variance = pca_test.explained_variance_ratio_
```

In [33]:

```
explained_variance
```

Out[33]:

```
array([9.76892057e-01, 1.72962520e-02, 4.86249650e-03, 5.49929960e-04,
       3.32771367e-04])
```