In [ ]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder, StandardScaler,MinMaxScaler
from sklearn.impute import SimpleImputer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.cluster import KMeans
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import mean_squared_error, r2_score,confusion_matrix,classification_re

file_path = 'LAbTest - LAbTest.csv'

# Loading the data
data = pd.read_csv(file_path)

# Display the first few rows of the dataset
data.head()
```

Out[ ]:

	Administrative	Administrative_Duration	Informational	Informational_Duration	ProductRelated
0	0	0.0	0	0.0	1
1	0	0.0	0	0.0	2
2	0	0.0	0	0.0	1
3	0	0.0	0	0.0	2
4	0	0.0	0	0.0	10

In [ ]:

```
# Exploratory analysis

# Display basic information about the dataset
data.info()

# Display statistical summary of the dataset
data.describe()
```

```
<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
```

Out[ ]:

	Administrative	Administrative_Duration	Informational	Informational_Duration	ProductRelat
count	12330.000000	12330.000000	12330.000000	12330.000000	12330.0000
mean	2.315166	80.818611	0.503569	34.472398	31.7314
std	3.321784	176.779107	1.270156	140.749294	44.4755
min	0.000000	0.000000	0.000000	0.000000	0.0000
25%	0.000000	0.000000	0.000000	0.000000	7.0000
50%	1.000000	7.500000	0.000000	0.000000	18.0000
75%	4.000000	93.256250	0.000000	0.000000	38.0000
max	27.000000	3398.750000	24.000000	2549.375000	705.0000

In [ ]:

```
# Check for missing values
missing_values = data.isnull().sum()
print(missing_values[missing_values > 0])

# Fill missing values with the mean for numerical columns and mode for categorical columns
for column in data.columns:
    if data[column].dtype == 'object':
        data[column].fillna(data[column].mode()[0], inplace=True)
    else:
        data[column].fillna(data[column].mean(), inplace=True)
```

```
Series([], dtype: int64)
```

In [ ]:
```
# convert categorical values to numberical using onehotencoding using pandas getdummies
# Encode categorical variables using one-hot encoding
data = pd.get_dummies(data, drop_first=True)
```

In [ ]:
```
# Preprocess the dataset using MinMaxScalar Normalization
from sklearn.preprocessing import MinMaxScaler
minmax = MinMaxScaler()
minmax.fit_transform(data)
data.head()
```

Out[ ]:

	Administrative	Administrative_Duration	Informational	Informational_Duration	ProductRelated
0	0	0.0	0	0.0	1
1	0	0.0	0	0.0	2
2	0	0.0	0	0.0	1
3	0	0.0	0	0.0	2
4	0	0.0	0	0.0	10

5 rows × 27 columns

In [ ]:
```
# split the dataset into train and split data
# The dataset consists of 18 attributes. The Revenue attribute can be used as the class lab
# Administrative, Administrative Duration, Informational, Informational Duration, Product
# Related and Product Related Duration represent the number of different types of pages vis
# by the visitor in that session and total time spent in each of these page categories.
from sklearn.model_selection import train_test_split
X = data[['Administrative','Informational','ProductRelated','ProductRelated_Duration']]
Y = data['Revenue']
x_train,x_test,y_train,y_test = train_test_split(X,Y,test_size=0.2,random_state=42)
```

In [ ]:
```
# Perform Classification using KNN classifier for atleast 4 diffrent values of K

k_values = [2,3,4,5]

for i in k_values:
    modelknn = KNeighborsClassifier(n_neighbors=i,metric="euclidean")
    modelknn.fit(x_train,y_train)

# Predictions for KNN
y_train_pred_knn = modelknn.predict(x_train)
y_test_pred_knn = modelknn.predict(x_test)
```

In [ ]:
```
# Perform Classification using NAiveBayes classifier using all possible parameters

# Build Naive Bayes classifier
nb = GaussianNB()
nb.fit(x_train, y_train)
```

```
# Predictions for Naive Bayes
y_train_pred_nb = nb.predict(x_train)
y_test_pred_nb = nb.predict(x_test)
```

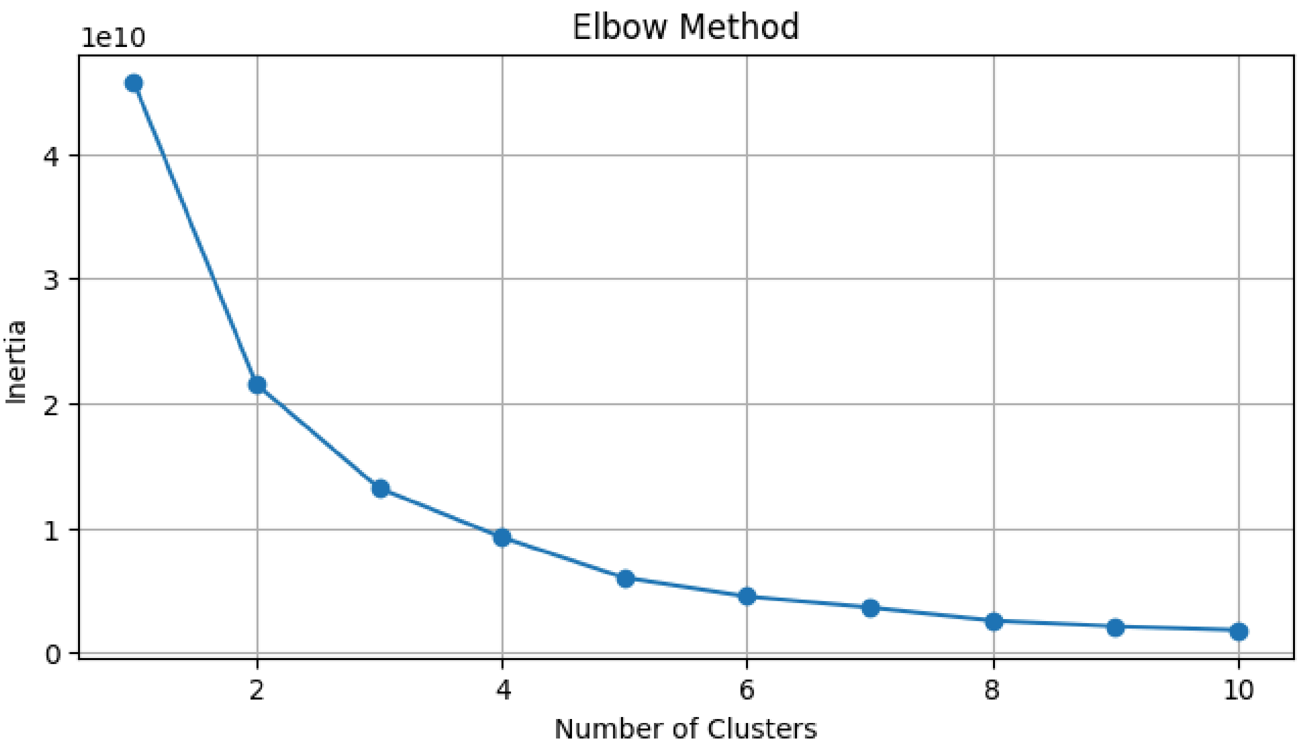
In [ ]:
```
# Cluster the visitors into 8 clusters ,visalize it and compute the silhoutte score score w
inertias = []
# Performing Elbow Method to find the Best number of clusters
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, random_state=42)
    kmeans.fit(data)
    inertias.append(kmeans.inertia_)

# plotting the possible k values
plt.figure(figsize=(8, 4))
plt.plot(range(1, 11), inertias, marker='o')
plt.title('Elbow Method')
plt.xlabel('Number of Clusters')
plt.ylabel('Inertia')
plt.grid(True)
plt.show()


kmeans = KMeans(n_clusters=4, random_state=42)
kmeans.fit(data)

labels = kmeans.labels_

data['Cluster'] = labels
```



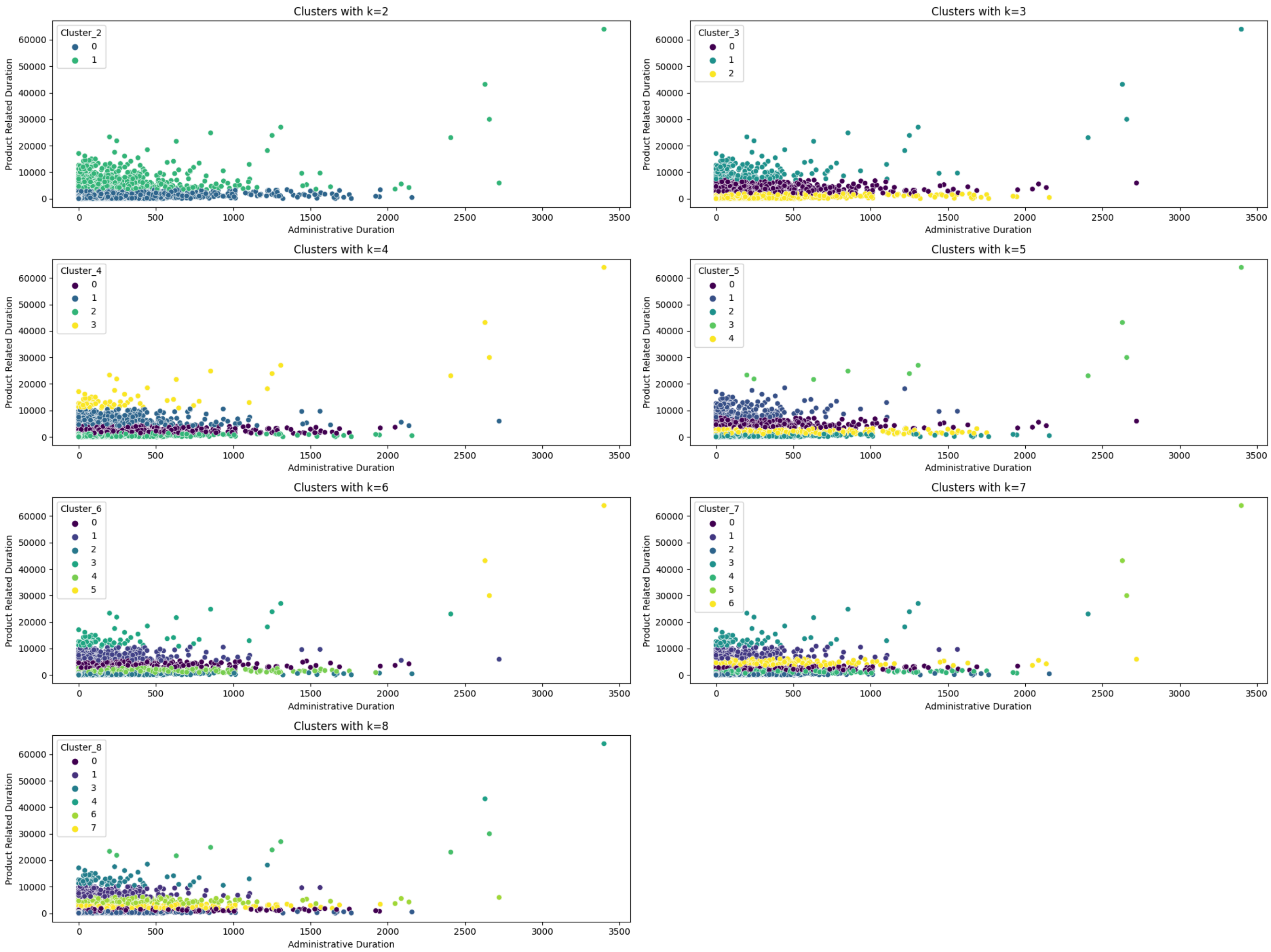
By interpreting the Elbow graph we could understand that the eblow is formed at 2 but it would cause many errors at that datapoint hence we are choosing 3 as the value for the numbers

In [ ]:
```
# providing all possible k values
kcluster_values = [2, 3, 4, 5, 6, 7, 8]
plt.figure(figsize=(20, 15))

# plotting the clusters for the  7 values of k
for idx, k in enumerate(kcluster_values):
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(data)
    labels = kmeans.labels_
    data[f'Cluster_{k}'] = labels

    plt.subplot(4, 2, idx + 1)
    sns.scatterplot(data=data, x='Administrative_Duration', y='ProductRelated_Duration', hu
    plt.title(f'Clusters with k={k}')
    plt.xlabel('Administrative Duration')
    plt.ylabel('Product Related Duration')

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



In [ ]:
```
# According to the question we are taking into account for validating wheather 8 clusters c
# We are taking into account for the following metrics
#  Silhouette score
# Calculating silhouettes score for K= 3 clustering
kmeans = KMeans(n_clusters=k, random_state=42)
kmeans.fit(data)
```

```
cluster_3_score = silhouette_score(X,data['Cluster_3'])
print("Silhouettes Score for K = 3",cluster_3_score)

# calculating silhouettes score for K= 8 clustering
kmeans = KMeans(n_clusters=k, random_state=42)
kmeans.fit(data)
cluster_3_score = silhouette_score(X,data['Cluster_8'])
print("Silhouettes Score for K = 8",cluster_3_score)
```

```
0.7108026008602492
0.6132905480245872
```

In [ ]:

```
# Display Classification report and confusion matrix for both classifiers
# Evaluate KNN classifier
print('\nKNN Classifier:')
print(f'Accuracy on training set: {accuracy_score(y_train, y_train_pred_knn)}')
print(f'Accuracy on testing set: {accuracy_score(y_test, y_test_pred_knn)}')
print('Classification Report (Testing set):')
print(classification_report(y_test, y_test_pred_knn))
print('Confusion Matrix (Testing set):')
print(confusion_matrix(y_test, y_test_pred_knn))

print("\n-------------------------------------------------------------------------")

# Evaluate Naive Bayes classifier
print('\nNaive Bayes Classifier:')
print(f'Accuracy on training set: {accuracy_score(y_train, y_train_pred_nb)}')
print(f'Accuracy on testing set: {accuracy_score(y_test, y_test_pred_nb)}')
print('Classification Report (Testing set):')
print(classification_report(y_test, y_test_pred_nb))
print('Confusion Matrix (Testing set):')
print(confusion_matrix(y_test, y_test_pred_nb))
```

```
KNN Classifier:
Accuracy on training set: 0.8577656123276561
Accuracy on testing set: 0.813463098134631
Classification Report (Testing set):
              precision    recall  f1-score   support

       False       0.84      0.96      0.90      2055
        True       0.25      0.06      0.10       411

    accuracy                           0.81      2466
   macro avg       0.54      0.51      0.50      2466
weighted avg       0.74      0.81      0.76      2466

Confusion Matrix (Testing set):
[[1981   74]
 [ 386   25]]

--------------------------------------------------------------------------

Naive Bayes Classifier:
Accuracy on training set: 0.811536901865369
Accuracy on testing set: 0.810624493106245
Classification Report (Testing set):
              precision    recall  f1-score   support

       False       0.85      0.94      0.89      2055
        True       0.36      0.18      0.24       411

    accuracy                           0.81      2466
   macro avg       0.61      0.56      0.57      2466
weighted avg       0.77      0.81      0.78      2466

Confusion Matrix (Testing set):
[[1925  130]
 [ 337   74]]
```

In [ ]:
```
# Provide suitable inferences for the models created
```

Inferences

Clssification inference

For Naive bayes Accuracy on training set: 0.811536901865369 Accuracy on testing set: 0.810624493106245

For KNN Accuracy on training set: 0.8577656123276561 Accuracy on testing set: 0.813463098134631

knn has better accuracy for a very small variation precision is better for Naive bayes for true classification recall is better for KNN for true classification

Clustering Inference

We could see that the k = 3 has a sillhouttes score value than k = 8 hence k =3 is a more accurate value for the number of clusters