SENTIMENT ANALYSIS: IMDB Movie review dataset

In [32]:
```
# We import neccessary libraries
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split, cross_val_score, KFold
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

In [33]:
```
# We load the IMDB dataset and view the top 10 entries
df = pd.read_csv("C:/Users/Sanayak/Desktop/IMDB Dataset.csv")
df.head(10)
```

Out[33]:

	review	sentiment
0	One of the other reviewers has mentioned that ...	positive
1	A wonderful little production. The...	positive
2	I thought this was a wonderful way to spend ti...	positive
3	Basically there's a family where a little boy ...	negative
4	Petter Mattei's "Love in the Time of Money" is...	positive
5	Probably my all-time favorite movie, a story o...	positive
6	I sure would like to see a resurrection of a u...	positive
7	This show was an amazing, fresh & innovative i...	negative
8	Encouraged by the positive comments about this...	negative
9	If you like original gut wrenching laughter yo...	positive

In [34]:
```
df.tail()
```

Out[34]:

	review	sentiment
49995	I thought this movie did a down right good job...	positive
49996	Bad plot, bad dialogue, bad acting, idiotic di...	negative
49997	I am a Catholic taught in parochial elementary...	negative
49998	I'm going to have to disagree with the previou...	negative
49999	No one expects the Star Trek movies to be high...	negative

In [35]:
```
df.describe()
```

Out[35]:

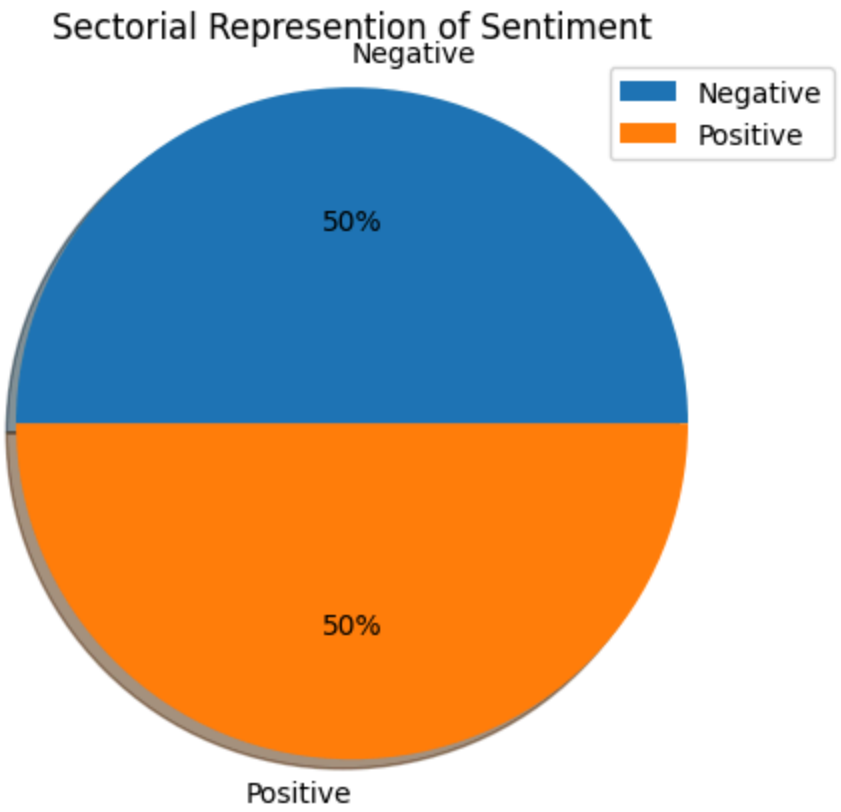
	review	sentiment
count	50000	50000
unique	49582	2
top	Loved today's show!!! It was a variety and not...	positive
freq	5	25000

In [36]:
```
# Categorical count on the sentiment
sentiment_count = pd.Categorical(df["sentiment"])
sentiment_count.describe()
```

Out[36]:

	counts	freqs
categories		
negative	25000	0.5
positive	25000	0.5

In [37]:
```
# Visual representation
counts=[2500,2500]
sentiment=["Negative","Positive"]
plt.pie(counts,labels=sentiment,shadow=True,autopct="%1.f%%")
plt.title("Sectorial Represention of Sentiment")
plt.axis("equal")
plt.legend()
plt.show()
```



In [38]:
```
# We use the map funtion to replace positive and negative review to 1 and 0 respect
df["sentiment"] = df["sentiment"].map({"positive":1,"negative":0})
df.head(10)
```

Out[38]:

	review	sentiment
0	One of the other reviewers has mentioned that ...	1
1	A wonderful little production. The...	1
2	I thought this was a wonderful way to spend ti...	1
3	Basically there's a family where a little boy ...	0
4	Petter Mattei's "Love in the Time of Money" is...	1
5	Probably my all-time favorite movie, a story o...	1
6	I sure would like to see a resurrection of a u...	1
7	This show was an amazing, fresh & innovative i...	0
8	Encouraged by the positive comments about this...	0
9	If you like original gut wrenching laughter yo...	1

In [39]:
```
df.nunique()
```

Out[39]:
```
review       49582
sentiment        2
dtype: int64
```

In [40]:
```
# Split data into training and testing set
X = df["review"] #feature variable
y = df["sentiment"] #Target Variable
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_s
# We convert text data to numerical vectors using TF-IDF vectorizer
vectorizer = TfidfVectorizer()
X_train = vectorizer.fit_transform(X_train)
X_test = vectorizer.transform(X_test)
```

In [48]:
```
# We evaluate using k-nearest neighbor
knn = KNeighborsClassifier()
knn.fit(X_train,y_train)
y_pred = knn.predict(X_test)
# Calculate classification report
class_rep = classification_report(y_test,y_pred)
print('Classification Report')
print(class_rep)
# Calculate confussion matrix
con_matrix = confusion_matrix(y_test,y_pred)
print('Confusion Matrix')
print(con_matrix)
```

```
Classification Report
              precision    recall  f1-score   support

           0       0.79      0.73      0.76      4961
           1       0.75      0.81      0.78      5039

    accuracy                           0.77     10000
   macro avg       0.77      0.77      0.77     10000
weighted avg       0.77      0.77      0.77     10000

Confusion Matrix
[[3620 1341]
 [ 969 4070]]
```

In [51]:
```
# We evaluate using Logistics Regresssion
logreg = LogisticRegression()
logreg.fit(X_train,y_train)
y_pred = logreg.predict(X_test)
# Calculate classification report
class_rep = classification_report(y_test,y_pred)
print('Classification Report')
print(class_rep)
# Calculate confussion matrix
con_matrix = confusion_matrix(y_test,y_pred)
print('Confusion Matrix')
print(con_matrix)
```

```
Classification Report
              precision    recall  f1-score   support

           0       0.91      0.89      0.90      4961
           1       0.89      0.91      0.90      5039

    accuracy                           0.90     10000
   macro avg       0.90      0.90      0.90     10000
weighted avg       0.90      0.90      0.90     10000

Confusion Matrix
[[4401  560]
 [ 441 4598]]
```

INTERPRETATION

The F1 scores and precision scores provided indicates that the logistic regression model performs significantly better than the K-nearest neighbors (KNN) model on our dataset. Using the logistic regression model with an F1 score of 0.90 would generally be preferred over the KNN model with an F1 score of 0.76. It promises higher accuracy, better balance between precision and recall, interpretability, and computational efficiency, making it a more reliable choice for many classification tasks.

PREPARED AND TRAINED BY VICTOR INIOBONG ECONOMIST | DATA SCIENTIST AND ANALYST | MATHEMATICS TUTOR