SENTIMENT ANALYSIS ON MOVIE REVIEWS

- Algorithm: Naive Bayes Classifier, Random Forest Classifier, K-Nearest Neighbors (KNN), xgboosting classifier,logistic Regression, Decision Tree Classifier,
- Description: Perform sentiment analysis on movie reviews to determine if the sentiment is positive or negative.
- For dataset - here

In [3]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from xgboost import XGBClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
import warnings
warnings.filterwarnings('ignore')
```

In [4]:
```
# Load the dataset
df = pd.read_csv('IMDB Dataset.csv')
df
```

Out[4]:

	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
...	...	...
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

50000 rows × 2 columns

In [5]:
```
print("The Number of rows =",df.shape[0])
print("The Number of columns =",df.shape[1])
```

```
The Number of rows = 50000
The Number of columns = 2
```

In [6]:
```
# Display the first few rows of the dataset to understand its structure
df.head()
```

Out[6]:

	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

In [7]:
```
# Display the last few rows of the dataset to understand its structure
df.tail()
```

Out[7]:

	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 [8]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   review     50000 non-null  object
 1   sentiment  50000 non-null  object
dtypes: object(2)
memory usage: 781.4+ KB
```

In [9]:
```
pd.isnull(df).sum()
```

Out[9]:
```
review       0
sentiment    0
dtype: int64
```

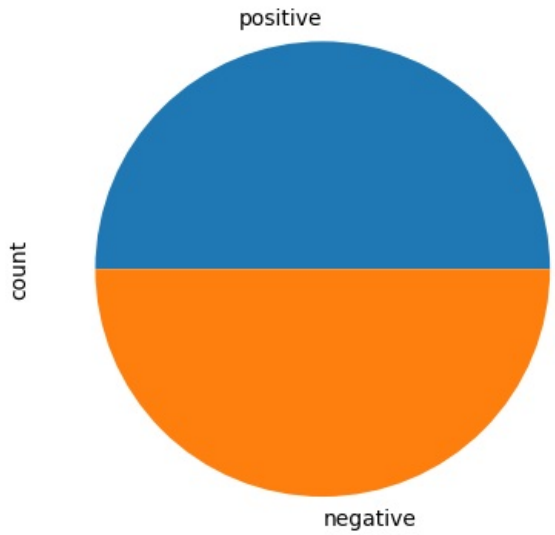
In [10]:
```
df.describe()
```

Out[10]:

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

In [11]:
```
df['sentiment'].value_counts().plot(kind='pie')
```

Out[11]:
```
<Axes: ylabel='count'>
```



In [12]:
```
# Encode the sentiments
df['sentiment'] = df['sentiment'].map({'positive': 1, 'negative': 0})
```

In [13]:
```
# Split the data into features and labels
X = df['review']
y = df['sentiment']
```

In [14]:
```
# Split dataset into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [15]:
```
# Convert text data to numerical data using TF-IDF
vectorizer = TfidfVectorizer(stop_words='english', max_features=5000)
X_train_vec = vectorizer.fit_transform(X_train)
X_test_vec = vectorizer.transform(X_test)
```

In [16]:
```
# Define models
models = [
```

```
    ("Naive Bayes", MultinomialNB()),
    ("Random Forest", RandomForestClassifier(random_state=42)),
    ("K-Nearest Neighbors", KNeighborsClassifier()),
    ("XGBoost", XGBClassifier(use_label_encoder=False, eval_metric='logloss', random_state=42)),
    ("Logistic Regression", LogisticRegression(max_iter=200)),
    ("Decision Tree", DecisionTreeClassifier(random_state=42))
]
```

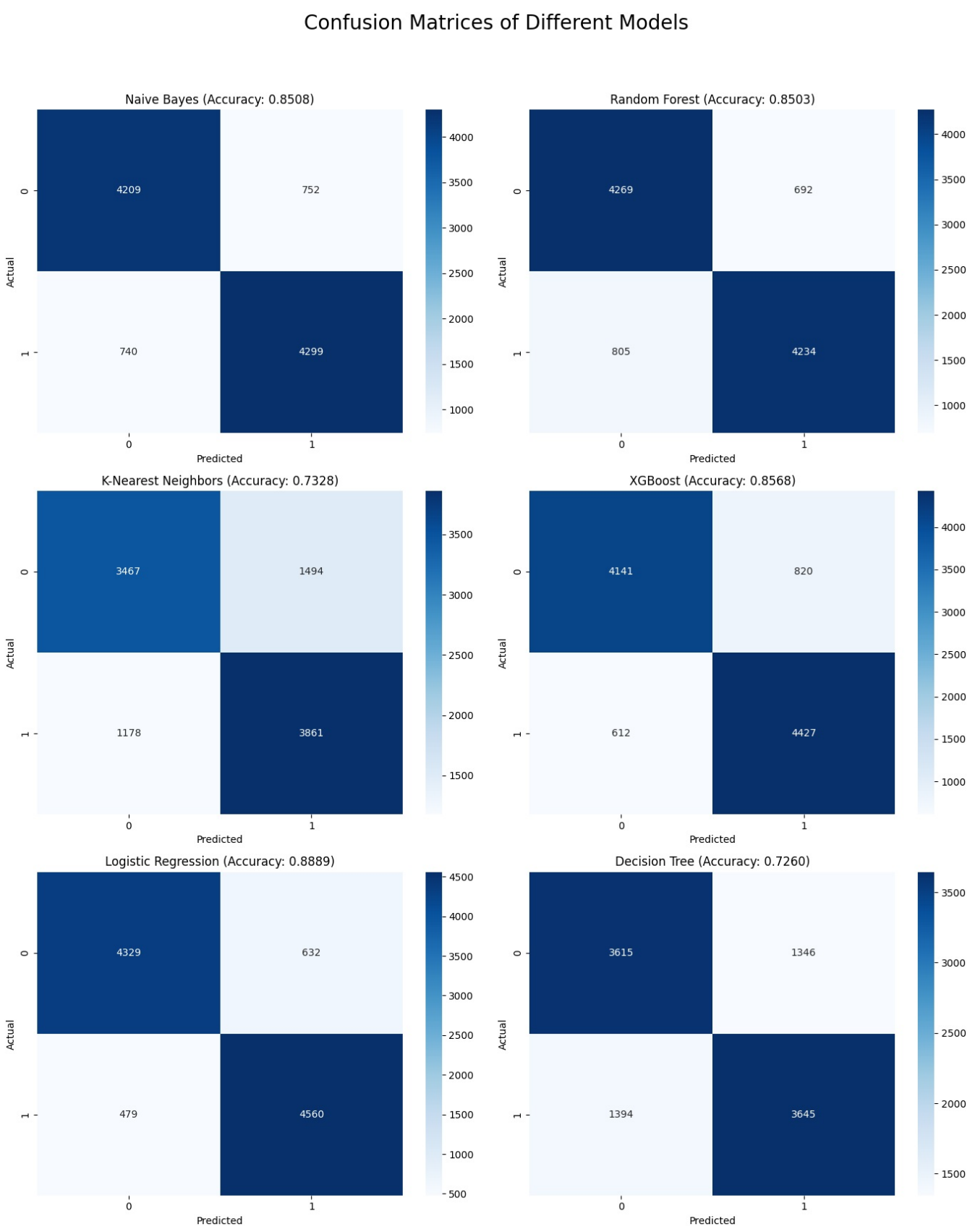
In [17]:

```
# Train and evaluate models, storing results
results = {}
for name, model in models:
    model.fit(X_train_vec, y_train)
    predictions = model.predict(X_test_vec)
    accuracy = accuracy_score(y_test, predictions)
    conf_matrix = confusion_matrix(y_test, predictions)

    results[name] = {
        "Accuracy": accuracy,
        "Confusion Matrix": conf_matrix
    }

    print(f"{name} Accuracy: {accuracy:.4f}")
    print(classification_report(y_test, predictions, target_names=["Negative", "Positive"]))
    print("-" * 50)
```

```
Naive Bayes Accuracy: 0.8508
              precision    recall  f1-score   support

    Negative       0.85      0.85      0.85      4961
    Positive       0.85      0.85      0.85      5039

    accuracy                           0.85     10000
   macro avg       0.85      0.85      0.85     10000
weighted avg       0.85      0.85      0.85     10000

--------------------------------------------------
Random Forest Accuracy: 0.8503
              precision    recall  f1-score   support

    Negative       0.84      0.86      0.85      4961
    Positive       0.86      0.84      0.85      5039

    accuracy                           0.85     10000
   macro avg       0.85      0.85      0.85     10000
weighted avg       0.85      0.85      0.85     10000

--------------------------------------------------
K-Nearest Neighbors Accuracy: 0.7328
              precision    recall  f1-score   support

    Negative       0.75      0.70      0.72      4961
    Positive       0.72      0.77      0.74      5039

    accuracy                           0.73     10000
   macro avg       0.73      0.73      0.73     10000
weighted avg       0.73      0.73      0.73     10000

--------------------------------------------------
XGBoost Accuracy: 0.8568
              precision    recall  f1-score   support

    Negative       0.87      0.83      0.85      4961
    Positive       0.84      0.88      0.86      5039

    accuracy                           0.86     10000
   macro avg       0.86      0.86      0.86     10000
weighted avg       0.86      0.86      0.86     10000

--------------------------------------------------
Logistic Regression Accuracy: 0.8889
              precision    recall  f1-score   support

    Negative       0.90      0.87      0.89      4961
    Positive       0.88      0.90      0.89      5039

    accuracy                           0.89     10000
   macro avg       0.89      0.89      0.89     10000
weighted avg       0.89      0.89      0.89     10000

--------------------------------------------------
Decision Tree Accuracy: 0.7260
              precision    recall  f1-score   support

    Negative       0.72      0.73      0.73      4961
    Positive       0.73      0.72      0.73      5039

    accuracy                           0.73     10000
   macro avg       0.73      0.73      0.73     10000
weighted avg       0.73      0.73      0.73     10000

--------------------------------------------------
```

In [18]:

```python
# Plot the confusion matrices
fig, axes = plt.subplots(3, 2, figsize=(14, 18))
fig.suptitle('Confusion Matrices of Different Models', fontsize=20)

for (name, result), ax in zip(results.items(), axes.flatten()):
    sns.heatmap(result['Confusion Matrix'], annot=True, fmt='d', cmap='Blues', ax=ax)
    ax.set_title(f'{name} (Accuracy: {result["Accuracy"]:.4f})')
    ax.set_xlabel('Predicted')
    ax.set_ylabel('Actual')

plt.tight_layout(rect=[0, 0.03, 1, 0.95])
plt.show()
```



In [19]:
```
# New dataset for testing the models
new_reviews = ["I loved the movie, it was wonderful!", "I hated the film, it was terrible."]
new_reviews_vec = vectorizer.transform(new_reviews)
```

In [20]:
```
# Dictionary to store predictions
model_predictions = {}

# Predict sentiments using each model
for name, model in models:
    new_predictions = model.predict(new_reviews_vec)
    sentiment_labels = ['negative' if pred == 0 else 'positive' for pred in new_predictions]
    model_predictions[name] = sentiment_labels
```

In [21]:
```
# Print predictions for each model
for name, predictions in model_predictions.items():
    print(f"{name} Predictions: {predictions[0]}, {predictions[1]}")
```

```
Naive Bayes Predictions: positive, negative
Random Forest Predictions: positive, negative
K-Nearest Neighbors Predictions: positive, negative
XGBoost Predictions: positive, negative
Logistic Regression Predictions: positive, negative
Decision Tree Predictions: positive, negative
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: