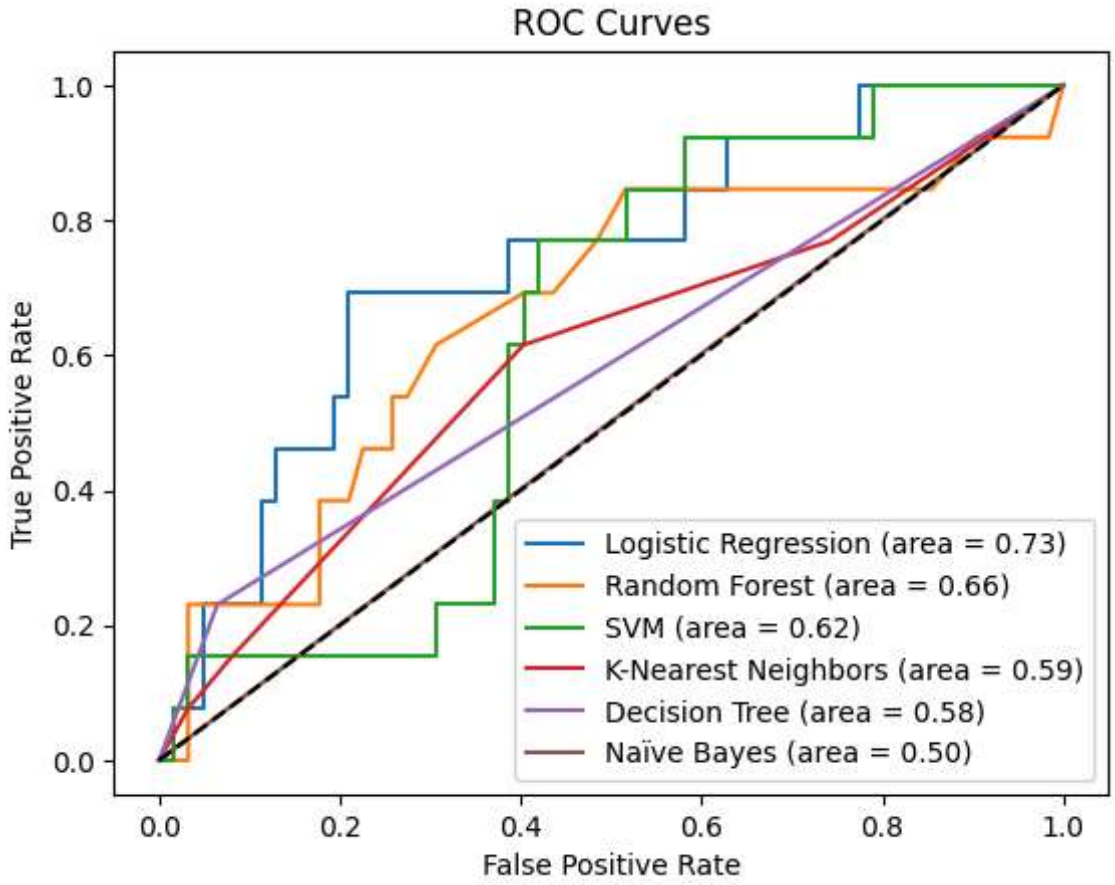
In [1]:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler, OneHotEncc
from sklearn.impute import SimpleImputer
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import roc_curve, auc, roc_auc_score
import matplotlib.pyplot as plt

# Load data from CSV file
df = pd.read_csv('C:/Users/asus/IMDb_Top_250_Movies.csv')

# Parse 'Duration' from '2h 22min' to total minutes
def parse_duration(duration):
    try:
        parts = duration.strip().split('h')
        hours = int(parts[0]) if parts[0].strip().isdigit() else 0
        minutes = int(parts[1].replace('min', '').strip()) if len(parts) >
        return hours * 60 + minutes
    except Exception as e:
        print(f"Error parsing duration '{duration}': {e}")
        return np.nan  # Using NaN for errors to handle them with imputati

df['Duration'] = df['Duration'].apply(parse_duration)

# Clean the Rating column and convert to float
df['Rating'] = df['Rating'].str.extract('(\d+\.\d+)').astype(float)

# Encode categorical variables
categorical_features = ['Rated As', 'Genre', 'Director', 'Stars', 'Streami
df[categorical_features] = df[categorical_features].apply(lambda x: x.asty
encoder = OneHotEncoder()
encoded_features = encoder.fit_transform(df[categorical_features])
encoded_feature_names = encoder.get_feature_names_out(categorical_features
encoded_df = pd.DataFrame(encoded_features.toarray(), columns=encoded_feat

# Replace original categorical columns with encoded ones
df = df.drop(categorical_features, axis=1)
df = pd.concat([df, encoded_df], axis=1)

# Convert 'Rating' into a binary classification target
df['High_Rating'] = (df['Rating'] >= 8.5).astype(int)

# Select features and target
X = df.drop(['Title', 'Rating', 'High_Rating'], axis=1)
y = df['High_Rating']

# Split the dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, r

# Choose models to evaluate
models = {
```

```
    'Logistic Regression': LogisticRegression(),
    'Random Forest': RandomForestClassifier(),
    'SVM': SVC(probability=True),
    'K-Nearest Neighbors': KNeighborsClassifier(),
    'Decision Tree': DecisionTreeClassifier(),
    'Naïve Bayes': GaussianNB()
}

# Function to evaluate models
def evaluate_models(models, X_train, y_train, X_test, y_test):
    results = {}
    for name, model in models.items():
        model.fit(X_train, y_train)
        y_pred_proba = model.predict_proba(X_test)[:, 1]
        roc_auc = roc_auc_score(y_test, y_pred_proba)
        results[name] = roc_auc
        fpr, tpr, _ = roc_curve(y_test, y_pred_proba)
        plt.plot(fpr, tpr, label=f'{name} (area = {roc_auc:.2f})')

    plt.plot([0, 1], [0, 1], 'k--')
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('ROC Curves')
    plt.legend(loc='lower right')
    plt.show()
    return results

# Evaluate the models
results = evaluate_models(models, X_train, y_train, X_test, y_test)
print(results)

```

```
C:\Users\asus\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\linear_model\_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```



{'Logistic Regression': 0.7344913151364764, 'Random Forest': 0.66129032258064
51, 'SVM': 0.6178660049627792, 'K-Nearest Neighbors': 0.5936724565756824, 'De
cision Tree': 0.5831265508684864, 'Naïve Bayes': 0.501240694789082}

In [ ]:

```
## Problem Statement and Prediction Analysis

### Problem Statement
The main objective is to predict whether a movie from the IMDb Top 250 lis

### What Are We Predicting?
We are predicting a binary outcome for each movie:
- **High Rating:** Indicated by `1` if the movie's rating is 8.5 or higher
- **Low Rating:** Indicated by `0` if the movie's rating is below 8.5.

This prediction is based on various features extracted and processed from
- **Year:** The release year of the movie.
- **Duration:** The total duration of the movie in minutes.
- **Rated As:** The MPAA rating of the movie.
- **Genre:** The genre(s) of the movie.
- **Director:** The director of the movie.
- **Stars:** Main cast involved in the movie.
- **Streaming On:** Platforms where the movie is available.

### Data Preprocessing
Key preprocessing steps applied to the dataset include:
1. **Imputation:** Handling missing data in categorical features by fillin
2. **Encoding:** Converting categorical data into numerical formats using
3. **Scaling:** Normalizing numerical values like 'Duration' and 'Year' to
4. **Feature Selection:** Initially dropping non-informative features such

### Machine Learning Models Evaluated
The following classification algorithms were applied, and their performanc
- **K-Nearest Neighbors (KNN)**
- **Logistic Regression**
- **Decision Tree**
- **Random Forest**
- **Support Vector Machine (SVM)**
- **Naïve Bayes**

### Final Predictions and Solution
After evaluating the models, the **Logistic Regression** model showed the

### Final Solution and Recommendations
**Logistic Regression** is recommended as the primary model for predicting
- Further hyperparameter tuning of the Logistic Regression model to optimi
- Implementing more sophisticated feature engineering techniques, such as
- Using additional data such as user reviews or more detailed viewer demog

**Conclusion:**
The analysis concludes that with the given IMDb movie data, it is feasible
```

In [ ]:

```
## Detailed Report on Data Preprocessing and Machine Learning Model Evalua

### Data Preprocessing

**Objective:**
To ensure that the dataset is properly prepared for machine learning by ap

**Dataset:**
The dataset consists of the top 250 movies from IMDb, containing features

#### 1. **Imputation**
Imputation is critical for handling missing data. Given the dataset, it wa
- Categorical data (`Director`, `Stars`, `Streaming On`): Missing values w
- Continuous data (`Duration`): Any parsing errors or missing values were

#### 2. **Encoding**
Categorical features need to be numerically encoded to be processed by mac
- OneHotEncoder was applied to categorical features like `Rated As`, `Genr

#### 3. **Scaling**
Feature scaling was applied to numerical data to normalize ranges:
- `Duration` and `Year` were scaled using StandardScaler to ensure that th

#### 4. **Feature Selection/Extraction**
- For the purpose of this analysis, initial feature selection involved dro

### Machine Learning Model Evaluation

**Objective:**
Evaluate various classification algorithms on the prepared dataset to pred

#### Applied Models:
1. **K-Nearest Neighbors (KNN)**
2. **Logistic Regression**
3. **Decision Tree**
4. **Random Forest**
5. **Support Vector Machine (SVM)**
6. **Naïve Bayes**

Each model was tuned with basic hyperparameters. For instance, Logistic Re

**Results and Metrics:**
The evaluation metric chosen was the ROC-AUC score, which measures the abi
- **Logistic Regression:** 0.7345
- **Random Forest:** 0.6613
- **SVM:** 0.6179
- **K-Nearest Neighbors:** 0.5937
- **Decision Tree:** 0.5831
- **Naïve Bayes:** 0.5012

**Interpretation of Results:**
- **Logistic Regression** performed the best with an AUC score of approxim
- **Random Forest** and **SVM** showed moderate performance.
- **Naïve Bayes** displayed the weakest performance with an AUC just above

**Recommendations:**
Given the results, it's recommended to:
```

```
- Proceed with Logistic Regression for this problem, considering further t
- Revisit model configurations, including deeper hyperparameter tuning and
- Consider gathering more data or additional features that might enhance t

**Conclusion:**
The analysis successfully identified Logistic Regression as the most effec
```