ML PRACTICE ON THIS NETFLIX DATASET:

In [1]:

```
# Ignore all warnings:
import warnings
warnings.filterwarnings('ignore')
```

DATA-CLEANING PART OF THE PROJECT:

In [2]:

```
import pandas as pd

df = pd.read_csv("netflix_titles.csv")

df.head()
```

Out[2]:

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	d
0	81145628	Movie	Norm of the North: King Sized Adventure	Richard Finn, Tim Maltby	Alan Marriott, Andrew Toth, Brian Dobson, Cole...	United States, India, South Korea, China	September 9, 2019	2019	TV-PG	90 min	Children & Family Movies, Comedies	
1	80117401	Movie	Jandino: Whatever it Takes	NaN	Jandino Asporaat	United Kingdom	September 9, 2016	2016	TV-MA	94 min	Stand-Up Comedy	ri c
2	70234439	TV Show	Transformers Prime	NaN	Peter Cullen, Sumalee Montano, Frank Welker, J...	United States	September 8, 2018	2013	TV-Y7-FV	1 Season	Kids' TV	
3	80058654	TV Show	Transformers: Robots in Disguise	NaN	Will Friedle, Darren Criss, Constance Zimmer, ...	United States	September 8, 2018	2016	TV-Y7	1 Season	Kids' TV	p u
4	80125979	Movie	#realityhigh	Fernando Lebrija	Nesta Cooper, Kate Walsh, John Michael Higgins...	United States	September 8, 2017	2017	TV-14	99 min	Comedies	r D

In [3]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6234 entries, 0 to 6233
Data columns (total 12 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   show_id        6234 non-null   int64
 0   show_id       6234 non-null   int64
 1   type          6234 non-null   object
 2   title         6234 non-null   object
 3   director      4265 non-null   object
 4   cast          5664 non-null   object
 5   country       5758 non-null   object
 6   date_added    6223 non-null   object
 7   release_year  6234 non-null   int64
 8   rating        6224 non-null   object
 9   duration      6234 non-null   object
 10  listed_in     6234 non-null   object
 11  description   6234 non-null   object
dtypes: int64(2), object(10)
memory usage: 584.6+ KB
```

In [4]:

```
df.isnull().sum()
```

Out[4]:

```
show_id            0
type               0
title              0
director        1969
cast             570
country          476
date_added        11
release_year       0
rating            10
duration           0
listed_in          0
description        0
dtype: int64
```

In [5]:

```
# 1. Remove duplicates
df = df.drop_duplicates()

# 2. Handle missing values
# Fill missing 'rating' with mode
df['rating'] = df['rating'].fillna(df['rating'].mode()[0])

# Fill missing 'country' and 'date_added' with 'Unknown'
df['country'] = df['country'].fillna('Unknown')
df['date_added'] = df['date_added'].fillna('Unknown')

# Fill missing 'duration' with '0 min' for shows or '0 season' for TV Shows
df['duration'] = df['duration'].fillna('0')

# 3. Convert 'date_added' to datetime format if not Unknown
df['date_added'] = df['date_added'].replace('Unknown', pd.NaT)
df['date_added'] = pd.to_datetime(df['date_added'], errors='coerce')

# 4. Extract useful date features
df['year_added'] = df['date_added'].dt.year
df['month_added'] = df['date_added'].dt.month

# 5. Clean 'duration' column
# Extract only numbers from 'duration' (convert "90 min" -> 90)
import re
df['duration_num'] = df['duration'].apply(lambda x: int(re.findall(r'\d+', x)[0]) if re.findall(r'\d+', x) else 0)

# 6. Clean text columns (trim whitespaces)
df['title'] = df['title'].str.strip()
df['director'] = df['director'].fillna('Unknown').str.strip()
df['cast'] = df['cast'].fillna('Unknown').str.strip()

# 7. Split genres for analysis (expand the first genre)
df['main_genre'] = df['listed_in'].apply(lambda x: x.split(',')[0] if isinstance(x, str)
else 'Unknown')

# Create a new column that counts how many cast members are listed
df['num_cast'] = df['cast'].apply(lambda x: len(x.split(',')) if isinstance(x, str) else
0)

# Preview
df[['cast', 'num_cast']].head(5)
```

Out[5]:

	cast	num_cast
0	Alan Marriott, Andrew Toth, Brian Dobson, Cole...	10
1	Jandino Asporaat	1
2	Peter Cullen, Sumalee Montano, Frank Welker, J...	12
3	Will Friedle, Darren Criss, Constance Zimmer, ...	8
4	Nesta Cooper, Kate Walsh, John Michael Higgins...	12

In [ ]:

In [ ]:

In [ ]:

Q1] (LINEAR REGRESSION) Can we predict the duration (in minutes/seasons) based on the release year , type of content and number of cast members?

In [6]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6234 entries, 0 to 6233
Data columns (total 17 columns):
 #   Column        Non-Null Count  Dtype         
---  ------        --------------  -----         
 0   show_id       6234 non-null   int64         
 1   type          6234 non-null   object        
 2   title         6234 non-null   object        
 3   director      6234 non-null   object        
 4   cast          6234 non-null   object        
 5   country       6234 non-null   object        
 6   date_added    5583 non-null   datetime64[ns]
 7   release_year  6234 non-null   int64         
 8   rating        6234 non-null   object        
 9   duration      6234 non-null   object        
 10  listed_in     6234 non-null   object        
 11  description   6234 non-null   object        
 12  year_added    5583 non-null   float64       
 13  month_added   5583 non-null   float64       
 14  duration_num  6234 non-null   int64         
 15  main_genre    6234 non-null   object        
 16  num_cast      6234 non-null   int64         
dtypes: datetime64[ns](1), float64(2), int64(4), object(10)
memory usage: 828.1+ KB
```

In [7]:

```
from sklearn.preprocessing import LabelEncoder
```

```
le = LabelEncoder()
df["type_Encoded"] = le.fit_transform(df["type"])
```

In [8]:

```
from sklearn import linear_model
reg = linear_model.LinearRegression(positive=True)
reg.fit(df[["release_year","num_cast","type_Encoded"]],df["duration_num"])
reg.predict([[2019,3,1]])
```

Out[8]:

```
array([64.71613684])
```

In [ ]:

```

```

In [ ]:

```

```

In [9]:

```
df.head(3)
```

Out[9]:

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	des
0	81145628	Movie	Norm of the North: King Sized Adventure	Richard Finn, Tim Maltby	Alan Marriott, Andrew Toth, Brian Dobson, Cole...	United States, India, South Korea, China	2019-09-09	2019	TV-PG	90 min	Children & Family Movies, Comedies	p av v
1	80117401	Movie	Jandino: Whatever it Takes	Unknown	Jandino Asporaat	United Kingdom	2016-09-09	2016	TV-MA	94 min	Stand-Up Comedy	A riff cha
2	70234439	TV Show	Transformers Prime	Unknown	Peter Cullen, Sumalee Montano, Frank Welker, J...	United States	2018-09-08	2013	TV-Y7-FV	1 Season	Kids' TV	\ al

Q2] (LINEAR REGRESSION) Can we estimate when a show or movie was released based on its genre, rating, and type (Movie/TV Show)?

In [10]:

```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
df["rating_encoded"] = le.fit_transform(df["rating"])
df["listed_in_encoded"] = le.fit_transform(df["listed_in"])


from sklearn import linear_model
reg = linear_model.LinearRegression(positive=True)
reg.fit(df[["listed_in_encoded","rating_encoded","type_Encoded"]],df["release_year"])
reg.predict([[110,9,0]])
```

Out[10]:

```
array([2012.7429627])
```

In [ ]:

Q3] (DECISION TREE) Can we predict whether a Netflix title is a Movie or a TV Show based on its duration, release year, and rating?

In [11]:

```
from sklearn.preprocessing import LabelEncoder

inputs = df.copy()
le_duration = LabelEncoder()
le_release_year = LabelEncoder()
le_rating = LabelEncoder()

inputs["duration_n"] = le_duration.fit_transform(df["duration_num"])
inputs["release_year_n"] = le_release_year.fit_transform(df["year_added"])
inputs["rating_n"] = le_rating.fit_transform(df["rating"])
inputs["listed_in_n"] = le_rating.fit_transform(df["listed_in"])
inputs["year_added_n"] = le_rating.fit_transform(df["year_added"])
```

In [12]:

```
inputs.drop(columns=["show_id","title","director","cast","country","date_added","release
_year","rating","duration","month_added","duration_num","main_genre","num_cast","type_Enc
oded","rating_encoded","listed_in_encoded","description"],axis=1,inplace=True)
```

In [13]:

```
inputs.head(5)
```

Out[13]:

	type	listed_in	year_added	duration_n	release_year_n	rating_n	listed_in_n	year_added_n
0	Movie	Children & Family Movies, Comedies	2019.0	85	11	9	110	11
1	Movie	Stand-Up Comedy	2016.0	89	8	8	420	8
2	TV Show	Kids' TV	2018.0	0	10	12	381	10
3	TV Show	Kids' TV	2018.0	0	10	11	381	10
4	Movie	Comedies	2017.0	94	9	6	167	9

In [14]:

```
outputs = inputs.copy()
outputs.drop(columns=["listed_in","year_added","duration_n","release_year_n","rating_n",
"listed_in_n","year_added_n"],axis=1,inplace=True)
```

In [15]:

```
outputs.head(5)
```

Out[15]:

	type
0	Movie
1	Movie
2	TV Show
3	TV Show

4	Mtypie

In [16]:

```
inputs.drop(columns=["type","listed_in","year_added"],axis=1,inplace=True)
```

In [17]:

```
inputs.head(5)
```

Out[17]:

	duration_n	release_year_n	rating_n	listed_in_n	year_added_n
0	85	11	9	110	11
1	89	8	8	420	8
2	0	10	12	381	10
3	0	10	11	381	10
4	94	9	6	167	9

In [18]:

```
outputs.head(5)
```

Out[18]:

	type
0	Movie
1	Movie
2	TV Show
3	TV Show
4	Movie

In [19]:

```
from sklearn import tree
model = tree.DecisionTreeClassifier()

model.fit(inputs,outputs)
model.score(inputs,outputs)
```

Out[19]:

```
1.0
```

In [20]:

```
model.predict([[89,8,8,420,8]])
```

Out[20]:

```
array(['Movie'], dtype=object)
```

In [21]:

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(inputs, outputs, test_size=0.2, random_state=42)

model = tree.DecisionTreeClassifier()
model.fit(X_train, y_train)

print("Training Accuracy:", model.score(X_train, y_train))
print("Testing Accuracy:", model.score(X_test, y_test))
```

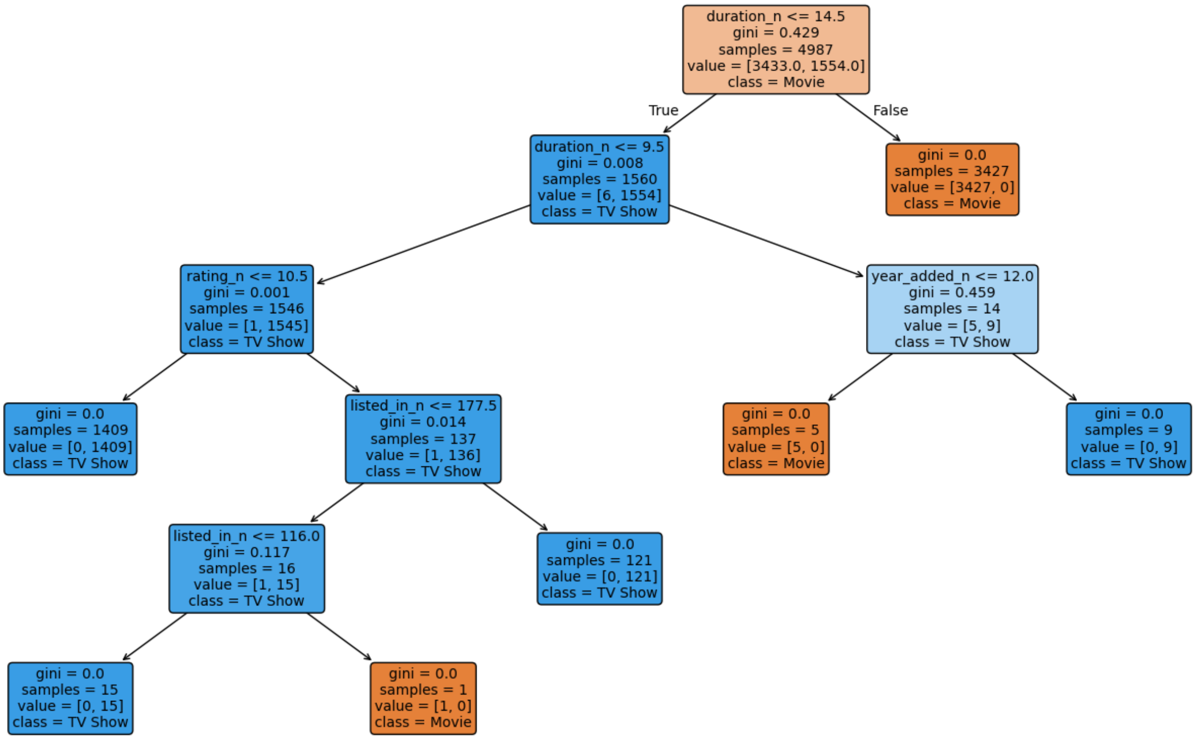
```
Training Accuracy: 1.0
Testing Accuracy: 0.9991980753809142
```

In [22]:

```
from sklearn import tree
import matplotlib.pyplot as plt


# ⍰ Plot the Decision Tree
plt.figure(figsize=(18,10))  # Bigger figure for better readability
tree.plot_tree(
    model,
    feature_names=inputs.columns,     # names of your input features
    class_names=['Movie', 'TV Show'],# labels (adjust as per your encoding)
    filled=True,                      # fill colors to visualize class splits
    rounded=True,                     # rounded node boxes
    fontsize=10
)
plt.title("Decision Tree Visualization for Netflix Titles", fontsize=16)
plt.show()
```

Decision Tree Visualization for Netflix Titles



In [ ]:

In [ ]:

In [ ]:

KMEANS QUESTION:

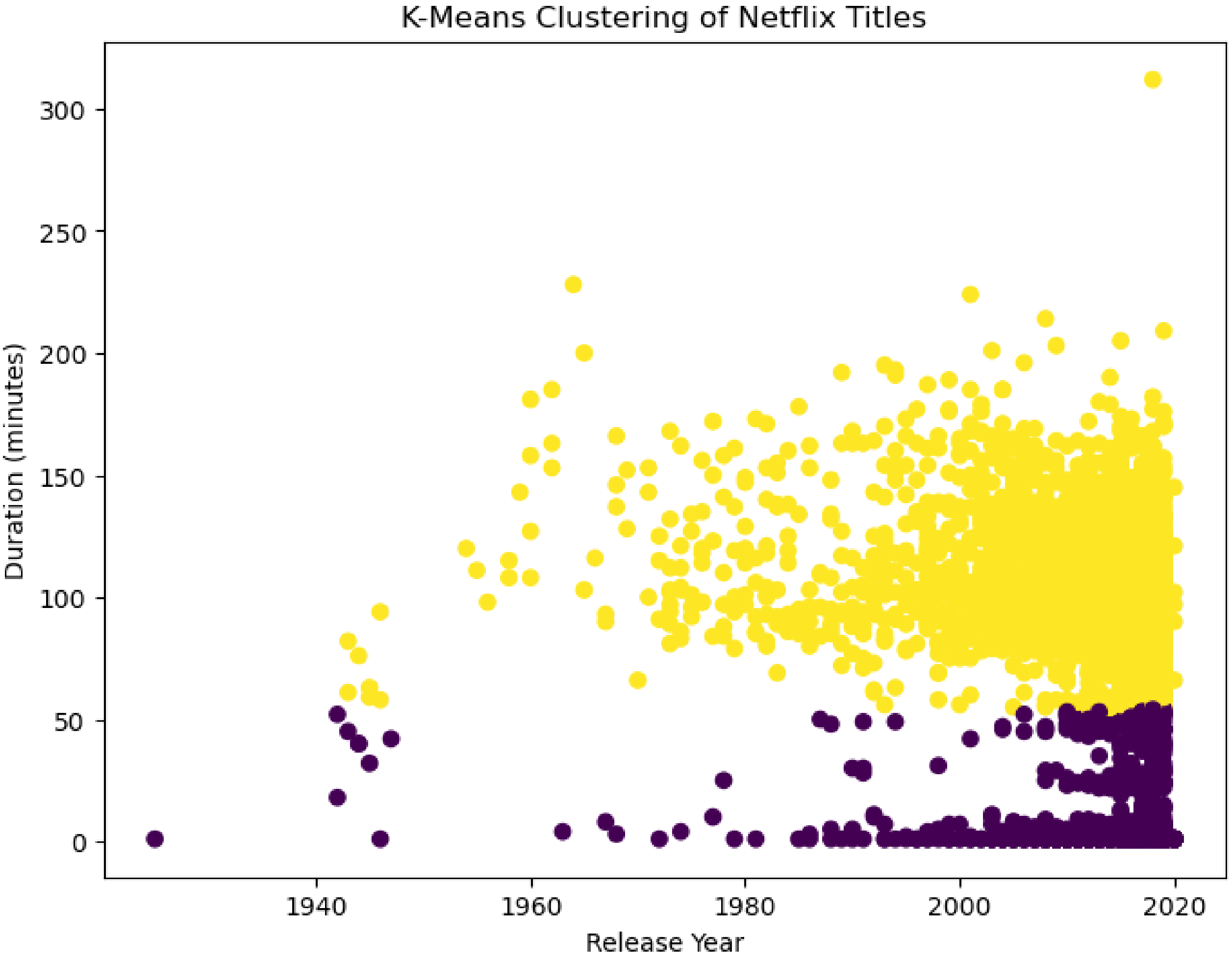
In [24]:

```
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
```

```
# Select features
dataset = df[["release_year", "duration_num", "type_Encoded"]].dropna()

# Apply K-Means
kmeans = KMeans(n_clusters=2, random_state=42)
dataset["cluster"] = kmeans.fit_predict(dataset[["release_year", "duration_num", "type_E
ncoded"]])

# Plot clusters
plt.figure(figsize=(8,6))
plt.scatter(dataset["release_year"], dataset["duration_num"], c=dataset["cluster"], cmap
='viridis')
plt.xlabel("Release Year")
plt.ylabel("Duration (minutes)")
plt.title("K-Means Clustering of Netflix Titles")
plt.show()
```



```
In [ ]:
```

Q3] (USING RANDOM FOREST)

```
In [25]:
inputs.head(5)
Out[25]:
```

	duration_n	release_year_n	rating_n	listed_in_n	year_added_n
0	85	11	9	110	11
1	89	8	8	420	8
2	0	10	12	381	10
3	0	10	11	381	10

	duration_n	release_year_n	rating_n	listed_in_n	year_added_n
4	94	9	6	167	9

In [26]:

```
outputs.head(5)
```

Out[26]:

	type
0	Movie
1	Movie
2	TV Show
3	TV Show
4	Movie

In [27]:

```
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
X_train , X_test , y_train, y_test = train_test_split(inputs,outputs,test_size=0.2, rand
om_state=42)

from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(n_estimators=20)
model.fit(X_train,y_train)

model.score(X_test, y_test)
```

Out[27]:

```
0.9991980753809142
```

In [28]:

```
y_predicted = model.predict(X_test)
```

Confusion Matrix:

In [29]:

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_predicted)
cm
```

Out[29]:

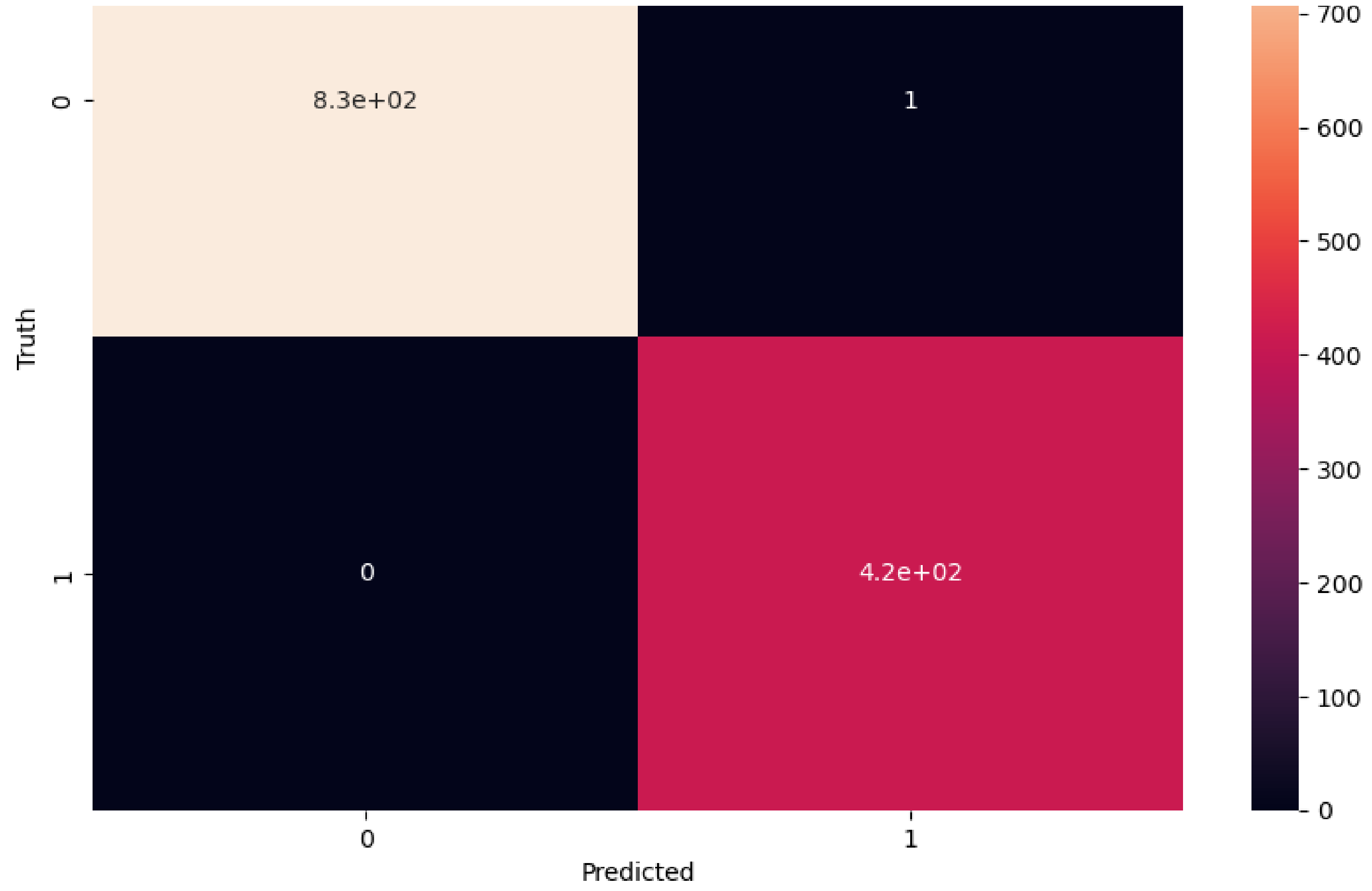
```
array([[831,   1],
       [  0, 415]])
```

In [30]:

```
import matplotlib.pyplot as plt
import seaborn as sn
plt.figure(figsize=(10,7))
sn.heatmap(cm, annot=True)
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

Out[30]:

```
Text(95.72222222222221, 0.5, 'Truth')
```

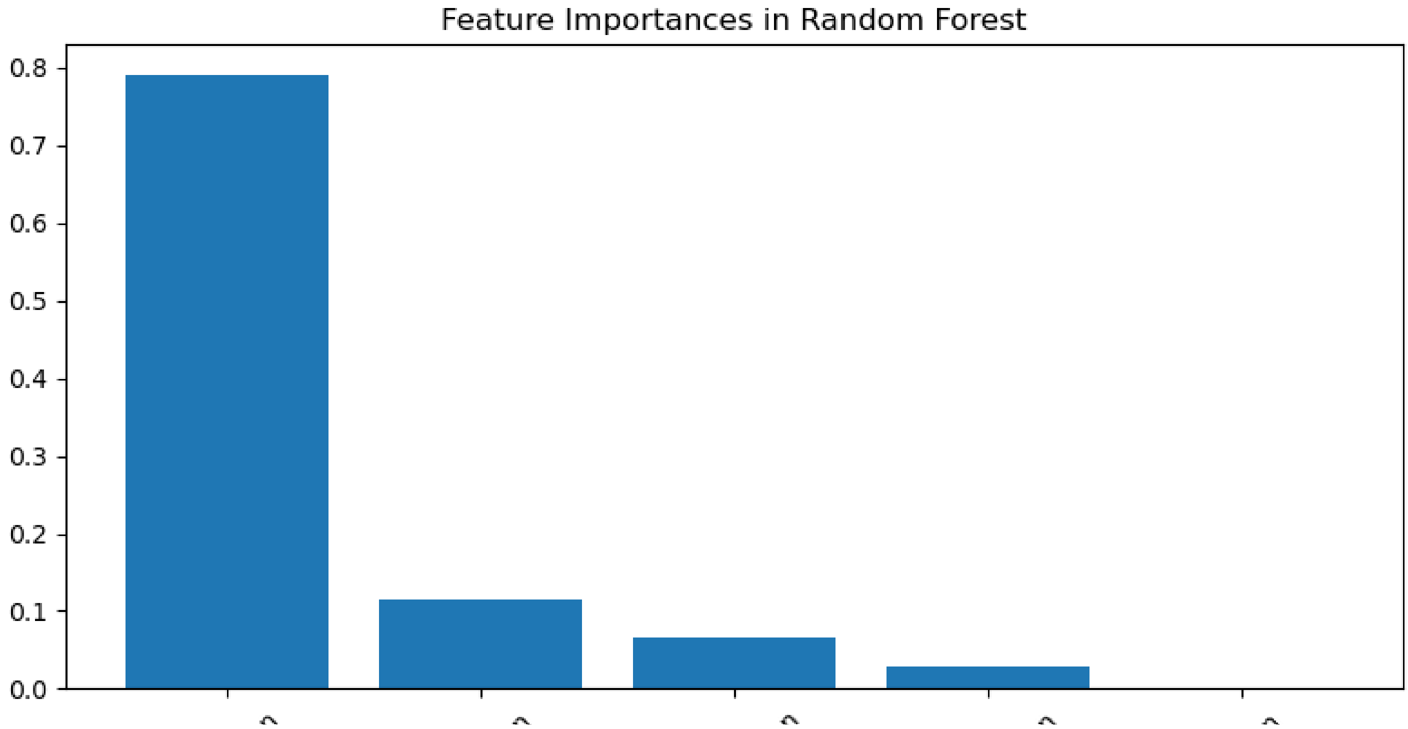


In [33]:

```
import numpy as np
import matplotlib.pyplot as plt

# Get feature importances
importances = model.feature_importances_
indices = np.argsort(importances)[::-1]

# Plot
plt.figure(figsize=(8,5))
plt.title("Feature Importances in Random Forest")
plt.bar(range(len(importances)), importances[indices], align="center")
plt.xticks(range(len(importances)), [inputs.columns[i] for i in indices], rotation=45)
plt.tight_layout()
plt.show()
```



In [ ]:

```

```

In [ ]:

```

```

Q4] (LOGISTIC REGRESSION (multi-class classification)) Can we predict whether a Netflix title is a Movie or a TV Show based on its duration, release year, and rating?

In [58]:

```
inputs
```

Out[58]:

	duration_n	release_year_n	rating_n	listed_in_n	year_added_n
0	85	11	9	110	11
1	89	8	8	420	8
2	0	10	12	381	10
3	0	10	11	381	10
4	94	9	6	167	9
...	...	...	...	...	...
6229	12	13	2	427	13
6230	3	13	8	435	13
6231	55	13	8	396	13
6232	1	13	8	104	13
6233	9	13	6	142	13

6234 rows × 5 columns

In [59]:

```
outputs["type_encoded"] = outputs["type"].map({"Movie": 0, "TV Show": 1})
```

In [60]:

```
outputs
```

Out[60]:

	type	type_encoded
0	Movie	0
1	Movie	0
2	TV Show	1
3	TV Show	1
4	Movie	0
...	...	...
6229	TV Show	1
6230	TV Show	1

	type	type_encoded
6231	Movie	0
6232	TV Show	1
6233	TV Show	1

6234 rows × 2 columns

In [62]:

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()

from sklearn.model_selection import train_test_split

X_train , X_test , y_train, y_test = train_test_split(inputs,outputs["type_encoded"],tes
t_size=0.3)
```

In [63]:

```
model.fit(X_train , y_train)
```

Out[63]:

```
▼
 LogisticRegression
                  i ?

▶ Parameters
```

In [64]:

```
# Measure accuracy of the model created:
model.score(X_test , y_test)
```

Out[64]:

```
1.0
```

In [69]:

```
prediction_made = model.predict([[94,9,6,167,10]])

if prediction_made==1:
    print("TV Show")
elif prediction_made==0:
    print("Movie")
else:
    print("Prediction wasn't accurate")
```

```
TV Show
```

In [ ]: