```
import matplotlib.pyplot as plt
import pandas as pd
```

```
data = pd.read_csv('/content/tv_shows (1).csv')
```

```
data.head()
```

	Unnamed: 0	ID	Title	Year	Age	IMDb	Rotten Tomatoes	Netflix	Hulu	Prime Video	Disney+	Typ
0	0	1	Breaking Bad	2008	5	1.1/10	100/100	1	0	0	0	
1	1	2	Stranger Things	2016	5	1.5/10	96/100	1	0	0	0	
2	2	3	Attack on Titan	2013	5	1.8/10	95/100	1	1	0	0	

Next steps: View recommended plots

```
data['IMDb']=data['IMDb'].str.replace('/10',' ').astype(float)
```

```
data.head()
```

	Unnamed: 0	ID	Title	Year	Age	IMDb	Rotten Tomatoes	Netflix	Hulu	Prime Video	Disney+	Type
0	0	1	Breaking Bad	2008	5	1.1	100/100	1	0	0	0	1
1	1	2	Stranger Things	2016	5	1.5	96/100	1	0	0	0	1
2	2	3	Attack on Titan	2013	5	1.8	95/100	1	1	0	0	1

Next steps: View recommended plots

```
feature_cols = ['Age','IMDb']
X = data.iloc[:,[1,5]].values
y = data.iloc[:,4].values
```

```
print(X)
print(y)
```

```
[[1.000e+00 1.100e+00]
 [2.000e+00 1.500e+00]
 [3.000e+00 1.800e+00]
 ...
 [5.613e+03 9.500e+00]
 [5.615e+03 9.600e+00]
 [5.616e+03 9.600e+00]]
[ 5  5  5 ... 18 18 18]
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = 0.3, random_state= 1)
```

```
#feature scaling
```

```
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)
print(X_train)
```

```
[[-1.46654646 -2.42963572]
 [ 0.03446687  0.23409348]
 [ 0.4347371   0.50965167]
 ...
 [-0.59161369 -0.22517018]
 [-1.37108728 -1.60296114]
 [-0.62584733 -0.31702291]]
```

```
print(X_test)
```

```
[[ 0.04105026  0.23409348]
 [-0.69628962 -0.31702291]
 [ 0.87714103  0.78520987]
 ...
 [-0.53762988 -0.22517018]
 [ 2.05556816  1.33632625]
 [ 1.11085144  0.96891533]]
```

```
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier() #we are using default parameters: criteria = gini, r
classifier = classifier.fit(X_train,y_train)
```

```
#prediction
y_pred = classifier.predict(X_test)#Accuracy
from sklearn import metrics
print('Accuracy Score:', metrics.accuracy_score(y_test,y_pred))
```

```
Accuracy Score: 0.9968847352024922
```

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

```
[[170   1   0   0   0]
 [  0 256   0   0   0]
 [  0   0   2   0   0]
 [  0   0   1 297   0]
 [  0   0   0   1 235]]
```

```
# Define a function to filter recommendations for ages 5 to 16
def recommend_for_5_to_16(age_recommendations):
    recommended_ids = []
    for idx, probs in enumerate(age_recommendations):
        # Check if there are enough values to unpack
        if len(probs) >= 2:
            above_18_prob, above_5_prob = probs[0], probs[1]
            # Check if the probability of being above 5 is higher than above 18
            if above_5_prob > above_18_prob:
                # Add the index to recommended list
                recommended_ids.append(idx)  # Using idx as the ID
    return recommended_ids
```

```
# Calculate class probabilities for each prediction
proba = classifier.predict_proba(X_test)
proba = classifier.predict_proba(X_test)

# Filter recommendations for ages 5 to 16 based on class probabilities
filtered_recommendations = recommend_for_5_to_16(proba)

# Print the recommended IDs for ages 5 to 16
#print("\nRecommended IDs for ages 5 to 16:", filtered_recommendations)
for id in filtered_recommendations:
  print(id)
```

```
1
8
10
11
15
17
18
24
25
40
44
45
48
50
54
57
58
60
61
63
64
65
66
69
75
79
80
81
82
87
93
94
95
97
98
99
101
116
119
131
133
137
139
141
146
151
152
154
157
158
161
165
167
172
177
184
187
190
```

190