1.

```python
import matplotlib.pyplot as plt
#####################################################
# References:
#####################################################
#https://www.youtube.com/watch?v=NxEHSAfFlK8
#https://www.youtube.com/watch?v=sgQAhG5Q7iY
#https://www.kaggle.com/code/abrahamanderson/decision-tree-entropy-information-gain

# Entropy function
def compute_entropy(subset):...
# info gain
def compute_info_gain(feature_column, target, root_entropy):...

data = pd.read_csv('balloons_2features.csv')

data['Act'] = data['Act'].map({'Stretch': 0, 'Dip': 1})
data['Age'] = data['Age'].map({'Adult': 0, 'Child': 1})
data['Inflated'] = data['Inflated'].map({'T': 1, 'F': 0})

X = np.array(data[['Act', 'Age']])
y = np.array(data['Inflated'])

#dataToPred = np.array(['Stretch', 'Adult'])
dataToPred = np.array([0, 0]).reshape(1, -1)

#####################################################
# Calculate information gain to decide where to split
#####################################################

# NEED
#   Root Entropy (of parent Node)
#   child entropy
# once achieved calculate info gain and split based on that feature

# Calculate root entropy
true_count = sum(y)
false_count = len(y) - true_count
print(f"True count: {true_count}, False count: {false_count}")
total = len(y)
p_T = true_count / total  # 8/20 = 0.4
p_F = false_count / total  # 12/20 = 0.6
root_entropy = -p_T * np.log2(p_T + 1e-10) - p_F * np.log2(p_F + 1e-10)
print(f"Root Entropy: {root_entropy:.3f}")

#get info gain for each feature
info_gains = []
for i, feature_name in enumerate(['Act', 'Age']):
    feature_col = X[:, i]
    ig = compute_info_gain(feature_col, y, root_entropy)
    info_gains.append(ig)
    print(f"Information Gain for feature '{feature_name}': {ig:.4f}")

split_class = np.argmax(info_gains)
second_split_class = 1 - split_class

# Map paths to (total count, true count)
path_counts = {}

for i in range(X.shape[0]):
    key = (X[i, split_class], X[i, second_split_class])
    if key not in path_counts:
        path_counts[key] = [0, 0]  # [total, true]
    path_counts[key][0] += 1
    if y[i] == 1:
        path_counts[key][1] += 1

#Decide T or F based on majority
results = {}
for key, (total, true_count) in path_counts.items():
    results[key] = 1 if true_count >= (total - true_count) else 0

# Predict new data point
key = (dataToPred[0, split_class], dataToPred[0, second_split_class])
pred = results.get(key, 'Unknown')  # fallback if unseen path
print('Predicted Class:', pred)

# Train Decision Tree
dt = DecisionTreeClassifier(criterion='entropy', random_state=42)
dt.fit(X, y)

# predict
print(f'Predicted Class (built in)  {dt.predict(dataToPred)}')
```

C:\WINDOWS\system32\cmd.

```
True count: 8, False count: 12
Root Entropy: 0.971
Information Gain for feature 'Act': 0.1166
Information Gain for feature 'Age': 0.1656
Predicted Class: 1
Predicted Class (built in)  [1]
Press any key to continue . . .
```