Classification (Decision Tree)

This code imports the necessary libraries for data manipulation, model training, evaluation, and visualization in a decision tree classification task. It includes pandas for data handling, scikit-learn for model-related functions, and matplotlib for plotting the decision tree.

In [1]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
import matplotlib.pyplot as plt
from sklearn import tree
```

Data Preparation

In [2]:
```
# Load the dataset
df = pd.read_csv('Starbucks satisfactory survey_modified.csv')
df = df.drop(columns=['Timestamp','Gender', 'Age'])
```

In [3]:
```
# Separate the dataset into features and target variable
x = df.drop(columns=['Continue to buy?'])
y = df['Continue to buy?']
```

In [4]:
```
# Encode the categorical variables
le_x = LabelEncoder()
x = x.apply(le_x.fit_transform)
```

This code initializes a LabelEncoder and applies it to the target variable y to convert its categorical labels into numerical values for model training.

In [5]:
```
le_y = LabelEncoder()
y = le_y.fit_transform(y)
```

Modeling

In [6]:
```
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
```

In [7]:
```
# Define the classifiers
clf_gini = DecisionTreeClassifier(max_depth=3, random_state=0)
clf_entropy = DecisionTreeClassifier(criterion='entropy', max_depth=3, random_state=0)
```

In [8]:
```
# Fit the models
clf_gini.fit(X_train, y_train)
clf_entropy.fit(X_train, y_train)
```

Out[8]:
```
DecisionTreeClassifier(criterion='entropy', max_depth=3, random_state=0)
```

Model Evaluation

In [9]:
```
# Predict using the models
y_pred_gini = clf_gini.predict(X_test)
y_pred_entropy = clf_entropy.predict(X_test)
```

In [13]:
```
# Evaluate the models
print('Accuracy score for Gini criterion:', accuracy_score(y_test, y_pred_gini))
print('Accuracy score for Entropy criterion:', accuracy_score(y_test, y_pred_entropy))

print('\nGini criterion')
print('Training set score: {:.4f}'.format(clf_gini.score(X_train, y_train)))
print('Test set score: {:.4f}'.format(clf_gini.score(X_test, y_test)))

print('\nEntropy criterion')
print('Training set score: {:.4f}'.format(clf_entropy.score(X_train, y_train)))
print('Test set score: {:.4f}'.format(clf_entropy.score(X_test, y_test)))
```

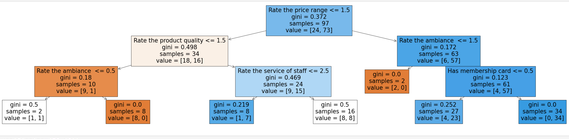
```
Accuracy score for Gini criterion: 0.84
Accuracy score for Entropy criterion: 0.84

Gini criterion
Training set score: 0.8557
Test set score: 0.8400

Entropy criterion
Training set score: 0.8557
Test set score: 0.8400
```

In [14]:
```
# Confusion matrix and classification report
cm_gini = confusion_matrix(y_test, y_pred_gini)
cm_entropy = confusion_matrix(y_test, y_pred_entropy)
```

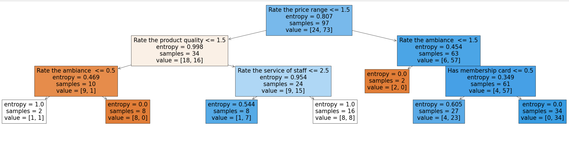
In [15]:
```
print('Confusion matrix with criterion gini index: \n', cm_gini)
print('Confusion matrix with criterion entropy: \n', cm_entropy)

print('Classification report with criterion gini index: \n', classification_report(y_test, y_pred_gini, zero_division=1))
print('Classification report with criterion entropy: \n', classification_report(y_test, y_pred_entropy, zero_division=1))
```

```
Confusion matrix with criterion gini index:
 [[ 2  2]
 [ 2 19]]
Confusion matrix with criterion entropy:
 [[ 2  2]
 [ 2 19]]
Classification report with criterion gini index:
              precision    recall  f1-score   support

           0       0.50      0.50      0.50         4
           1       0.90      0.90      0.90        21

    accuracy                           0.84        25
   macro avg       0.70      0.70      0.70        25
weighted avg       0.84      0.84      0.84        25

Classification report with criterion entropy:
              precision    recall  f1-score   support

           0       0.50      0.50      0.50         4
           1       0.90      0.90      0.90        21

    accuracy                           0.84        25
   macro avg       0.70      0.70      0.70        25
weighted avg       0.84      0.84      0.84        25
```

In [17]:
```
# Visualize the decision trees
plt.figure(figsize=(50,10))
tree.plot_tree(clf_gini, feature_names=x.columns, filled=True)
plt.show()
```



In [18]:
```
plt.figure(figsize=(50,10))
tree.plot_tree(clf_entropy, feature_names=x.columns, filled=True)
plt.show()
```



Data Preparation

In [19]:
```
# Load the dataset again for balancing
df = pd.read_csv('Starbucks satisfactory survey_modified.csv')
df = df.drop(columns=['Timestamp','Gender', 'Age'])
```

In [20]:
```
#Separate the dataset into two DataFrames based on the class label
df_majority = df[df['Continue to buy?'] == "No"]
df_minority = df[df['Continue to buy?'] == "Yes"]
```

In [21]:
```
# Randomly sample the minority class with replacement to match the majority class size
df_minority_oversampled = df_minority.sample(n=len(df_majority), replace=True)
```

In [22]:
```
# Concatenate the oversampled minority class DataFrame with the majority class DataFrame
df_balanced = pd.concat([df_majority, df_minority_oversampled], ignore_index=True)
```

In [23]:
```
# Shuffle the DataFrame
df_balanced = df_balanced.sample(frac=1).reset_index(drop=True)
```

In [24]:
```
# Separate the balanced dataset into features and target variable
x_balanced = df_balanced.drop(columns=['Continue to buy?'])
y_balanced = df_balanced['Continue to buy?']
```

In [25]:
```
# Encode the categorical variables in balanced data
x_balanced = x_balanced.apply(le_x.fit_transform)
y_balanced = le_y.fit_transform(y_balanced)
```

Modeling

In [26]:
```
# Split the balanced dataset into training and testing sets
X_train_balanced, X_test_balanced, y_train_balanced, y_test_balanced = train_test_split(x_balanced, y_balanced, test_size=0.2, random_state=42)
```

In [27]:
```
# Train the models with the balanced dataset
clf_gini.fit(X_train_balanced, y_train_balanced)
clf_entropy.fit(X_train_balanced, y_train_balanced)
```

Out[27]:
```
DecisionTreeClassifier(criterion='entropy', max_depth=3, random_state=0)
```

Model Evaluation

In [28]:
```
# Make predictions
y_pred_gini_balanced = clf_gini.predict(X_test_balanced)
y_pred_entropy_balanced = clf_entropy.predict(X_test_balanced)
```

In [29]:
```
print('Gini criterion with balanced data')
print('Training set score: {:.4f}'.format(clf_gini.score(X_train_balanced, y_train_balanced)))
print('Test set score: {:.4f}'.format(clf_gini.score(X_test_balanced, y_test_balanced)))

print('\nEntropy criterion with balanced data')
print('Training set score: {:.4f}'.format(clf_entropy.score(X_train_balanced, y_train_balanced)))
print('Test set score: {:.4f}'.format(clf_entropy.score(X_test_balanced, y_test_balanced)))
```

```
Gini criterion with balanced data
Training set score: 0.8409
Test set score: 0.8333

Entropy criterion with balanced data
Training set score: 0.8409
Test set score: 0.8333
```

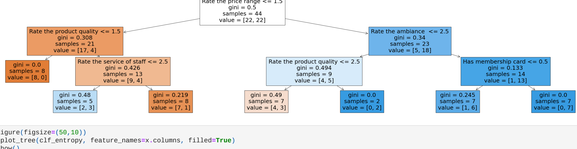
In [32]:
```
# Confusion matrix and classification report for balanced data
cm_gini_balanced = confusion_matrix(y_test_balanced, y_pred_gini_balanced)
cm_entropy_balanced = confusion_matrix(y_test_balanced, y_pred_entropy_balanced)

print('Confusion matrix with criterion gini index on balanced data: \n', cm_gini_balanced)
print('Confusion matrix with criterion entropy on balanced data: \n', cm_entropy_balanced)
```

```
Confusion matrix with criterion gini index on balanced data:
 [[5 1]
 [1 5]]
Confusion matrix with criterion entropy on balanced data:
 [[5 1]
 [1 5]]
```

In [33]:
```
print('\nClassification report with criterion gini index on balanced data: \n', classification_report(y_test_balanced, y_pred_gini_balanced, zero_division=1))
print('Classification report with criterion entropy on balanced data: \n', classification_report(y_test_balanced, y_pred_entropy_balanced, zero_division=1))
```

```
Classification report with criterion gini index on balanced data:
              precision    recall  f1-score   support

           0       0.83      0.83      0.83         6
           1       0.83      0.83      0.83         6

    accuracy                           0.83        12
   macro avg       0.83      0.83      0.83        12
weighted avg       0.83      0.83      0.83        12

Classification report with criterion entropy on balanced data:
              precision    recall  f1-score   support

           0       0.83      0.83      0.83         6
           1       0.83      0.83      0.83         6

    accuracy                           0.83        12
   macro avg       0.83      0.83      0.83        12
weighted avg       0.83      0.83      0.83        12
```

In [36]:
```
# Visualize the decision trees with balanced data
plt.figure(figsize=(50,10))
tree.plot_tree(clf_gini, feature_names=x.columns, filled=True)
plt.show()
```



In [35]:
```
plt.figure(figsize=(50,10))
tree.plot_tree(clf_entropy, feature_names=x.columns, filled=True)
plt.show()
```