In [1]:
```
import pandas as pd

import warnings
warnings.filterwarnings('ignore')

# Load dataset
df = pd.read_csv("D:\\mcdonalds.csv")
```

In [2]:
```
df.head()
```

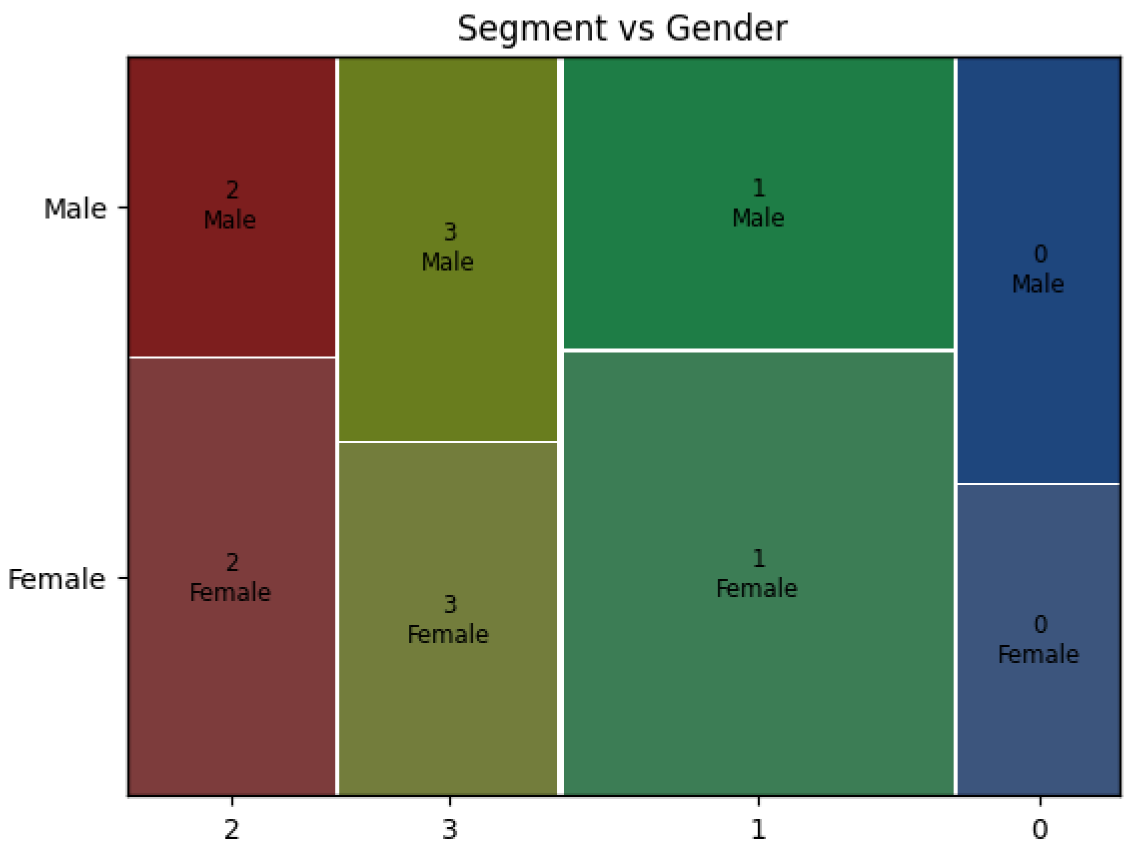
Out[2]:

	yummy	convenient	spicy	fattening	greasy	fast	cheap	tasty	expensive	healthy	disgusting	Like	Age	VisitFrequency	Gender
0	No	Yes	No	Yes	No	Yes	Yes	No	Yes	No	No	-3	61	Every three months	Female
1	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	No	No	+2	51	Every three months	Female
2	No	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	No	+1	62	Every three months	Female
3	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	No	No	Yes	+4	69	Once a week	Female
4	No	Yes	No	Yes	Yes	Yes	Yes	No	No	Yes	No	+2	49	Once a month	Male

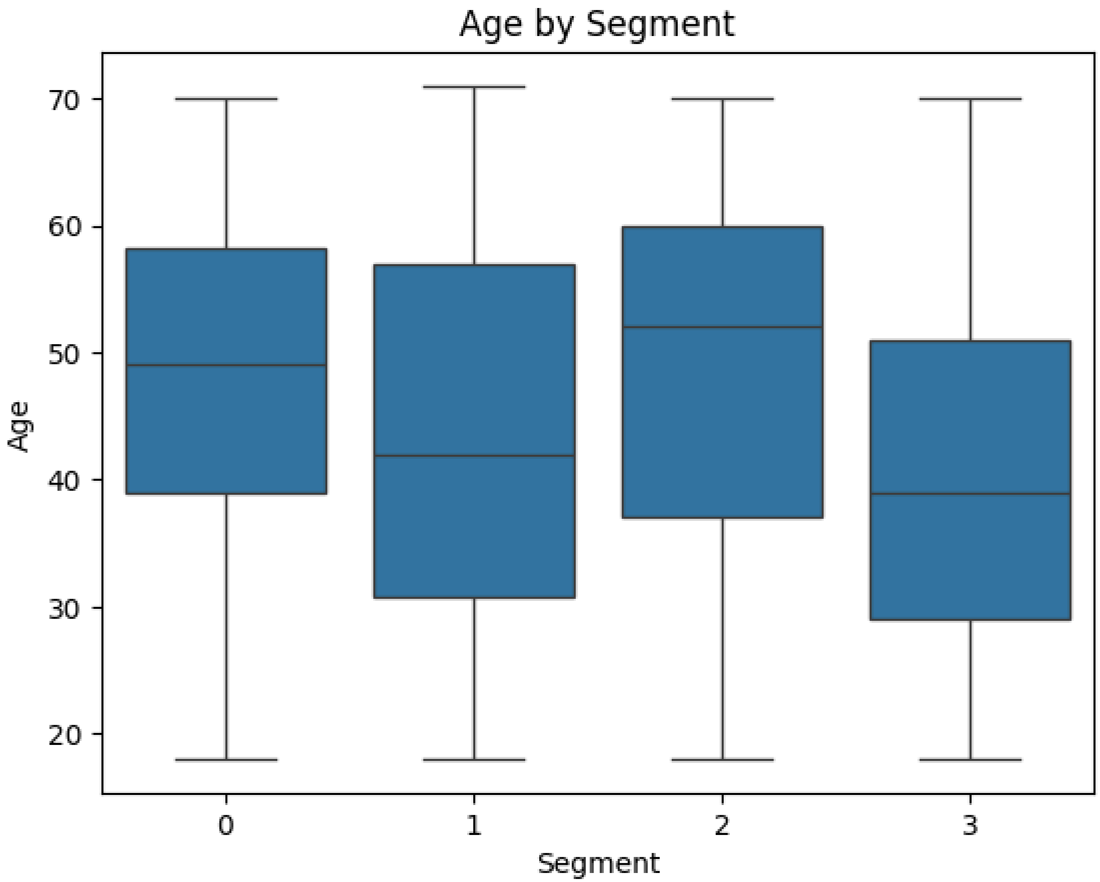
In [3]:
```
# Drop any rows with missing values
df.dropna(inplace=True)

# Ensure the 'Age' column is numeric
df['Age'] = pd.to_numeric(df['Age'], errors='coerce')

# Drop rows with invalid Age
df.dropna(subset=['Age'], inplace=True)
```

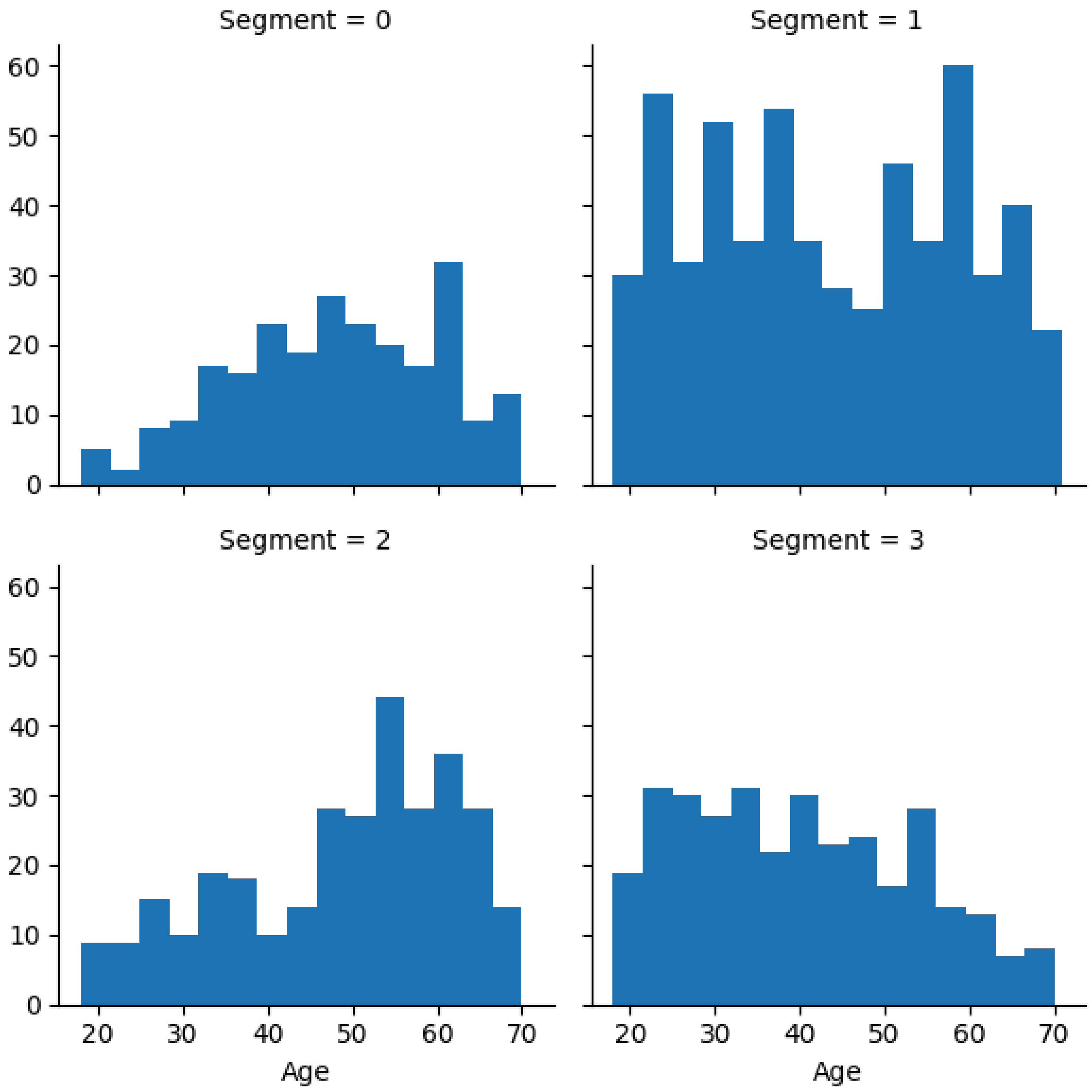
In [4]:
```
# Convert Yes/No to 1/0 for all binary columns (excluding 'Like')
binary_columns = ['yummy', 'convenient', 'spicy', 'fattening', 'greasy', 'fast', 'cheap', 'tasty', 'expensive', 'healthy', 'di
df[binary_columns] = df[binary_columns].apply(lambda x: x.map({'Yes': 1, 'No': 0}))
```

In [5]:
```
from sklearn.cluster import KMeans
from statsmodels.graphics.mosaicplot import mosaic
import matplotlib.pyplot as plt

# Apply KMeans clustering to the binary columns (excluding 'Like')
kmeans = KMeans(n_clusters=4, random_state=42, n_init=10)
df['Segment'] = kmeans.fit_predict(df[binary_columns])

# Mosaic plot for Segment vs Gender
mosaic(df, ['Segment', 'Gender'])
plt.title('Segment vs Gender')
plt.show()
```



In [6]:
```
import seaborn as sns

# Boxplot of Age by Segment
sns.boxplot(x='Segment', y='Age', data=df)
plt.title('Age by Segment')
plt.show()

# Histograms of Age by Segment
g = sns.FacetGrid(df, col="Segment", col_wrap=2)
g.map(plt.hist, "Age", bins=15)
plt.show()
```





In [7]:
```
from scipy.stats import chi2_contingency

# Cross-tabulation and chi-square test for Gender vs Segment
gender_table = pd.crosstab(df['Segment'], df['Gender'])
chi2, p, dof, expected = chi2_contingency(gender_table)
```

```
print("Chi-square p-value:", p)
```

```
Chi-square p-value: 8.697992697736528e-07
```

In [8]:
```
import statsmodels.api as sm
from statsmodels.formula.api import ols
from statsmodels.stats.multicomp import pairwise_tukeyhsd

# ANOVA to compare Age across Segments
model = ols('Age ~ C(Segment)', data=df).fit()
anova_table = sm.stats.anova_lm(model, typ=2)
print(anova_table)

# Tukey’s HSD for pairwise comparison of Age between Segments
tukey = pairwise_tukeyhsd(endog=df['Age'], groups=df['Segment'], alpha=0.05)
print(tukey)
```

```
                   sum_sq      df          F        PR(>F)
C(Segment)   16966.145572     3.0  29.616809  1.390447e-18
Residual    276689.098750  1449.0        NaN           NaN
 Multiple Comparison of Means - Tukey HSD, FWER=0.05
=====================================================
group1 group2 meandiff p-adj   lower    upper  reject
-----------------------------------------------------
     0      1  -4.6085 0.0001  -7.3363 -1.8806   True
     0      2    0.937 0.8599   -2.121   3.995  False
     0      3  -8.2242    0.0 -11.2511 -5.1973   True
     1      2   5.5455    0.0   3.0423  8.0487   True
     1      3  -3.6158  0.001  -6.0808 -1.1507   True
     2      3  -9.1612    0.0 -11.9873 -6.3352   True
-----------------------------------------------------
```

In [9]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split

# Define target variable: Predict 'Like'
X = pd.get_dummies(df[['Age', 'Gender']], drop_first=True)  # Feature matrix (Age, Gender)
y = df['Like']  # Target variable (Like)

# Split the data into training and testing sets
```

```python
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Train logistic regression model
log_reg = LogisticRegression(max_iter = 1000)
log_reg.fit(X_train, y_train)

# Print accuracy
print("Logistic regression accuracy:", log_reg.score(X_test, y_test))
```

```
Logistic regression accuracy: 0.15825688073394495
```

In [10]:

```python
from sklearn.linear_model import LogisticRegression

# Define features and target
X = pd.get_dummies(df[['Age', 'Gender']], drop_first=True)
y = df['Segment']

# Train multinomial logistic regression model
multi_model = LogisticRegression(multi_class='multinomial', solver='lbfgs', max_iter=1000)
multi_model.fit(X, y)

# Print accuracy
print("Multinomial Logistic Regression Accuracy:", multi_model.score(X, y))
```

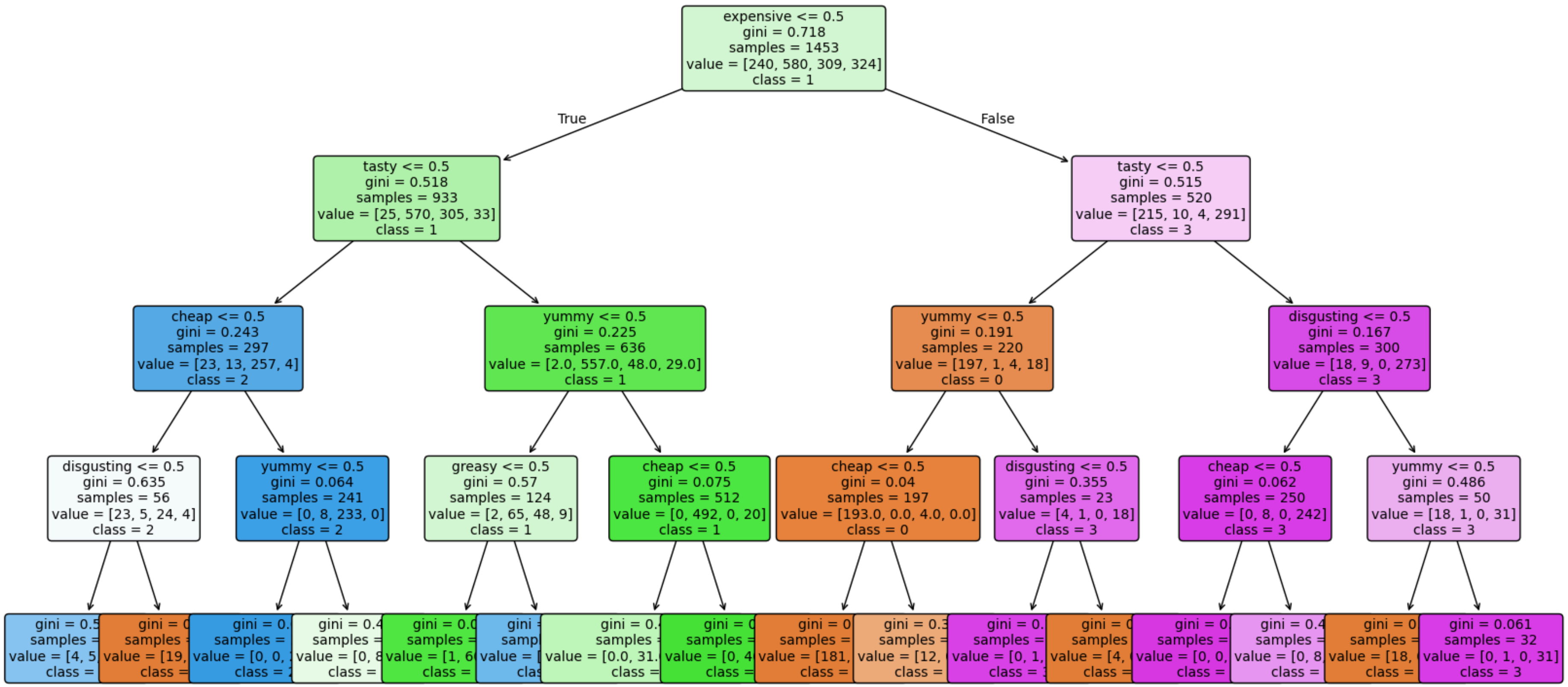
```
Multinomial Logistic Regression Accuracy: 0.39022711631108054
```

In [11]:

```python
from sklearn.tree import DecisionTreeClassifier, plot_tree

X = pd.get_dummies(df.drop(columns=['Segment', 'Like']), drop_first=True)
y = df['Segment']

# Train the model
tree_model = DecisionTreeClassifier(max_depth=4, random_state=42)
tree_model.fit(X, y)

# Visualize the decision tree
plt.figure(figsize=(20, 10))
plot_tree(tree_model,
          filled=True,
          feature_names=X.columns,
          class_names=[str(i) for i in sorted(df['Segment'].unique())],
```

```
             rounded=True,
             fontsize=10)
plt.title("Decision Tree for Customer Segmentation")
plt.show()
```

Decision Tree for Customer Segmentation




```
In [ ]:
```