7. Consider the Gini index, classification error, and entropy in a simple classification setting with two classes. Create a single plot that displays each of these quantities as a function of pˆm1. The x-axis should display pˆm1, ranging from 0 to 1, and the y-axis should display the value of the Gini index, classification error, and entropy. Hint: In a setting with two classes, pˆm1 = 1 − pˆm2. You could make this plot by hand, but it will be much easier to make in R.

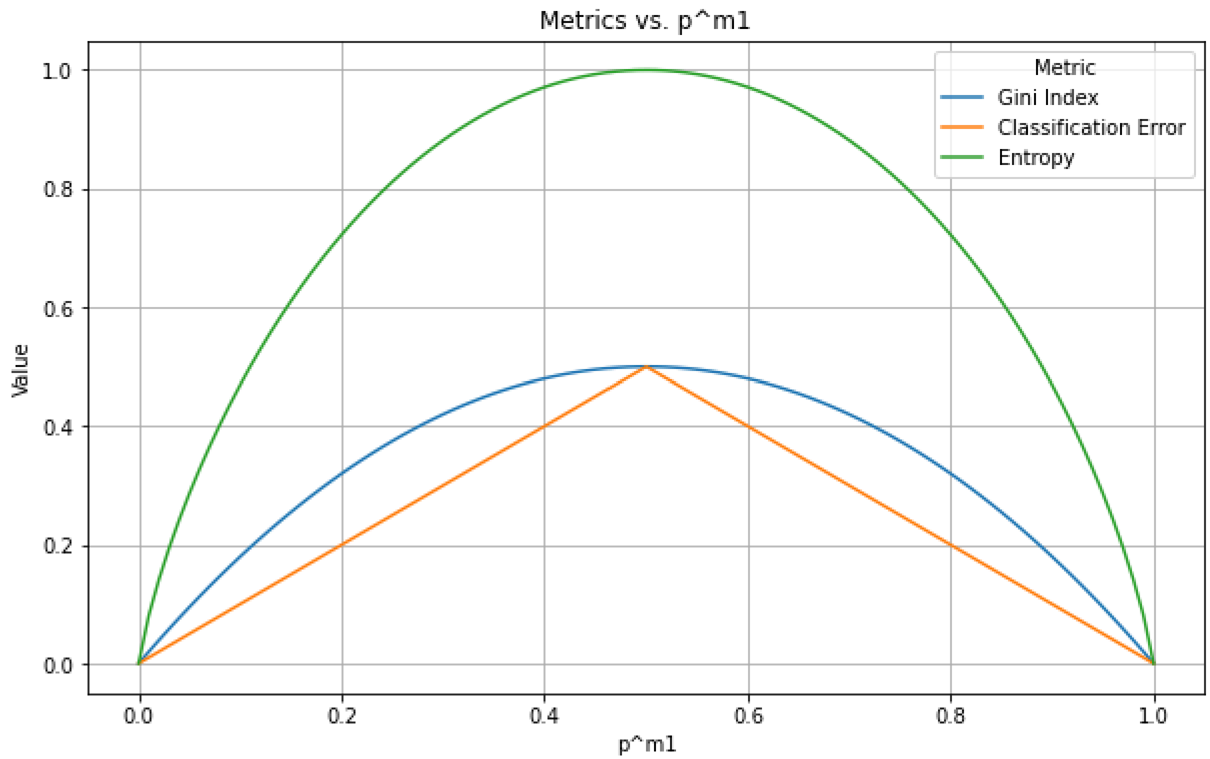
In [2]:

```
import pandas as pd
```

In [3]:

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

# Define a sequence of p^m1 values
p_m1 = np.linspace(0, 1, 101)

# Calculate corresponding p^m2 values
p_m2 = 1 - p_m1

# Calculate Gini index
gini = 2 * p_m1 * p_m2

# Calculate classification error
classification_error = np.where(p_m1 > 0.5, p_m2, p_m1)

# Calculate entropy
entropy = - (p_m1 * np.log2(p_m1 + 1e-10) + p_m2 * np.log2(p_m2 + 1e-10))


# Create a DataFrame
data = {'p^m1': p_m1,
        'Gini Index': gini,
        'Classification Error': classification_error,
        'Entropy': entropy}
df = pd.DataFrame(data)

# Melt the DataFrame
df_melted = df.melt(id_vars='p^m1', var_name='Metric', value_name='Value')

# Plot
plt.figure(figsize=(10, 6))
sns.lineplot(x='p^m1', y='Value', hue='Metric', data=df_melted)
```

```python
plt.xlabel('p^m1')
plt.ylabel('Value')
plt.title('Metrics vs. p^m1')
plt.grid(True)
plt.show()
```



8. In the lab, a classification tree was applied to the Carseats data set after converting Sales into a qualitative response variable. Now we will seek to predict Sales using regression trees and related approaches, treating the response as a quantitative

In [4]:
```python
import ISLP
from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor, plot_tree
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import confusion_matrix, mean_squared_error

carseats_df = ISLP.load_data('Carseats')
carseats_df = pd.get_dummies(carseats_df, drop_first=True)
```

In [5]:
```
carseats_df.head()
```

Out[5]:

	Sales	CompPrice	Income	Advertising	Population	Price	Age	Education	ShelveLoc_Good	ShelveLoc_Medium	Urban_Yes	US_Yes
0	9.50	138	73	11	276	120	42	17	0	0	1	1
1	11.22	111	48	16	260	83	65	10	1	0	1	1
2	10.06	113	35	10	269	80	59	12	0	1	1	1
3	7.40	117	100	4	466	97	55	14	0	1	1	1
4	4.15	141	64	3	340	128	38	13	0	0	1	0

(a) Split the data set into a training set and a test set.

In [6]:
```
y = carseats_df['Sales']
X = carseats_df.drop('Sales', axis = 1)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=27)
```

(b) Fit a regression tree to the training set. Plot the tree, and interpret the results. What test MSE do you obtain?

In [7]:
```
reg = DecisionTreeRegressor(random_state=27)
reg.fit(X_train, y_train)

y_pred = reg.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
mse
```

Out[7]:
```
4.6926151515151515
```

(c) Use cross-validation in order to determine the optimal level of tree complexity. Does pruning the tree improve the test MSE?

In [8]:
```
# Initialize lists to store results
depths = range(1, 11)
cv_scores = []

# Perform cross-validation for different tree depths
for depth in depths:
    reg = DecisionTreeRegressor(max_depth=depth, random_state=27)
```

```python
    scores = cross_val_score(reg, X, y, cv=5, scoring='neg_mean_squared_error')
    cv_scores.append(-1 * scores.mean())

# Find the optimal depth
optimal_depth = depths[cv_scores.index(min(cv_scores))]
print(f'Optimal tree depth: {optimal_depth}')

# Train a decision tree regressor with the optimal depth
reg = DecisionTreeRegressor(max_depth=optimal_depth, random_state=27)
reg.fit(X, y)

# Evaluate test MSE
y_pred = reg.predict(X_test)
mse_with_pruning = mean_squared_error(y_test, y_pred)
print(f'Test MSE with pruning: {mse_with_pruning}')
```

```
Optimal tree depth: 7
Test MSE with pruning: 0.9247495477218481
```

- yes pruning improved mse

(d) Use the bagging approach in order to analyze this data. What test MSE do you obtain? Use the feature*importance* values to determine which variables are most important.

In [9]:

```python
from sklearn.ensemble import BaggingRegressor
from sklearn.metrics import mean_squared_error
import numpy as np

# Instantiate and fit the BaggingRegressor
bagging_reg = BaggingRegressor(n_estimators=100, random_state=27)
bagging_reg.fit(X_train, y_train)

# Predict on the test set
y_pred_bagging = bagging_reg.predict(X_test)

# Compute test MSE
mse_bagging = mean_squared_error(y_test, y_pred_bagging)
print(f'Test MSE with bagging: {mse_bagging}')

# Determine feature importances
feature_importances = np.mean([tree.feature_importances_ for tree in bagging_reg.estimators_], axis=0)

# Pair feature importances with corresponding feature names
```

```python
feature_importance_dict = dict(zip(X.columns, feature_importances))

# Sort feature importances in descending order
sorted_feature_importances = sorted(feature_importance_dict.items(), key=lambda x: x[1], reverse=True)

# Display feature importances
print("\nFeature importances:")
for feature, importance in sorted_feature_importances:
    print(f'{feature}: {importance}')
```

```
Test MSE with bagging: 2.1090011781818188

Feature importances:
Price: 0.29324373679346644
ShelveLoc_Good: 0.20738940586218785
Age: 0.1172611094407813
CompPrice: 0.10688952448693337
ShelveLoc_Medium: 0.08122922258128064
Advertising: 0.06364941402468288
Income: 0.056181264863082586
Population: 0.035933303280670875
Education: 0.028065748215321297
US_Yes: 0.005094431936019011
Urban_Yes: 0.005062838515557367
```

- Price, ShelveLoc_Good, Age are the most important

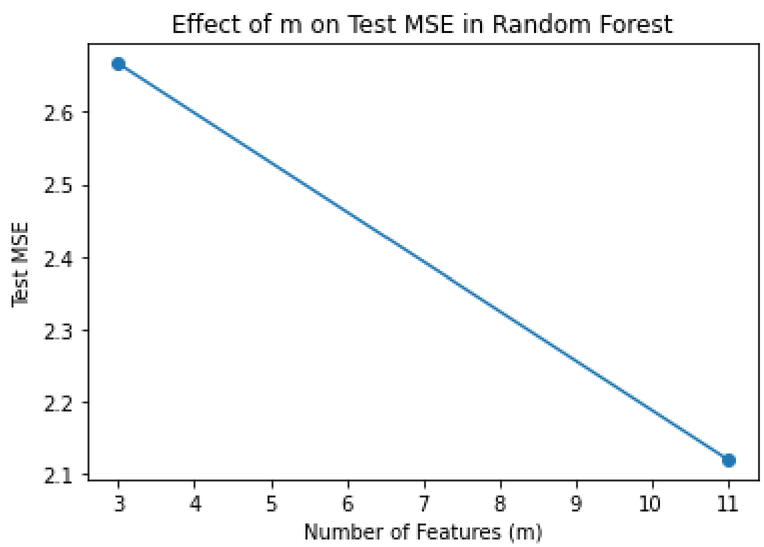
(e) Use random forests to analyze this data. What test MSE do you obtain? Use the feature*importance* values to determine which variables are most important. Describe the effect of m, the number of variables considered at each split, on the error rate obtained.

- when m is higher the tree has less randomness.
- our tree performs better with higher m

In [11]:
```python
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
import numpy as np

# Example values for m
m_values = [int(np.sqrt(X_train.shape[1])), X_train.shape[1] // 3, X_train.shape[1]]

# Dictionary to store MSE for each m
```

```python
mse_results = {}

for m in m_values:
    rf_reg = RandomForestRegressor(n_estimators=100, max_features=m, random_state=27)
    rf_reg.fit(X_train, y_train)
    y_pred_rf = rf_reg.predict(X_test)
    mse_rf = mean_squared_error(y_test, y_pred_rf)
    mse_results[m] = mse_rf
    print(f'Test MSE with m={m}: {mse_rf}')

# Optionally, plot the results to see the trend
import matplotlib.pyplot as plt
plt.plot(list(mse_results.keys()), list(mse_results.values()), marker='o')
plt.xlabel('Number of Features (m)')
plt.ylabel('Test MSE')
plt.title('Effect of m on Test MSE in Random Forest')
plt.show()
```

```
Test MSE with m=3: 2.666859729924242
Test MSE with m=3: 2.666859729924242
Test MSE with m=11: 2.119790734469697
```



In [12]:

```python
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error

# Instantiate and fit the RandomForestRegressor
rf_reg = RandomForestRegressor(n_estimators=100, random_state=27)
rf_reg.fit(X_train, y_train)
```

```python
# Predict on the test set
y_pred_rf = rf_reg.predict(X_test)

# Compute test MSE
mse_rf = mean_squared_error(y_test, y_pred_rf)
print(f'Test MSE with random forests: {mse_rf}')

# Determine feature importances
feature_importances_rf = rf_reg.feature_importances_

# Pair feature importances with corresponding feature names
feature_importance_dict_rf = dict(zip(X.columns, feature_importances_rf))

# Sort feature importances in descending order
sorted_feature_importances_rf = sorted(feature_importance_dict_rf.items(), key=lambda x: x[1], reverse=True)

# Display feature importances
print("\nFeature importances:")
for feature, importance in sorted_feature_importances_rf:
    print(f'{feature}: {importance}')
```

```
Test MSE with random forests: 2.119790734469697

Feature importances:
Price: 0.2943948293746041
ShelveLoc_Good: 0.20733343765809079
Age: 0.1177052136139735
CompPrice: 0.10721878797494723
ShelveLoc_Medium: 0.08148240247553497
Advertising: 0.06306791535350183
Income: 0.05585153800815752
Population: 0.03587210242201306
Education: 0.027466385183679342
Urban_Yes: 0.0048691117727555355
US_Yes: 0.004738276162742239
```

(f) Now analyze the data using BART, and report your results.

In [26]:
```python
type(X_test['CompPrice'].iloc[0])
```

Out[26]:
```
numpy.int64
```

In [28]:

```
import numpy as np
import pandas as pd
from ISLP.bart import BART

# Load your data
# X_train, X_test, y_train, y_test

# Initialize BART model
bart_model = BART(num_trees=50, ndraw = 15)

# Fit model
bart_model.fit(X_train, y_train)

# Predict on test data
# Convert all columns to numeric, coercing errors to NaN (adjust as necessary for your data)
# Ensure X_test is array-like
if isinstance(X_test, pd.DataFrame):
    X_test = X_test.values

y_pred = bart_model.predict(X_test)

# Calculate Mean Squared Error
mse = np.mean((y_test - y_pred) ** 2)
print(f'Test MSE with BART: {mse}')
```

```
Test MSE with BART: 1.3431335109120113
```

9. This problem involves the OJ data set which is part of the ISLP package.

(a) Create a training set containing a random sample of 800 observations, and a test set containing the remaining observations.

In [31]:

```
import pandas as pd
from sklearn.model_selection import train_test_split

# Load the dataset (assuming 'ISLP.load_data' correctly fetches the dataset)
oj = ISLP.load_data('OJ')
oj = pd.get_dummies(oj, drop_first = True)

# Check if the dataset has been loaded as a pandas DataFrame
if not isinstance(oj, pd.DataFrame):
    oj = pd.DataFrame(oj)
```

```python
# Specify the size of the dataset to ensure the training set has 800 observations
n_samples = len(oj)
train_size = 800

# If there are at least 800 samples in the dataset, proceed with the split
if n_samples >= train_size:
    # Calculate the proportion for the train size
    train_prop = train_size / n_samples

    # Split the dataset
    oj_train, oj_test = train_test_split(oj, train_size=train_prop, random_state=42)
else:
    print("The dataset contains fewer than 800 observations.")

# Display the sizes of the train and test sets to verify
print(f'Training Set Size: {len(oj_train)}')
print(f'Test Set Size: {len(oj_test)}')
```

```
Training Set Size: 800
Test Set Size: 270
```

(b) Fit a tree to the training data, with Purchase as the response and the other variables as predictors. What is the training error rate?

In [32]:

```python
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

# Assuming 'oj_train' has been defined and contains the necessary data
# Prepare the data
X_train = oj_train.drop('Purchase_MM', axis=1)  # Drop the response variable from the training data
y_train = oj_train['Purchase_MM']  # Extract the response variable

# Fit the decision tree model
tree_clf = DecisionTreeClassifier(random_state=42)
tree_clf.fit(X_train, y_train)

# Predict on the training set
y_train_pred = tree_clf.predict(X_train)

# Calculate the accuracy and the error rate
accuracy = accuracy_score(y_train, y_train_pred)
error_rate = 1 - accuracy

# Print the results
```

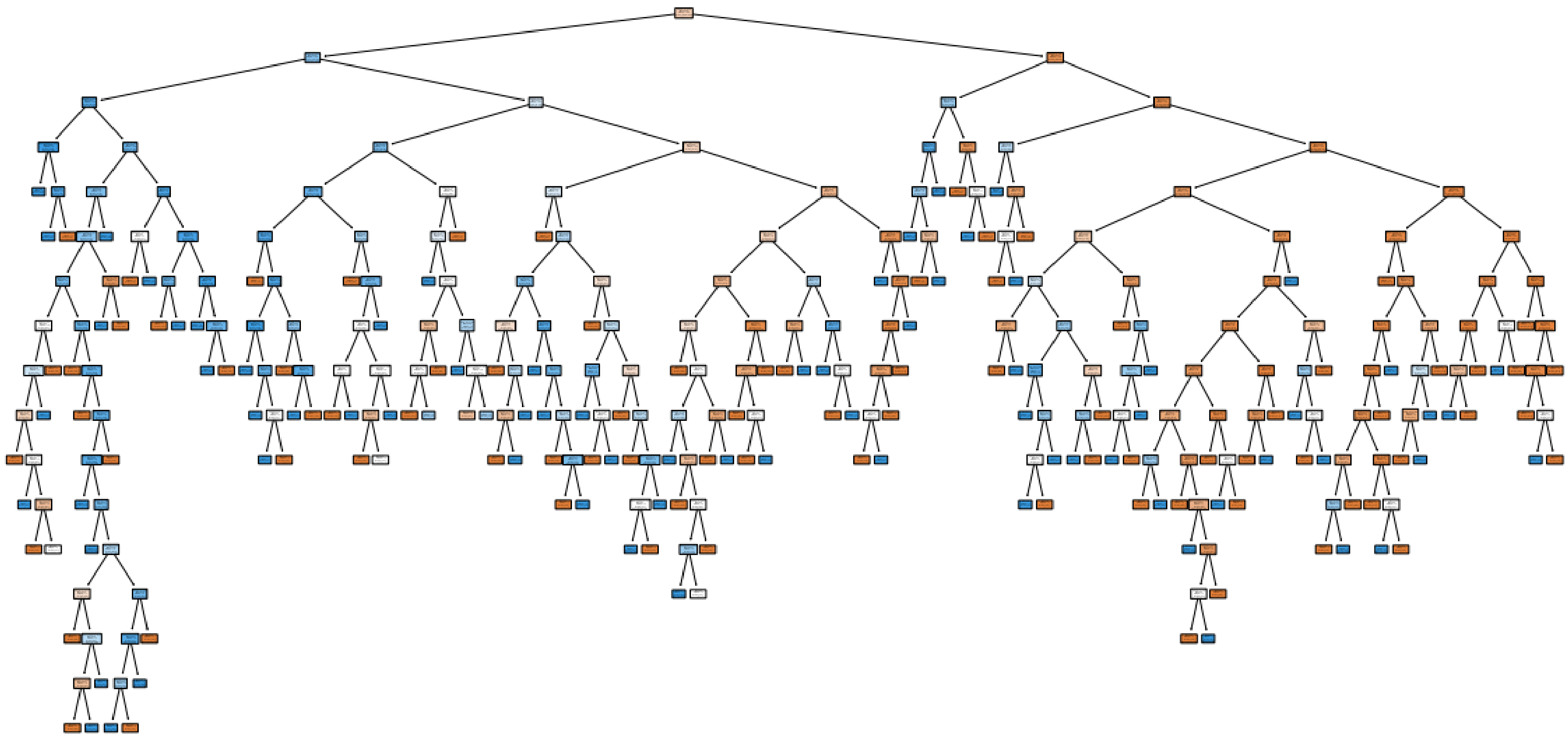
```
print(f"Training Accuracy: {accuracy:.2f}")
print(f"Training Error Rate: {error_rate:.2f}")
```

```
Training Accuracy: 0.99
Training Error Rate: 0.01
```

(c) Create a plot of the tree, and interpret the results. How many terminal nodes does the tree have?

In [33]:

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

# Assuming tree_clf has already been fitted with your data
plt.figure(figsize=(20,10))
plot_tree(tree_clf,
          filled=True,
          rounded=True,
          class_names=['Not Purchase', 'Purchase'],
          feature_names=X_train.columns)
plt.show()

# Counting the number of terminal nodes
num_terminal_nodes = sum(1 for _ in tree_clf.tree_.children_left if _ == -1)

print(f"Number of Terminal Nodes: {num_terminal_nodes}")
```



```
Number of Terminal Nodes: 163
```

(d) Use the export_tree() function to produce a text summary of the fitted tree. Pick one of the terminal nodes, and interpret the information displayed.

In [38]:
```python
from sklearn.tree import export_text

# Export the tree to a text representation
tree_text = export_text(tree_clf, feature_names=list(X_train.columns))
print(tree_text)
```

```
|--- LoyalCH <= 0.50
|   |--- LoyalCH <= 0.28
|   |   |--- LoyalCH <= 0.06
|   |   |   |--- WeekofPurchase <= 268.50
|   |   |   |   |--- class: 1
|   |   |   |--- WeekofPurchase >  268.50
|   |   |   |   |--- PriceDiff <= 0.29
|   |   |   |   |   |--- class: 1
|   |   |   |   |--- PriceDiff >  0.29
|   |   |   |   |   |--- class: 0
|   |   |--- LoyalCH >  0.06
|   |   |   |--- LoyalCH <= 0.21
|   |   |   |   |--- WeekofPurchase <= 273.00
|   |   |   |   |   |--- WeekofPurchase <= 261.00
|   |   |   |   |   |   |--- PriceDiff <= -0.13
|   |   |   |   |   |   |   |--- STORE <= 0.50
|   |   |   |   |   |   |   |   |--- WeekofPurchase <= 236.50
|   |   |   |   |   |   |   |   |   |--- LoyalCH <= 0.12
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- LoyalCH >  0.12
|   |   |   |   |   |   |   |   |   |   |--- LoyalCH <= 0.16
|   |   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |   |--- LoyalCH >  0.16
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 2
|   |   |   |   |   |   |   |   |--- WeekofPurchase >  236.50
|   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |--- STORE >  0.50
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |--- PriceDiff >  -0.13
|   |   |   |   |   |   |   |--- LoyalCH <= 0.06
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |--- LoyalCH >  0.06
|   |   |   |   |   |   |   |   |--- WeekofPurchase <= 228.00
|   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |--- WeekofPurchase >  228.00
|   |   |   |   |   |   |   |   |   |--- SalePriceMM <= 2.26
|   |   |   |   |   |   |   |   |   |   |--- WeekofPurchase <= 237.50
|   |   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |   |--- WeekofPurchase >  237.50
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 6
|   |   |   |   |   |   |   |   |   |--- SalePriceMM >  2.26
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- WeekofPurchase >  261.00
|   |   |   |   |   |   |--- PriceMM <= 2.04
|   |   |   |   |   |   |   |--- class: 1
```

```
|   |   |   |   |   |   |--- PriceMM >  2.04
|   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |--- WeekofPurchase >  273.00
|   |   |   |   |   |--- class: 1
|   |   |   |--- LoyalCH >  0.21
|   |   |   |   |--- StoreID <= 1.50
|   |   |   |   |   |--- SalePriceCH <= 1.78
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- SalePriceCH >  1.78
|   |   |   |   |   |   |--- class: 1
|   |   |   |   |--- StoreID >  1.50
|   |   |   |   |   |--- WeekofPurchase <= 229.50
|   |   |   |   |   |   |--- STORE <= 1.00
|   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |--- STORE >  1.00
|   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |--- WeekofPurchase >  229.50
|   |   |   |   |   |   |--- ListPriceDiff <= 0.31
|   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |--- ListPriceDiff >  0.31
|   |   |   |   |   |   |   |--- WeekofPurchase <= 272.50
|   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |--- WeekofPurchase >  272.50
|   |   |   |   |   |   |   |   |--- class: 0
|   |--- LoyalCH >  0.28
|   |   |--- PriceDiff <= 0.05
|   |   |   |--- SpecialCH <= 0.50
|   |   |   |   |--- SalePriceMM <= 1.74
|   |   |   |   |   |--- LoyalCH <= 0.28
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- LoyalCH >  0.28
|   |   |   |   |   |   |--- StoreID <= 3.50
|   |   |   |   |   |   |   |--- ListPriceDiff <= 0.23
|   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |--- ListPriceDiff >  0.23
|   |   |   |   |   |   |   |   |--- LoyalCH <= 0.47
|   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |--- LoyalCH >  0.47
|   |   |   |   |   |   |   |   |   |--- StoreID <= 1.50
|   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |--- StoreID >  1.50
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |--- StoreID >  3.50
|   |   |   |   |   |   |   |--- DiscMM <= 0.30
|   |   |   |   |   |   |   |   |--- class: 0
```

```
|   |   |   |   |   |   |   |--- DiscMM >  0.30
|   |   |   |   |   |   |   |   |--- WeekofPurchase <= 276.50
|   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |--- WeekofPurchase >  276.50
|   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |--- SalePriceMM >  1.74
|   |   |   |   |   |--- StoreID <= 2.50
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- StoreID >  2.50
|   |   |   |   |   |   |--- WeekofPurchase <= 229.50
|   |   |   |   |   |   |   |--- LoyalCH <= 0.44
|   |   |   |   |   |   |   |   |--- SpecialMM <= 0.50
|   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |--- SpecialMM >  0.50
|   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |--- LoyalCH >  0.44
|   |   |   |   |   |   |   |   |--- WeekofPurchase <= 228.00
|   |   |   |   |   |   |   |   |   |--- LoyalCH <= 0.49
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- LoyalCH >  0.49
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |--- WeekofPurchase >  228.00
|   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |--- WeekofPurchase >  229.50
|   |   |   |   |   |   |   |--- class: 1
|   |   |   |--- SpecialCH >  0.50
|   |   |   |   |--- STORE <= 3.50
|   |   |   |   |   |--- LoyalCH <= 0.37
|   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |--- LoyalCH >  0.37
|   |   |   |   |   |   |--- LoyalCH <= 0.47
|   |   |   |   |   |   |   |--- SalePriceMM <= 1.64
|   |   |   |   |   |   |   |   |--- LoyalCH <= 0.39
|   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |--- LoyalCH >  0.39
|   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |--- SalePriceMM >  1.64
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |--- LoyalCH >  0.47
|   |   |   |   |   |   |   |--- Store7_Yes <= 0.50
|   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |--- Store7_Yes >  0.50
|   |   |   |   |   |   |   |   |--- WeekofPurchase <= 233.50
|   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |--- WeekofPurchase >  233.50
```

```
|   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |--- STORE >  3.50
|   |   |   |   |   |--- class: 0
|   |   |--- PriceDiff >  0.05
|   |   |   |--- ListPriceDiff <= 0.26
|   |   |   |   |--- ListPriceDiff <= 0.12
|   |   |   |   |   |--- class: 0
|   |   |   |   |--- ListPriceDiff >  0.12
|   |   |   |   |   |--- LoyalCH <= 0.45
|   |   |   |   |   |   |--- ListPriceDiff <= 0.22
|   |   |   |   |   |   |   |--- WeekofPurchase <= 261.00
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |--- WeekofPurchase >  261.00
|   |   |   |   |   |   |   |   |--- STORE <= 0.50
|   |   |   |   |   |   |   |   |   |--- LoyalCH <= 0.39
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- LoyalCH >  0.39
|   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |--- STORE >  0.50
|   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |--- ListPriceDiff >  0.22
|   |   |   |   |   |   |   |--- PriceCH <= 1.81
|   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |--- PriceCH >  1.81
|   |   |   |   |   |   |   |   |--- LoyalCH <= 0.33
|   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |--- LoyalCH >  0.33
|   |   |   |   |   |   |   |   |   |--- LoyalCH <= 0.37
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- LoyalCH >  0.37
|   |   |   |   |   |   |   |   |   |   |--- WeekofPurchase <= 243.00
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |   |--- WeekofPurchase >  243.00
|   |   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |--- LoyalCH >  0.45
|   |   |   |   |   |   |--- PriceMM <= 2.04
|   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |--- PriceMM >  2.04
|   |   |   |   |   |   |   |--- PriceMM <= 2.13
|   |   |   |   |   |   |   |   |--- Store7_Yes <= 0.50
|   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |--- Store7_Yes >  0.50
|   |   |   |   |   |   |   |   |   |--- DiscCH <= 0.05
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- DiscCH >  0.05
```

```
|   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |--- PriceMM >  2.13
|   |   |   |   |   |   |   |   |--- LoyalCH <= 0.49
|   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |--- LoyalCH >  0.49
|   |   |   |   |   |   |   |   |   |--- PriceMM <= 2.21
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- PriceMM >  2.21
|   |   |   |   |   |   |   |   |   |   |--- WeekofPurchase <= 247.00
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 2
|   |   |   |   |   |   |   |   |   |   |--- WeekofPurchase >  247.00
|   |   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |--- ListPriceDiff >  0.26
|   |   |   |   |--- LoyalCH <= 0.46
|   |   |   |   |   |--- LoyalCH <= 0.41
|   |   |   |   |   |   |--- StoreID <= 3.50
|   |   |   |   |   |   |   |--- LoyalCH <= 0.31
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |--- LoyalCH >  0.31
|   |   |   |   |   |   |   |   |--- PriceDiff <= 0.33
|   |   |   |   |   |   |   |   |   |--- LoyalCH <= 0.36
|   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |--- LoyalCH >  0.36
|   |   |   |   |   |   |   |   |   |   |--- LoyalCH <= 0.40
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |   |--- LoyalCH >  0.40
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 3
|   |   |   |   |   |   |   |   |--- PriceDiff >  0.33
|   |   |   |   |   |   |   |   |   |--- LoyalCH <= 0.38
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- LoyalCH >  0.38
|   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |--- StoreID >  3.50
|   |   |   |   |   |   |   |--- WeekofPurchase <= 258.50
|   |   |   |   |   |   |   |   |--- WeekofPurchase <= 257.00
|   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |--- WeekofPurchase >  257.00
|   |   |   |   |   |   |   |   |   |--- LoyalCH <= 0.35
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- LoyalCH >  0.35
|   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |--- WeekofPurchase >  258.50
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- LoyalCH >  0.41
|   |   |   |   |   |   |--- StoreID <= 1.50
```

```
|   |   |   |   |   |   |   |--- LoyalCH <= 0.46
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |--- LoyalCH >  0.46
|   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |--- StoreID >  1.50
|   |   |   |   |   |   |   |--- LoyalCH <= 0.45
|   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |--- LoyalCH >  0.45
|   |   |   |   |   |   |   |   |--- StoreID <= 2.50
|   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |--- StoreID >  2.50
|   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |--- LoyalCH >  0.46
|   |   |   |   |   |--- WeekofPurchase <= 234.00
|   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |--- WeekofPurchase >  234.00
|   |   |   |   |   |   |--- ListPriceDiff <= 0.39
|   |   |   |   |   |   |   |--- LoyalCH <= 0.48
|   |   |   |   |   |   |   |   |--- WeekofPurchase <= 267.50
|   |   |   |   |   |   |   |   |   |--- PriceMM <= 2.13
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- PriceMM >  2.13
|   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |--- WeekofPurchase >  267.50
|   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |--- LoyalCH >  0.48
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |--- ListPriceDiff >  0.39
|   |   |   |   |   |   |   |--- class: 1
|--- LoyalCH >  0.50
|   |--- PriceDiff <= -0.39
|   |   |--- LoyalCH <= 0.76
|   |   |   |--- StoreID <= 1.50
|   |   |   |   |--- LoyalCH <= 0.63
|   |   |   |   |   |--- class: 1
|   |   |   |   |--- LoyalCH >  0.63
|   |   |   |   |   |--- LoyalCH <= 0.71
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- LoyalCH >  0.71
|   |   |   |   |   |   |--- class: 1
|   |   |   |--- StoreID >  1.50
|   |   |   |   |--- class: 1
|   |   |--- LoyalCH >  0.76
|   |   |   |--- LoyalCH <= 0.99
|   |   |   |   |--- class: 0
```

```
|   |   |   |--- LoyalCH >  0.99
|   |   |   |   |--- LoyalCH <= 1.00
|   |   |   |   |   |--- class: 1
|   |   |   |   |--- LoyalCH >  1.00
|   |   |   |   |   |--- class: 0
|   |--- PriceDiff >  -0.39
|   |   |--- PriceMM <= 1.74
|   |   |   |--- LoyalCH <= 0.71
|   |   |   |   |--- class: 1
|   |   |   |--- LoyalCH >  0.71
|   |   |   |   |--- LoyalCH <= 0.77
|   |   |   |   |   |--- StoreID <= 4.50
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- StoreID >  4.50
|   |   |   |   |   |   |--- class: 1
|   |   |   |   |--- LoyalCH >  0.77
|   |   |   |   |   |--- class: 0
|   |   |--- PriceMM >  1.74
|   |   |   |--- LoyalCH <= 0.71
|   |   |   |   |--- PriceDiff <= 0.09
|   |   |   |   |   |--- ListPriceDiff <= 0.23
|   |   |   |   |   |   |--- PriceDiff <= -0.25
|   |   |   |   |   |   |   |--- WeekofPurchase <= 272.50
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |--- WeekofPurchase >  272.50
|   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |--- PriceDiff >  -0.25
|   |   |   |   |   |   |   |--- LoyalCH <= 0.63
|   |   |   |   |   |   |   |   |--- Store7_Yes <= 0.50
|   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |--- Store7_Yes >  0.50
|   |   |   |   |   |   |   |   |   |--- PriceDiff <= 0.05
|   |   |   |   |   |   |   |   |   |   |--- LoyalCH <= 0.56
|   |   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |   |--- LoyalCH >  0.56
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- PriceDiff >  0.05
|   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |--- LoyalCH >  0.63
|   |   |   |   |   |   |   |   |--- SalePriceMM <= 1.74
|   |   |   |   |   |   |   |   |   |--- LoyalCH <= 0.69
|   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |--- LoyalCH >  0.69
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |--- SalePriceMM >  1.74
```

```
|   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- ListPriceDiff >  0.23
|   |   |   |   |   |   |--- LoyalCH <= 0.66
|   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |--- LoyalCH >  0.66
|   |   |   |   |   |   |   |--- SpecialCH <= 0.50
|   |   |   |   |   |   |   |   |--- WeekofPurchase <= 263.00
|   |   |   |   |   |   |   |   |   |--- LoyalCH <= 0.67
|   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |--- LoyalCH >  0.67
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |--- WeekofPurchase >  263.00
|   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |--- SpecialCH >  0.50
|   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |--- PriceDiff >  0.09
|   |   |   |   |   |--- LoyalCH <= 0.69
|   |   |   |   |   |   |--- SalePriceMM <= 2.26
|   |   |   |   |   |   |   |--- PriceMM <= 2.11
|   |   |   |   |   |   |   |   |--- ListPriceDiff <= 0.27
|   |   |   |   |   |   |   |   |   |--- WeekofPurchase <= 237.50
|   |   |   |   |   |   |   |   |   |   |--- LoyalCH <= 0.54
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |   |--- LoyalCH >  0.54
|   |   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |--- WeekofPurchase >  237.50
|   |   |   |   |   |   |   |   |   |   |--- StoreID <= 2.50
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |   |--- StoreID >  2.50
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 4
|   |   |   |   |   |   |   |   |--- ListPriceDiff >  0.27
|   |   |   |   |   |   |   |   |   |--- LoyalCH <= 0.68
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- LoyalCH >  0.68
|   |   |   |   |   |   |   |   |   |   |--- SpecialMM <= 0.50
|   |   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |   |--- SpecialMM >  0.50
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |--- PriceMM >  2.11
|   |   |   |   |   |   |   |   |--- LoyalCH <= 0.54
|   |   |   |   |   |   |   |   |   |--- STORE <= 1.00
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- STORE >  1.00
|   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |--- LoyalCH >  0.54
```

```
|   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |--- SalePriceMM >  2.26
|   |   |   |   |   |   |   |--- WeekofPurchase <= 255.50
|   |   |   |   |   |   |   |   |--- LoyalCH <= 0.64
|   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |--- LoyalCH >  0.64
|   |   |   |   |   |   |   |   |   |--- LoyalCH <= 0.68
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- LoyalCH >  0.68
|   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |--- WeekofPurchase >  255.50
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- LoyalCH >  0.69
|   |   |   |   |   |   |--- class: 1
|   |   |   |--- LoyalCH >  0.71
|   |   |   |   |--- WeekofPurchase <= 257.50
|   |   |   |   |   |--- WeekofPurchase <= 237.50
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- WeekofPurchase >  237.50
|   |   |   |   |   |   |--- LoyalCH <= 0.92
|   |   |   |   |   |   |   |--- SpecialCH <= 0.50
|   |   |   |   |   |   |   |   |--- LoyalCH <= 0.80
|   |   |   |   |   |   |   |   |   |--- STORE <= 0.50
|   |   |   |   |   |   |   |   |   |   |--- LoyalCH <= 0.76
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 2
|   |   |   |   |   |   |   |   |   |   |--- LoyalCH >  0.76
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- STORE >  0.50
|   |   |   |   |   |   |   |   |   |   |--- LoyalCH <= 0.79
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |   |--- LoyalCH >  0.79
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 2
|   |   |   |   |   |   |   |   |--- LoyalCH >  0.80
|   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |--- SpecialCH >  0.50
|   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |--- LoyalCH >  0.92
|   |   |   |   |   |   |   |--- LoyalCH <= 0.96
|   |   |   |   |   |   |   |   |--- ListPriceDiff <= 0.23
|   |   |   |   |   |   |   |   |   |--- Store7_Yes <= 0.50
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- Store7_Yes >  0.50
|   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |--- ListPriceDiff >  0.23
|   |   |   |   |   |   |   |   |   |--- class: 1
```

```
|   |   |   |   |   |   |   |--- LoyalCH >  0.96
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |--- WeekofPurchase >  257.50
|   |   |   |   |   |--- StoreID <= 2.50
|   |   |   |   |   |   |--- LoyalCH <= 0.98
|   |   |   |   |   |   |   |--- ListPriceDiff <= 0.13
|   |   |   |   |   |   |   |   |--- PriceDiff <= -0.02
|   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |--- PriceDiff >  -0.02
|   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |--- ListPriceDiff >  0.13
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |--- LoyalCH >  0.98
|   |   |   |   |   |   |   |--- LoyalCH <= 0.99
|   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |--- LoyalCH >  0.99
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- StoreID >  2.50
|   |   |   |   |   |   |--- STORE <= 3.50
|   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |--- STORE >  3.50
|   |   |   |   |   |   |   |--- WeekofPurchase <= 265.50
|   |   |   |   |   |   |   |   |--- WeekofPurchase <= 264.50
|   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |--- WeekofPurchase >  264.50
|   |   |   |   |   |   |   |   |   |--- LoyalCH <= 0.97
|   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |--- LoyalCH >  0.97
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |--- WeekofPurchase >  265.50
|   |   |   |   |   |   |   |   |--- class: 0
```

- First Condition: LoyalCH <= 0.50 This condition splits on the LoyalCH feature, which may represent some form of loyalty score towards a brand (CH could simply a specific brand name). The condition checks if the loyalty score is 0.50 or less. Second Condition: PriceDiff > 0.29
- Following the path where the loyalty score is low (≤ 0.50), the next condition focuses on PriceDiff, possibly representing the price difference between competing products. The decision tree checks if this price difference is greater than 0.29.
- Class Prediction: Class: 0 The prediction at this terminal node is '0', which might indicate a non-purchase decision, suggesting that with lower loyalty and a significant price difference favoring the competitor, the likelihood of purchasing the CH brand is low.

- |--- LoyalCH <= 0.50
- | |--- PriceDiff > 0.29
- | | |--- class: 0

(e) Predict the response on the test data, and produce a confusion matrix comparing the test labels to the predicted test labels. What is the test error rate?

In [40]:
```
X_test
```

Out[40]:

	CompPrice	Income	Advertising	Population	Price	Age	Education	ShelveLoc_Bad	ShelveLoc_Good	ShelveLoc_Medium	Urban_No	Ur
149	121	120	13	140	87	56	11	0	0	1	0	
21	134	29	12	239	109	62	18	0	1	0	1	
342	137	102	13	422	118	71	10	0	0	1	1	
29	104	99	15	226	102	58	17	1	0	0	0	
338	112	24	0	164	101	45	11	0	0	1	0	
...	...	...	...	...	...	...	...	...	...	...	...	
347	96	39	0	161	112	27	14	0	1	0	1	
330	122	59	0	501	112	32	14	1	0	0	1	
22	128	46	6	497	138	42	13	0	0	1	0	
49	157	93	0	51	149	32	17	0	1	0	0	
210	125	41	2	357	123	47	14	1	0	0	1	

132 rows × 14 columns

In [42]:
```
# Predict on the test set
X_test = oj_test.drop('Purchase_MM', axis=1)  # Ensure 'Purchase' is the name of the response variable
y_test = oj_test['Purchase_MM']
y_pred_test = tree_clf.predict(X_test)

# Generate the confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred_test)
```

```python
print("Confusion Matrix:")
print(conf_matrix)

# Calculate the test error rate
test_accuracy = accuracy_score(y_test, y_pred_test)
test_error_rate = 1 - test_accuracy

# Print the test error rate
print(f"Test Error Rate: {test_error_rate:.2f}")
```

```
Confusion Matrix:
[[122  37]
 [ 38  73]]
Test Error Rate: 0.28
```

(f) Use cross-validation on the training set in order to determine the optimal tree size.

In [43]:

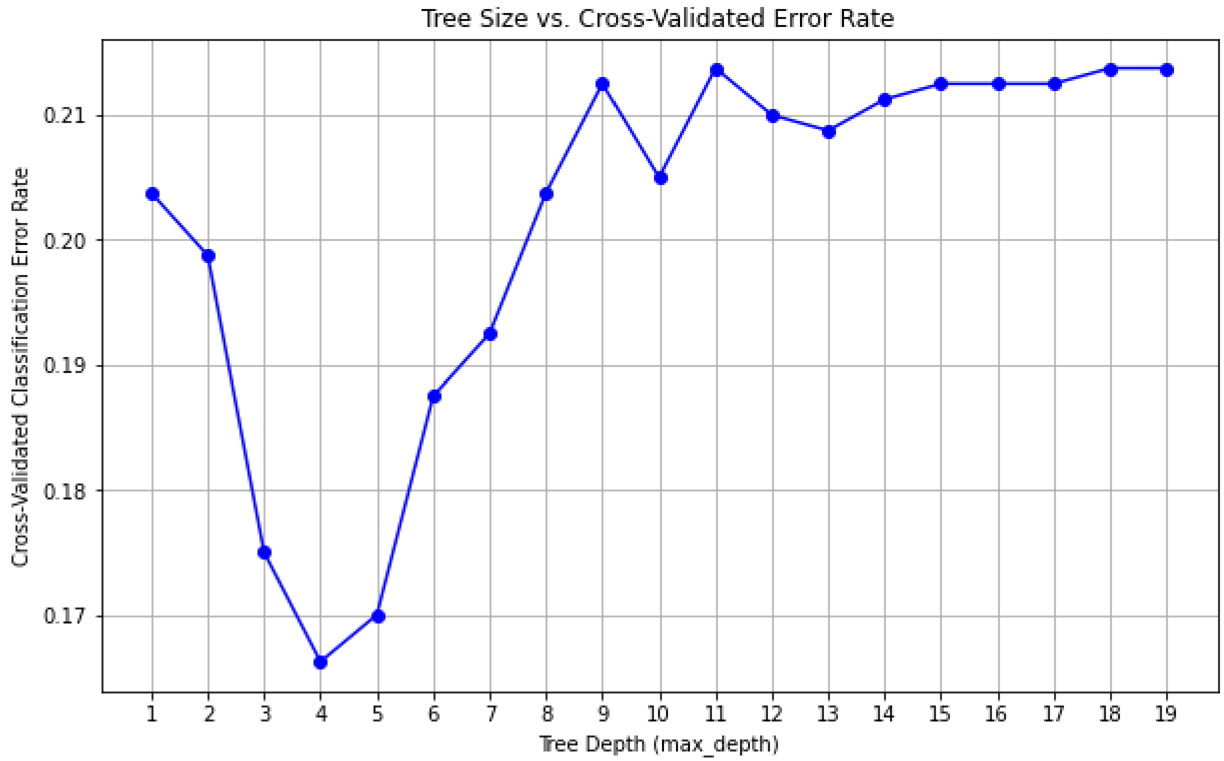
```python
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeClassifier

# Set up the parameter grid to tune 'max_depth' for the decision tree
param_grid = {
    'max_depth': range(1, 20)  # You can adjust the range based on prior knowledge or preliminary results
}

# Initialize the classifier
tree_clf = DecisionTreeClassifier(random_state=42)

# Set up GridSearchCV
grid_search = GridSearchCV(estimator=tree_clf, param_grid=param_grid, cv=5, scoring='accuracy')

# Fit grid search to the data
grid_search.fit(X_train, y_train)

# Best parameter and best score
print("Best Parameters:", grid_search.best_params_)
print("Best Cross-validation Score: {:.2f}".format(grid_search.best_score_))

# Optionally, you can also check the performance over each combination of parameters
cv_results = grid_search.cv_results_
for mean_score, params in zip(cv_results['mean_test_score'], cv_results['params']):
    print(params, '->', mean_score)
```

```
Best Parameters: {'max_depth': 4}
Best Cross-validation Score: 0.83
{'max_depth': 1} -> 0.7962499999999999
{'max_depth': 2} -> 0.8012499999999999
{'max_depth': 3} -> 0.825
{'max_depth': 4} -> 0.83375
{'max_depth': 5} -> 0.8300000000000001
{'max_depth': 6} -> 0.8125
{'max_depth': 7} -> 0.8074999999999999
{'max_depth': 8} -> 0.79625
{'max_depth': 9} -> 0.7875
{'max_depth': 10} -> 0.7950000000000002
{'max_depth': 11} -> 0.78625
{'max_depth': 12} -> 0.79
{'max_depth': 13} -> 0.79125
{'max_depth': 14} -> 0.7887500000000001
{'max_depth': 15} -> 0.7875
{'max_depth': 16} -> 0.7875
{'max_depth': 17} -> 0.7875
{'max_depth': 18} -> 0.7862500000000001
{'max_depth': 19} -> 0.7862500000000001
```

(g) Produce a plot with tree size on the x-axis and cross-validated classification error rate on the y-axis.

In [44]:

```python
import matplotlib.pyplot as plt

# Define the range of depths tested
depths = range(1, 20)  # Adjust if the range in GridSearchCV is different

# Calculate error rates as 1 - mean_test_score
error_rates = [1 - score for score in cv_results['mean_test_score']]

# Plotting the error rates
plt.figure(figsize=(10, 6))
plt.plot(depths, error_rates, marker='o', linestyle='-', color='b')
plt.title('Tree Size vs. Cross-Validated Error Rate')
plt.xlabel('Tree Depth (max_depth)')
plt.ylabel('Cross-Validated Classification Error Rate')
plt.grid(True)
plt.xticks(depths)  # Ensure all depth values appear as ticks
plt.show()
```



(h) Which tree size corresponds to the lowest cross-validated classification error rate?

- 4

(i) Produce a pruned tree corresponding to the optimal tree size obtained using cross-validation. If cross-validation does not lead to selection of a pruned tree, then create a pruned tree with five terminal nodes.

In [52]:
```
from sklearn.tree import DecisionTreeClassifier

# Instantiate the classifier with the optimal depth
optimal_depth_tree = DecisionTreeClassifier(max_depth=4, random_state=42)
optimal_depth_tree.fit(X_train, y_train)
```

Out[52]:
```
DecisionTreeClassifier
DecisionTreeClassifier(max_depth=4, random_state=42)
```

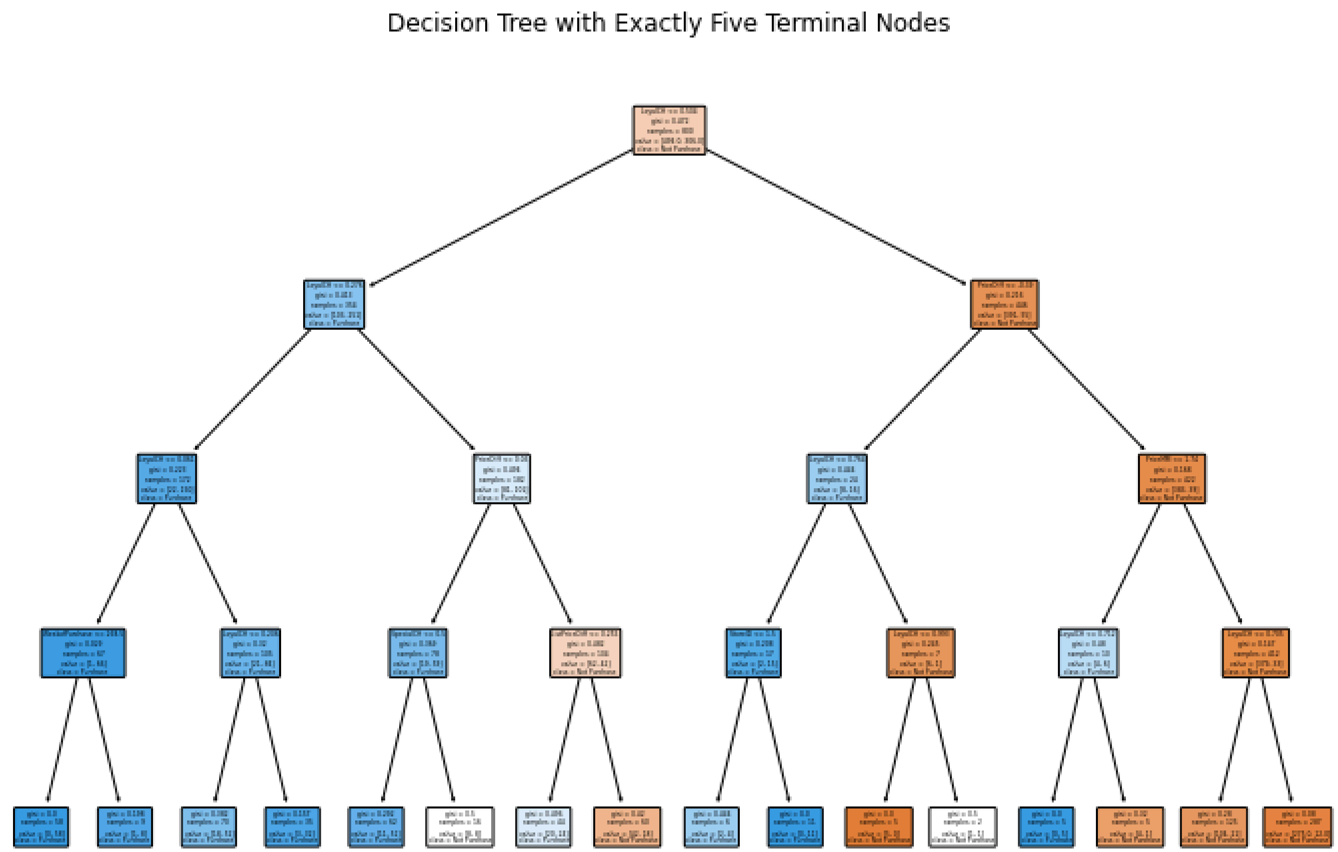
In [53]:

```python
# Count terminal nodes (leaves)
n_terminal_nodes_optimal = optimal_depth_tree.get_n_leaves()

print(f"Number of terminal nodes in the optimal tree: {n_terminal_nodes_optimal}")
```

```
Number of terminal nodes in the optimal tree: 16
```

In [54]:

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

# Plot the adjusted tree
plt.figure(figsize=(12, 8))
plot_tree(optimal_depth_tree, filled=True, feature_names=X_train.columns, class_names=['Not Purchase', 'Purchase'], rou
plt.title('Decision Tree with Exactly Five Terminal Nodes')
plt.show()
```



(j) Compare the training error rates between the pruned and unpruned trees. Which is higher?

In [56]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

# Unpruned Tree (Using optimal max_depth found from cross-validation)
unpruned_tree = DecisionTreeClassifier(max_depth=4, random_state=42)
unpruned_tree.fit(X_train, y_train)

# Pruned Tree (Using max_leaf_nodes to limit to five leaves)
pruned_tree = DecisionTreeClassifier(max_leaf_nodes=5, random_state=42)
pruned_tree.fit(X_train, y_train)

# Predictions from unpruned tree
y_pred_train_unpruned = unpruned_tree.predict(X_train)

# Predictions from pruned tree
y_pred_train_pruned = pruned_tree.predict(X_train)

# Calculate accuracy
accuracy_unpruned = accuracy_score(y_train, y_pred_train_unpruned)
accuracy_pruned = accuracy_score(y_train, y_pred_train_pruned)

# Calculate error rates
error_rate_unpruned = 1 - accuracy_unpruned
error_rate_pruned = 1 - accuracy_pruned

# Print error rates
print(f"Training Error Rate for Unpruned Tree: {error_rate_unpruned:.4f}")
print(f"Training Error Rate for Pruned Tree: {error_rate_pruned:.4f}")

if error_rate_unpruned < error_rate_pruned:
    print("The unpruned tree has a lower training error rate.")
elif error_rate_unpruned > error_rate_pruned:
    print("The pruned tree has a lower training error rate.")
else:
    print("Both trees have the same training error rate.")
```

```
Training Error Rate for Unpruned Tree: 0.1450
Training Error Rate for Pruned Tree: 0.1625
The unpruned tree has a lower training error rate.
```

(k) Compare the test error r

In [57]:

```python
# Predictions from unpruned tree on the test data
y_pred_test_unpruned = unpruned_tree.predict(X_test)

# Predictions from pruned tree on the test data
y_pred_test_pruned = pruned_tree.predict(X_test)

# Calculate accuracy for both models on the test set
accuracy_test_unpruned = accuracy_score(y_test, y_pred_test_unpruned)
accuracy_test_pruned = accuracy_score(y_test, y_pred_test_pruned)

# Calculate error rates
error_rate_test_unpruned = 1 - accuracy_test_unpruned
error_rate_test_pruned = 1 - accuracy_test_pruned

# Print test error rates
print(f"Test Error Rate for Unpruned Tree: {error_rate_test_unpruned:.4f}")
print(f"Test Error Rate for Pruned Tree: {error_rate_test_pruned:.4f}")

if error_rate_test_unpruned < error_rate_test_pruned:
    print("The unpruned tree has a lower test error rate.")
elif error_rate_test_unpruned > error_rate_test_pruned:
    print("The pruned tree has a lower test error rate.")
else:
    print("Both trees have the same test error rate.")
```

```
Test Error Rate for Unpruned Tree: 0.2407
Test Error Rate for Pruned Tree: 0.1926
The pruned tree has a lower test error rate.
```