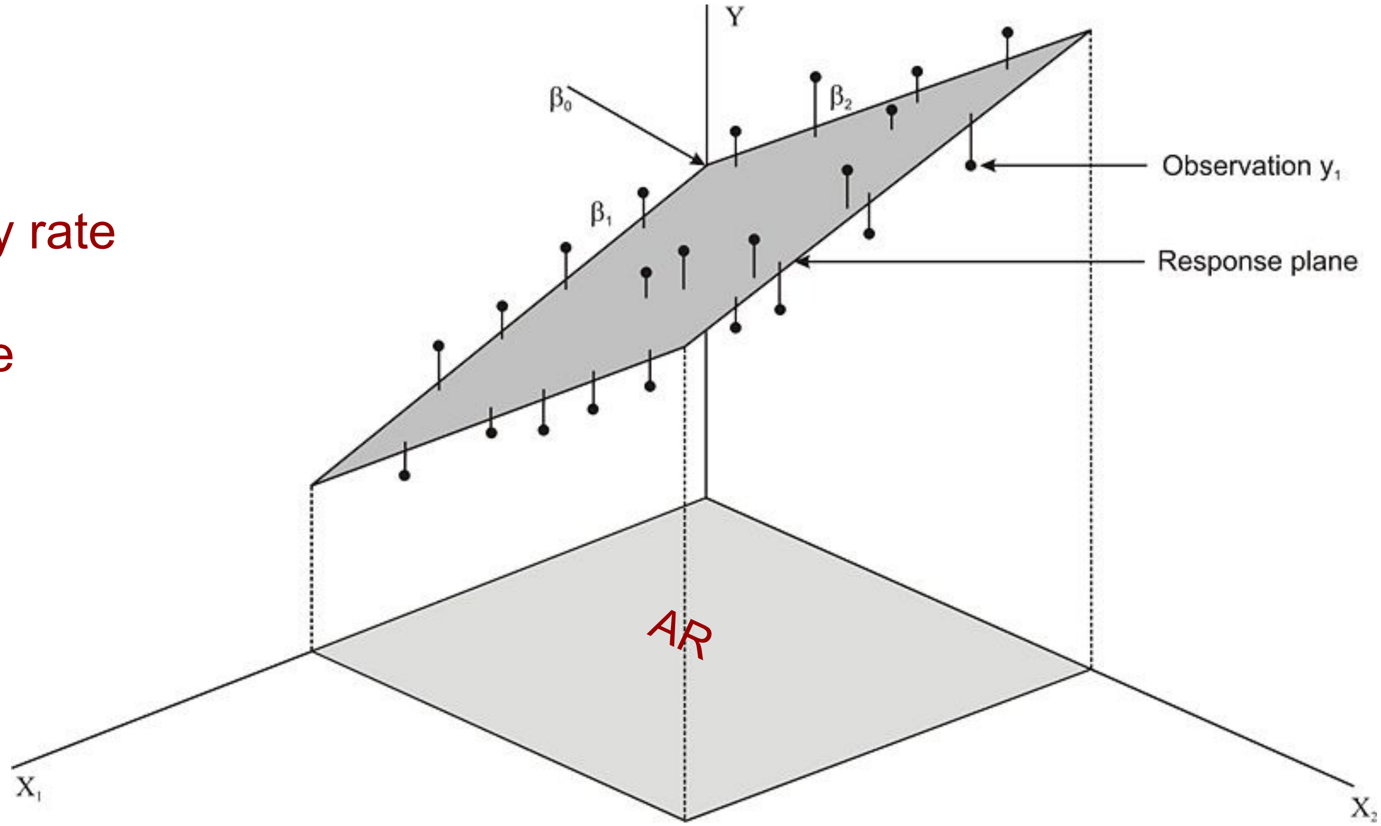
Classification Trees

Classification Trees – Procedure

- Can be used with classification problems with 2 or more categories
- The tree is grown (and the splits are chosen) the same way as with a regression tree
- For a regression tree the performance measure is SSE
- For a classification tree the performance measure is the *accuracy rate (AR)*
- There are other performance measures

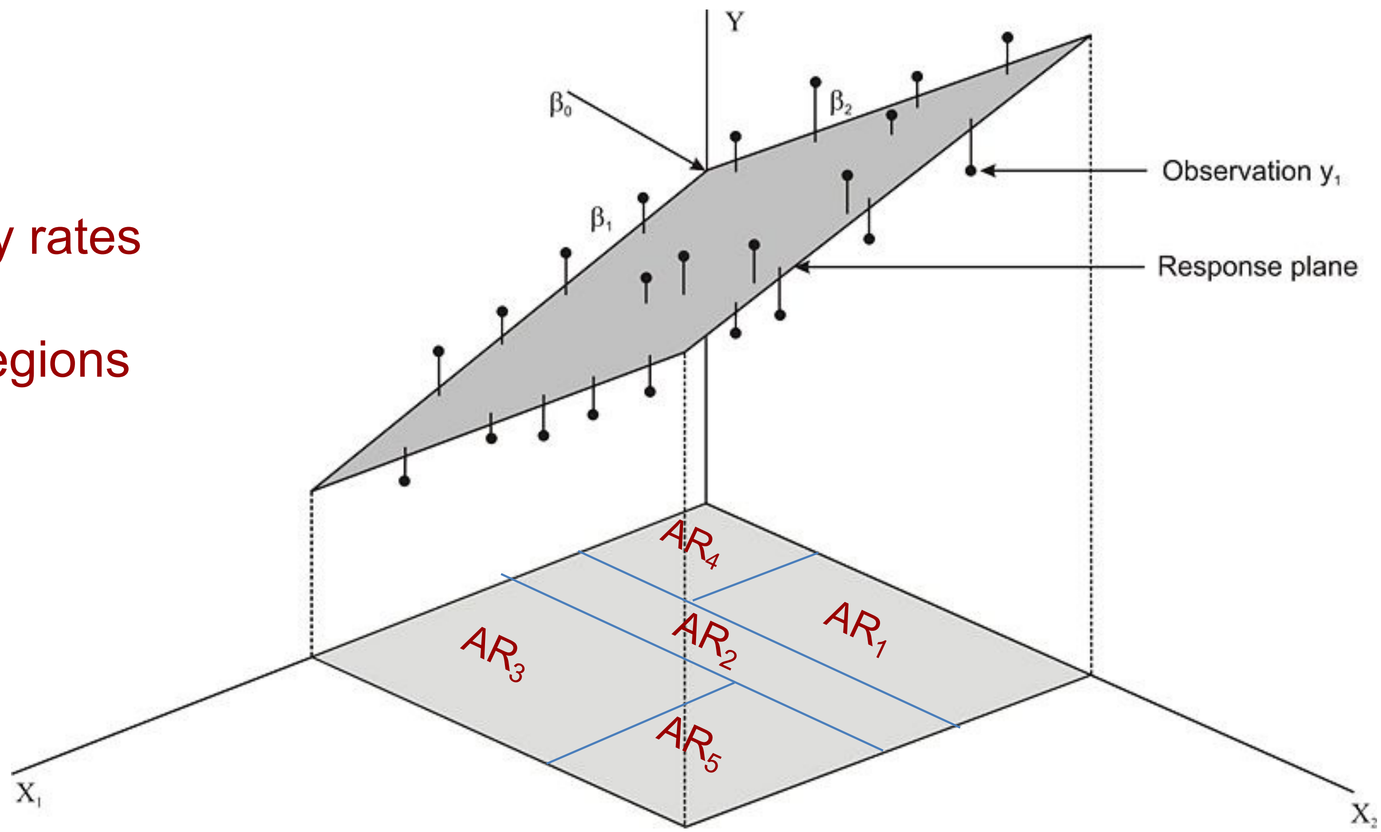
Classification Trees – Procedure

Accuracy rate
AR
for all the
dataset



Classification Trees – Procedure

Accuracy rates
AR
for the regions



Classification Trees – Procedure

- Split the predictors space into regions
- A region is pure if all observations belong to the same category
- For each region, the prediction is equal to the most common category in the region

Classification Trees – Example

- Response with K = 3 categories (classes)
- p_{mi} proportion of observations from category i in region m
- For region 4

$$\hat{p}_{41} = 10\% \quad \text{members from class 1}$$
$$\hat{p}_{42} = 20\% \quad \text{class 2}$$
$$\hat{p}_{43} = 70\% \quad \text{class 3}$$

- if a new observation falls in region 4, prediction is $\hat{y}_4 = 3$
- error rate for region 4 is $e_4 = 0.3$

Classification Trees

- Overall error rate on all T regions is $$E = \sum_{j=1}^{T} \frac{n_j}{n} e_j$$
- Other measures of performance are
    - Gini index
    - Cross entropy
- They are called measures of impurity
- If the region is pure, all are equal to zero

Classification Trees – Measures of Impurity

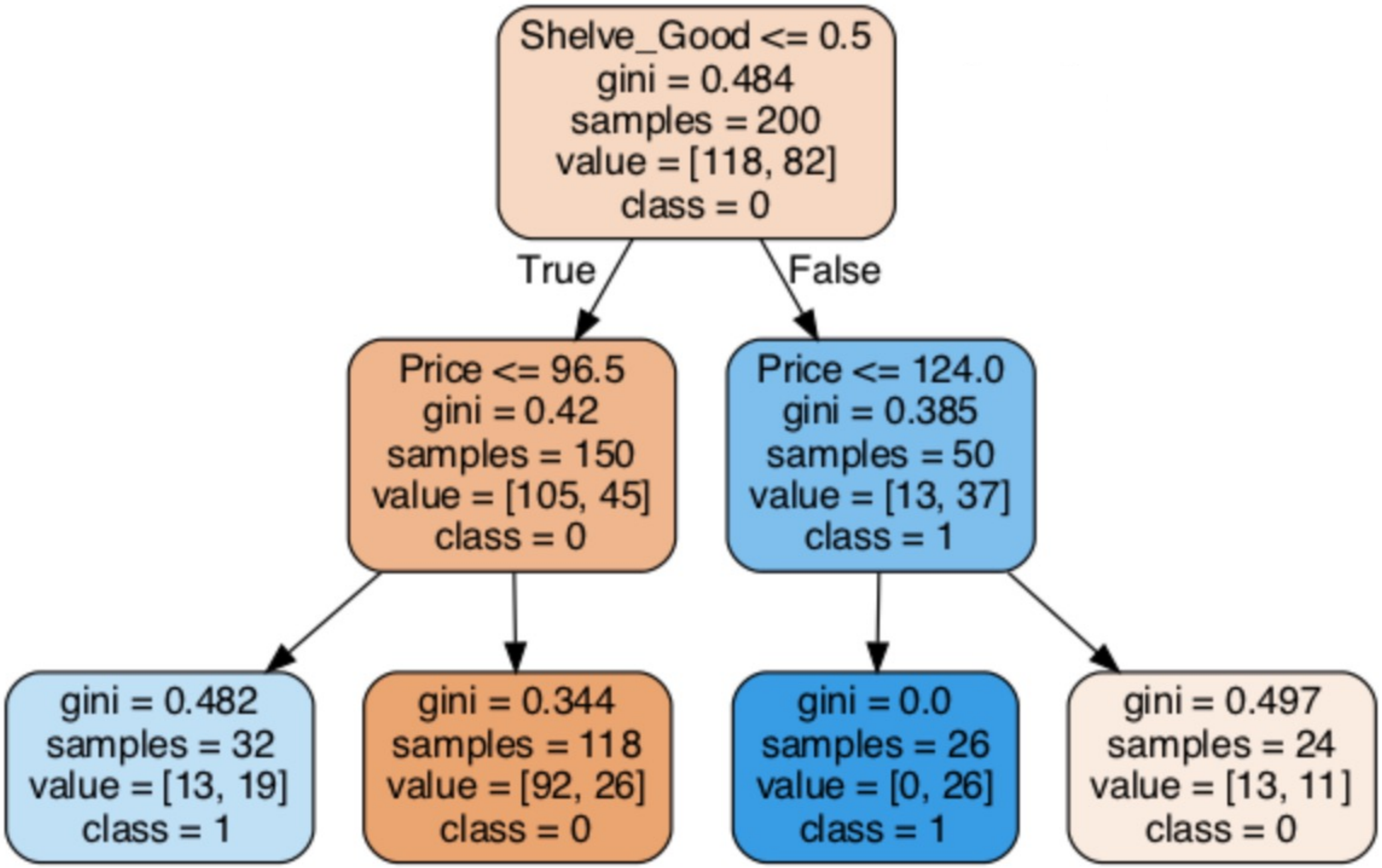
– E: classification error rate $$E = \sum_{m=1}^{T} \frac{n_m}{n} e_m$$

– G: Gini Index $$G = \sum_{m=1}^{T} \left[1 - \sum_{i=1}^{K} p_{im}^2 \right]$$

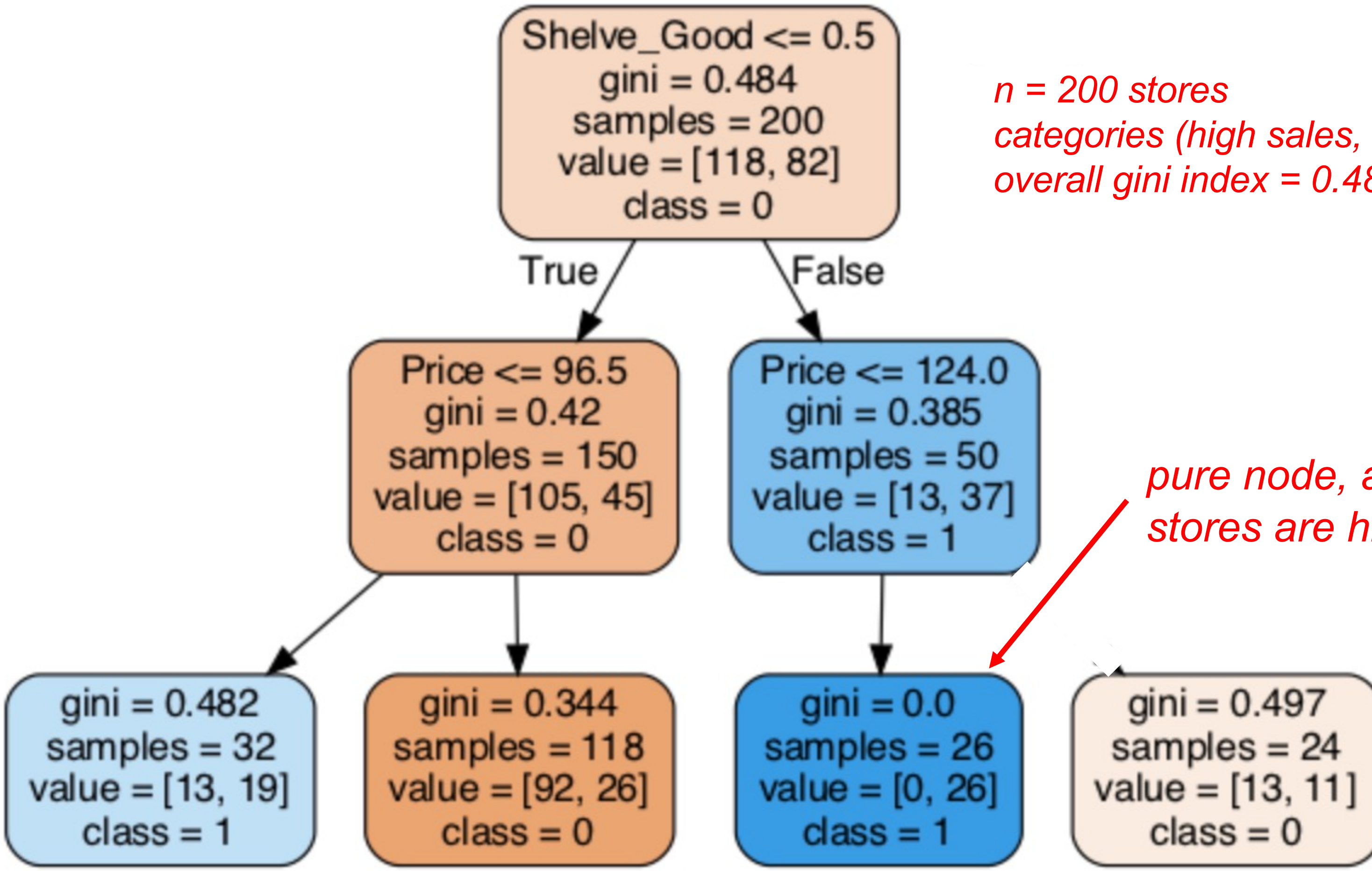
– D: Cross entropy $$D = \sum_{m=1}^{T} \left[\sum_{i=1}^{K} p_{im} \log_2(p_{im}) \right]$$

where p_{im} is the proportion of observations from category i in region m

Classification Trees – Store sales Example



Classification Trees – Store sales Example

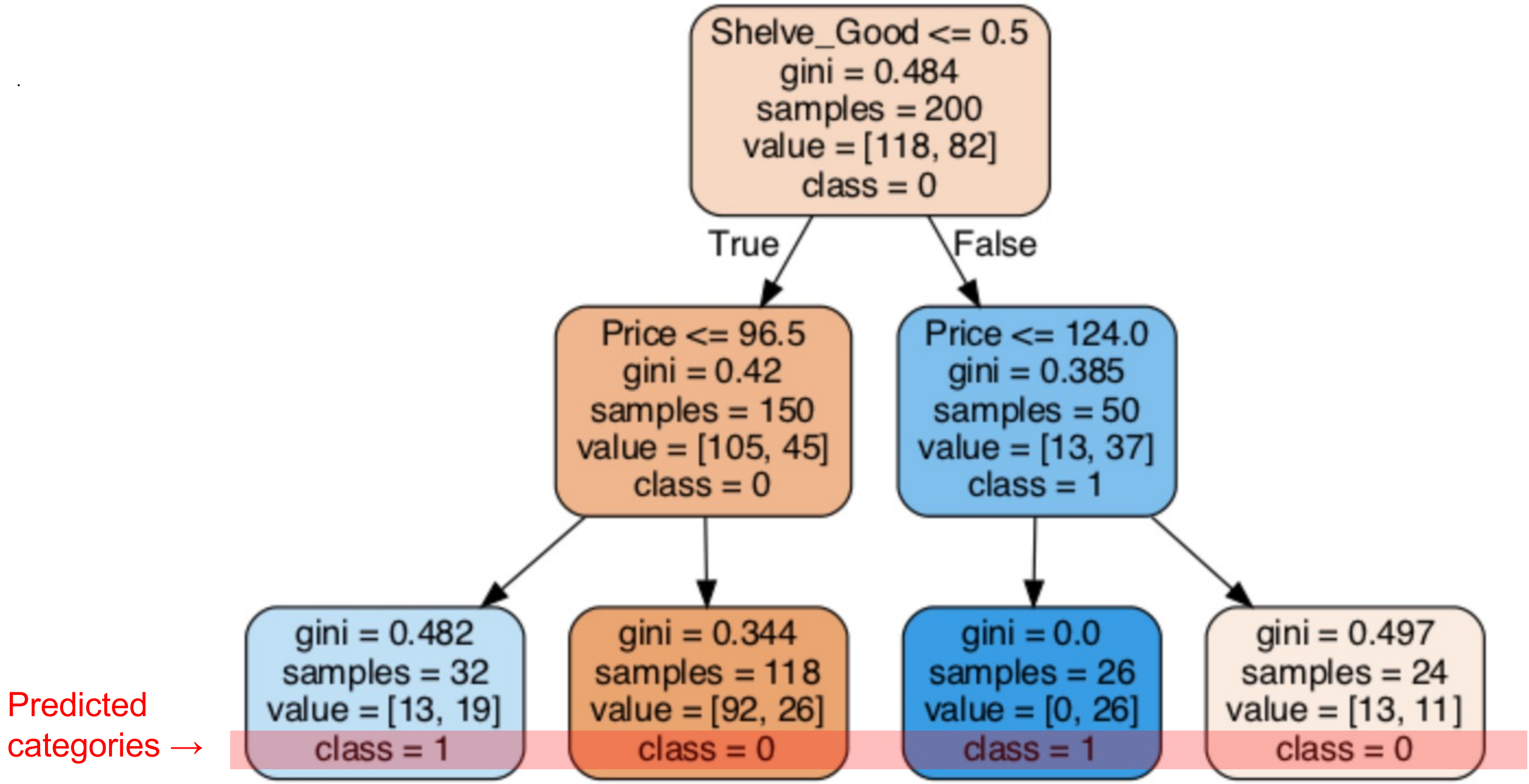


gini index tend to decrease as depth increases

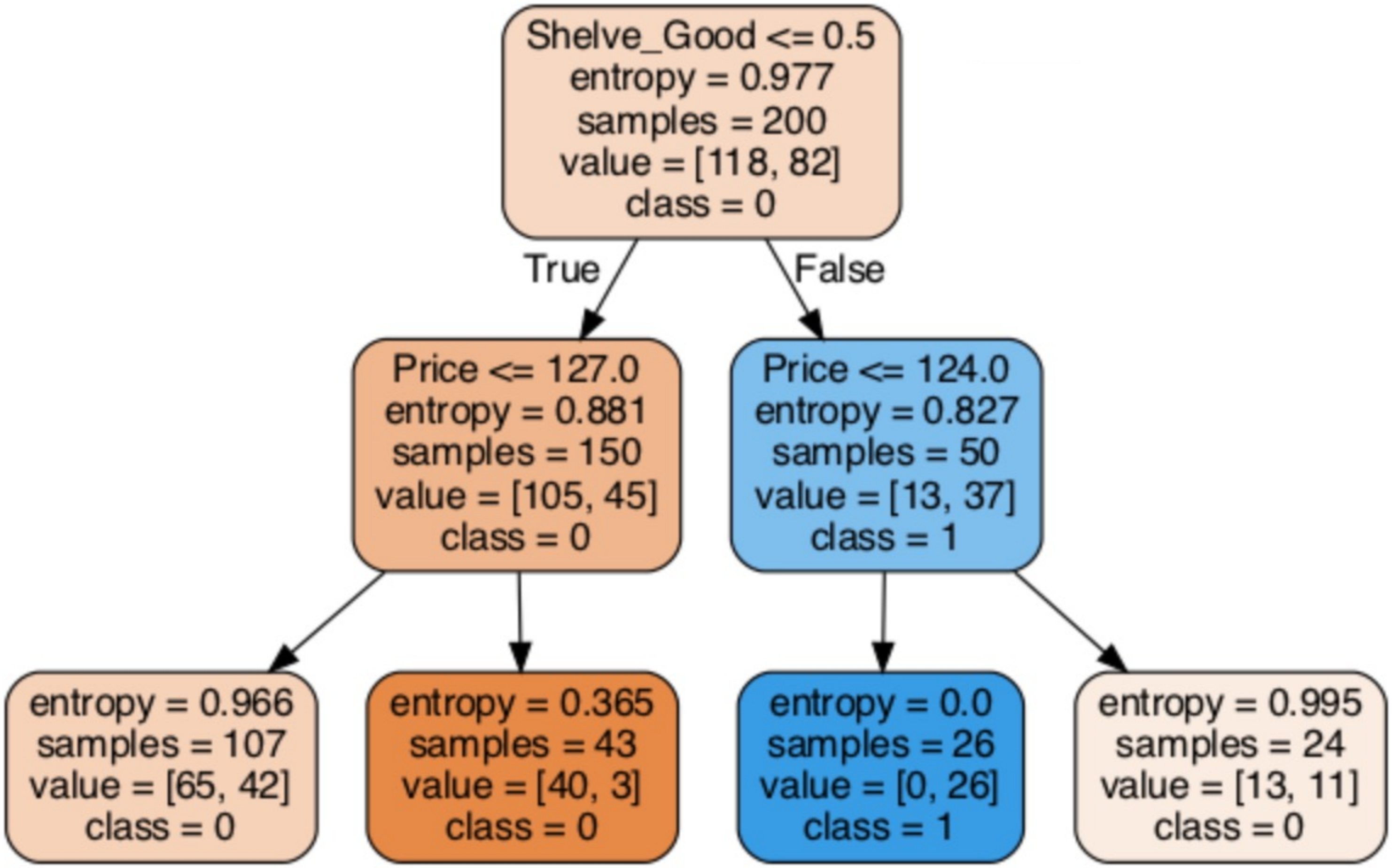
n = 200 stores
categories (high sales, low sales)
overall gini index = 0.484

pure node, all 26 stores are high sales

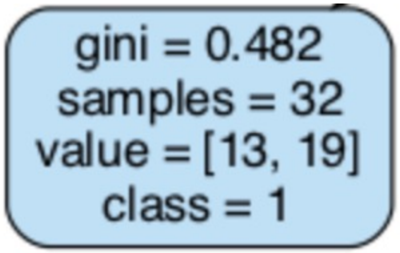
Classification Trees – Store sales Example



Classification Trees – Decision Tree with entropy



Gini Index — *left leaf node*



- gini index is

$$G = \sum_{m=1}^{T} \left[1 - \sum_{i=1}^{K} p_{im}^2 \right]$$

```
> p1 =13/32
> p2 = 19/32
> 1 - p1^2 - p2^2
[1] 0.4824219
```

Entropy *left leaf node*

entropy = 0.966
samples = 107
value = [65, 42]
class = 0

- entropy is

$$D = \sum_{m=1}^{T} \left[\sum_{i=1}^{K} p_{im} \log_2(p_{im}) \right]$$

```
> p1 = 65/107
> p2 = 42/107
> -p1*log2(p1) - p2*log2(p2)
[1] 0.9664087
```

Example – Carseats data

Classification Trees

- The *Carseats* dataset contains the sales of child car seats from 400 stores in different locations in the US
- It includes 10 features
- The response is *Sales*

Carseats variables

- `Sales` Unit sales (in thousands) at each location

  `CompPrice` Price charged by competitor at each location

  `Income` Community income level (in thousands of dollars)

  `Advertising` Local advertising budget for company at each location (in thousands of dollars)

  `Population` Population size in region (in thousands)

  `Price` Price company charges for car seats at each site

  `ShelveLoc` A factor with levels `Bad`, `Good` and `Medium` indicating the quality of the shelving location for the car seats at each site

  `Age` Average age of the local population

  `Education` Education level at each location

  `Urban` A factor with levels `No` and `Yes` to indicate whether the store is in an urban or rural location

  `US` A factor with levels `No` and `Yes` to indicate whether the store is in the US or not

Carseats data - categorical variables

`Sales` Unit sales (in thousands) at each location

`CompPrice` Price charged by competitor at each location

`Income` Community income level (in thousands of dollars)

`Advertising` Local advertising budget for company at each location (in thousands of dollars)

`Population` Population size in region (in thousands)

`Price` Price company charges for car seats at each site

`ShelveLoc` A factor with levels `Bad`, `Good` and `Medium` indicating the quality of the shelving location for the car seats at each site

`Age` Average age of the local population

`Education` Education level at each location

`Urban` A factor with levels `No` and `Yes` to indicate whether the store is in an urban or rural location

`US` A factor with levels `No` and `Yes` to indicate whether the store is in the US or not

Classification Trees – Procedure

- It is of interest to predict if the Sales of a store are high (more than 8000 seats) or low
- Transform Sales into a new categorical response *High*
- Which variables are most useful to predict *High* sales?

Classification Trees – Example

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

```
from sklearn.model_selection import KFold, cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
```

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

```
from sklearn.tree import export_graphviz
import graphviz
```

```
import pydotplus
from IPython.display import Image
```

Classification Trees – Example

```
df0 = pd.read_csv('Carseats.csv')
df0[:9]
```

Y

	Sales	CompPrice	Income	Advertising	Population	Price	ShelveLoc	Age	Education	Urban	US
0	9.50	138	73	11	276	120	Bad	42	17	Yes	Yes
1	11.22	111	48	16	260	83	Good	65	10	Yes	Yes
2	10.06	113	35	10	269	80	Medium	59	12	Yes	Yes
3	7.40	117	100	4	466	97	Medium	55	14	Yes	Yes
4	4.15	141	64	3	340	128	Bad	38	13	Yes	No
5	10.81	124	113	13	501	72	Bad	78	16	No	Yes
6	6.63	115	105	0	45	108	Medium	71	15	Yes	No
7	11.85	136	81	15	425	120	Good	67	10	Yes	Yes
8	6.54	132	110	0	108	124	Medium	76	10	No	No

Classification Trees – Example

```
df0 = pd.read_csv('Carseats.csv')
df0[:9]
```

categorical variable with 3 categories

categorical variables with 2 categories

	Sales	CompPrice	Income	Advertising	Population	Price	ShelveLoc	Age	Education	Urban	US
0	9.50	138	73	11	276	120	Bad	42	17	Yes	Yes
1	11.22	111	48	16	260	83	Good	65	10	Yes	Yes
2	10.06	113	35	10	269	80	Medium	59	12	Yes	Yes
3	7.40	117	100	4	466	97	Medium	55	14	Yes	Yes
4	4.15	141	64	3	340	128	Bad	38	13	Yes	No
5	10.81	124	113	13	501	72	Bad	78	16	No	Yes
6	6.63	115	105	0	45	108	Medium	71	15	Yes	No
7	11.85	136	81	15	425	120	Good	67	10	Yes	Yes
8	6.54	132	110	0	108	124	Medium	76	10	No	No

Classification Trees – Convert categorical features into binary columns

.

```
df1 = pd.get_dummies(df0,columns=['Shelve','Urban','US'])
df1 = df1.drop(['Shelve_Bad','Urban_No','US_No'],axis=1)
df1[:4]
```

	Sales	CompPrice	Income	Advertising	Population	Price	Age	Education	Shelve_Good	Shelve_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

Classification Trees – Create categorical response High

```
df1['High'] = (df1.Sales > 8)
df1[:6]
```

	Sales	CompPrice	Income	Advertising	Population	Price	Age	Education	Shelve_Good	Shelve_Medium	Urban_Yes	US_Yes	High
0	9.50	138	73	11	276	120	42	17	0	0	1	1	True
1	11.22	111	48	16	260	83	65	10	1	0	1	1	True
2	10.06	113	35	10	269	80	59	12	0	1	1	1	True
3	7.40	117	100	4	466	97	55	14	0	1	1	1	False
4	4.15	141	64	3	340	128	38	13	0	0	1	0	False
5	10.81	124	113	13	501	72	78	16	0	0	0	1	True

Classification Trees – Create categorical response High

```
df1['High'] = (df1.Sales > 8).astype(np.int32)
df1[:6]
```

	Sales	CompPrice	Income	Advertising	Population	Price	Age	Education	Shelve_Good	Shelve_Medium	Urban_Yes	US_Yes	High
0	9.50	138	73	11	276	120	42	17	0	0	1	1	1
1	11.22	111	48	16	260	83	65	10	1	0	1	1	1
2	10.06	113	35	10	269	80	59	12	0	1	1	1	1
3	7.40	117	100	4	466	97	55	14	0	1	1	1	0
4	4.15	141	64	3	340	128	38	13	0	0	1	0	0
5	10.81	124	113	13	501	72	78	16	0	0	0	1	1

Classification Trees – Example

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

```
y = df1.High
X = df1.drop(['Sales', 'High'], axis = 1)
X[:3]
```

- keep categorical response in *y*
- drop both responses

	CompPrice	Income	Advertising	Population	Price	Age	Education	Shelve_Good	Shelve_Medium	Urban_Yes	US_Yes
0	138	73	11	276	120	42	17	0	0	1	1
1	111	48	16	260	83	65	10	1	0	1	1
2	113	35	10	269	80	59	12	0	1	1	1

Classification Trees – Example

Split dataset

```
X_train, X_test, y_train, y_test = train_test_split(X,y,stratify = y,
                                                     train_size = 0.5,
                                                     test_size = 0.5,
                                                     random_state = 0)
```

max_depth = 2

```
model1 = DecisionTreeClassifier(criterion='entropy',max_depth = 2)
model1.fit(X_train, y_train)
yhat = model1.predict(X_test)
model1.score(X_test, y_test)
```

```
0.66
```

Classification Trees – Example

```
model2 = DecisionTreeClassifier(criterion='gini',max_depth = 2)
model2.fit(X_train, y_train)
yhat = model2.predict(X_test)
model2.score(X_test, y_test)
```

```
0.7
```

```
pd.crosstab(y_test,yhat,rownames = ['y_test'],colnames = ['yhat'])
```

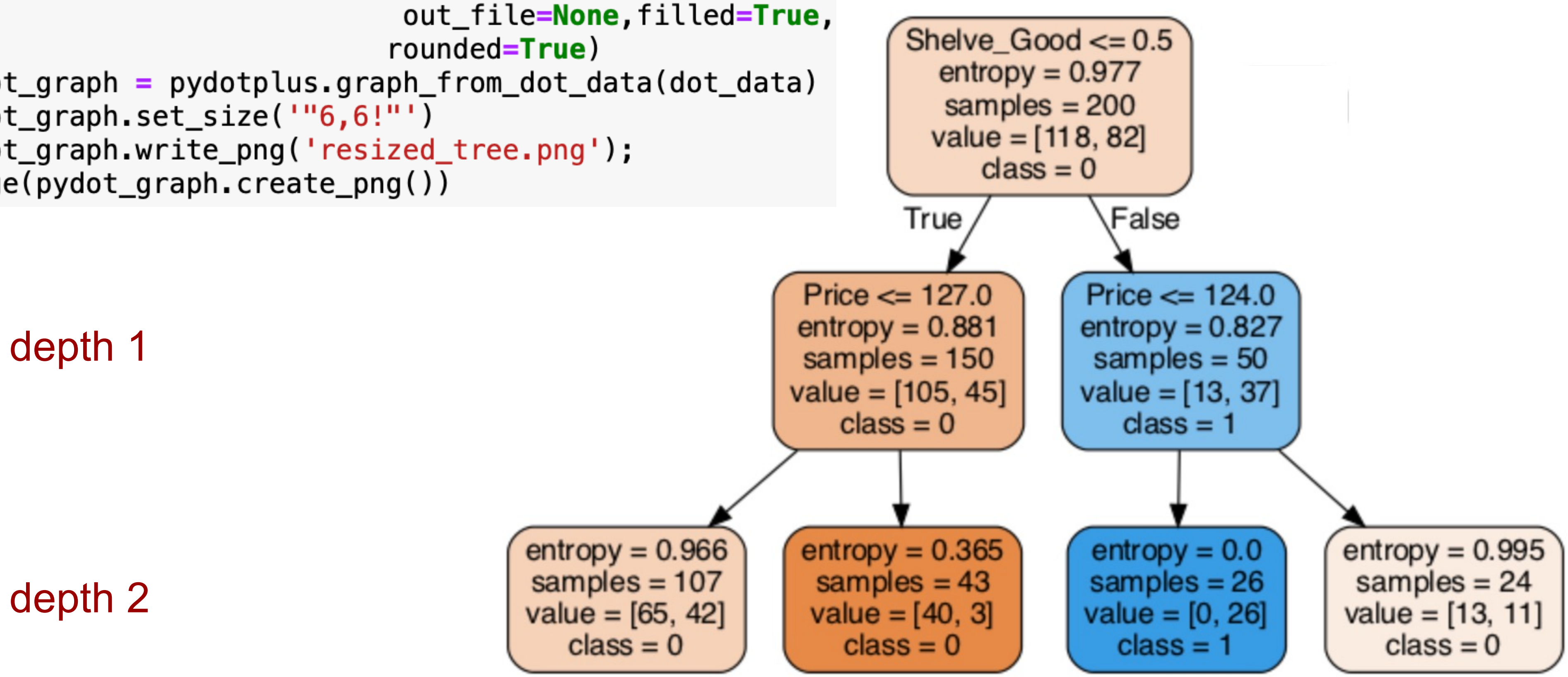
yhat	**0**	**1**
y_test		
0	104	14
1	46	36

test accuracy rate

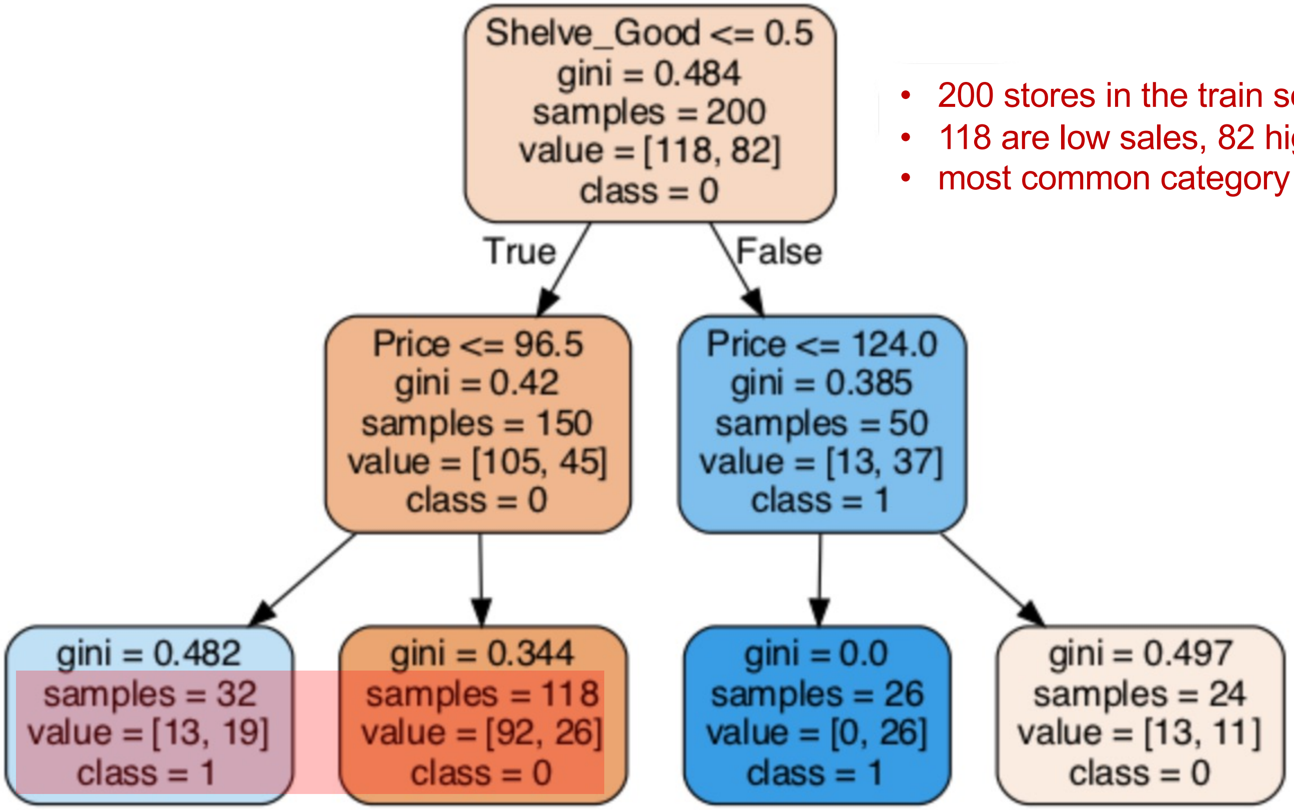
(104 + 36) / 200 = 0.7

Classification Trees – Decision Tree with entropy

```
dot_data = export_graphviz(model1,
                            feature_names=col_names,
                            class_names = ['0','1'],
                            out_file=None,filled=True,
                           rounded=True)
pydot_graph = pydotplus.graph_from_dot_data(dot_data)
pydot_graph.set_size('"6,6!"')
pydot_graph.write_png('resized_tree.png');
Image(pydot_graph.create_png())
```

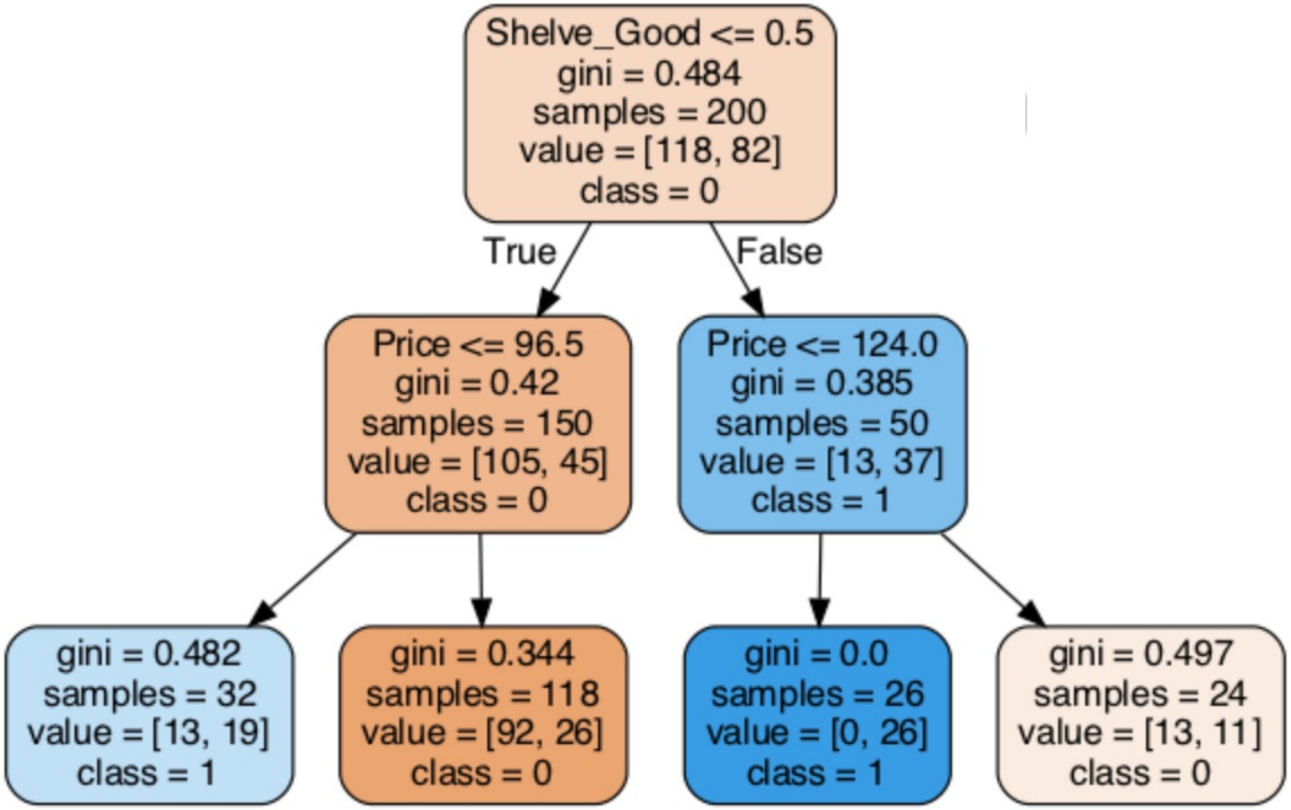


Classification Trees – Decision Tree with gini index



- 200 stores in the train set
- 118 are low sales, 82 high sales
- most common category is category 0

Classification Trees – Predict Sales (High 1, or Low 0) of a new store



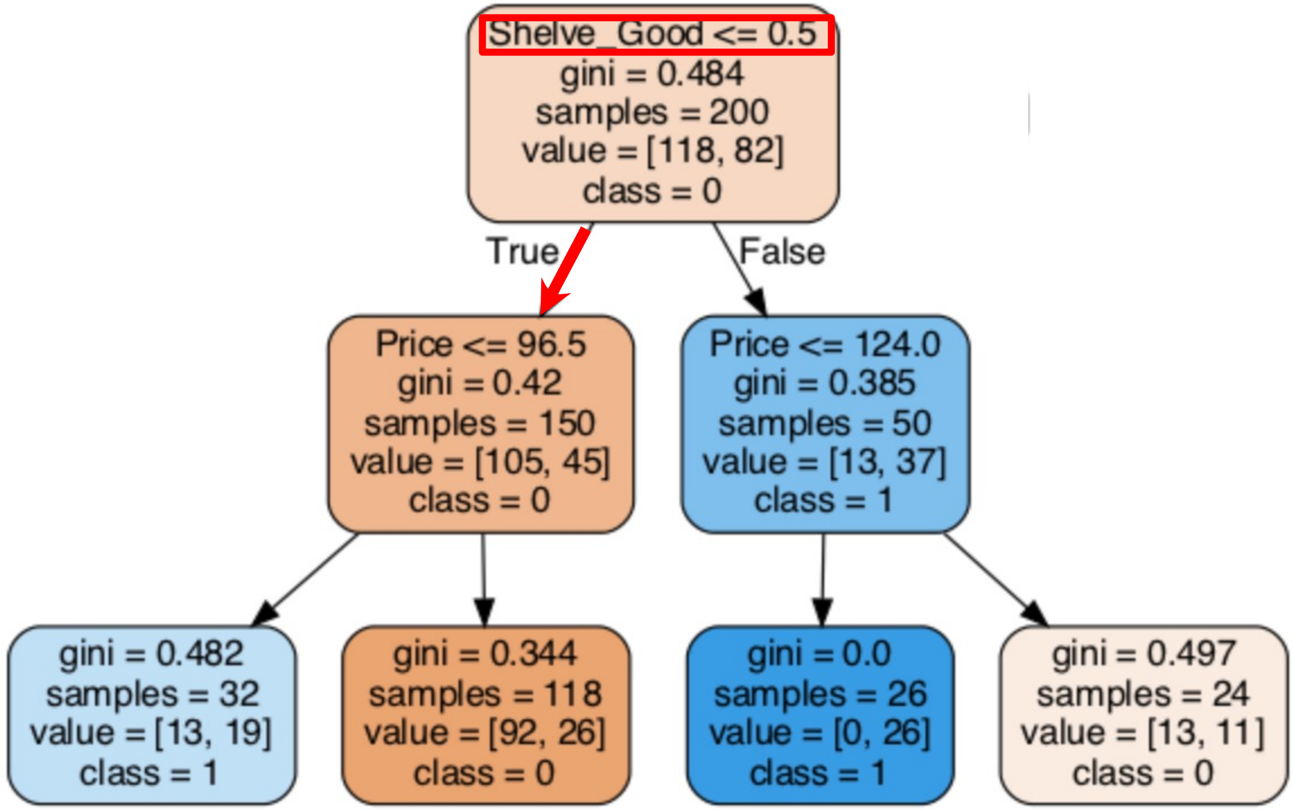
Use gini index model to predict if this new store is low or high sales

```
newval
```

CompPrice	Income	Advertising	Population	Price	Age	Education	Shelve_Good	Shelve_Medium	Urban_Yes	US_Yes
138	73	11	276	120	42	17	0	0	1	1

Classification Trees – Predict Sales (High 1, or Low 0) of a new store

.

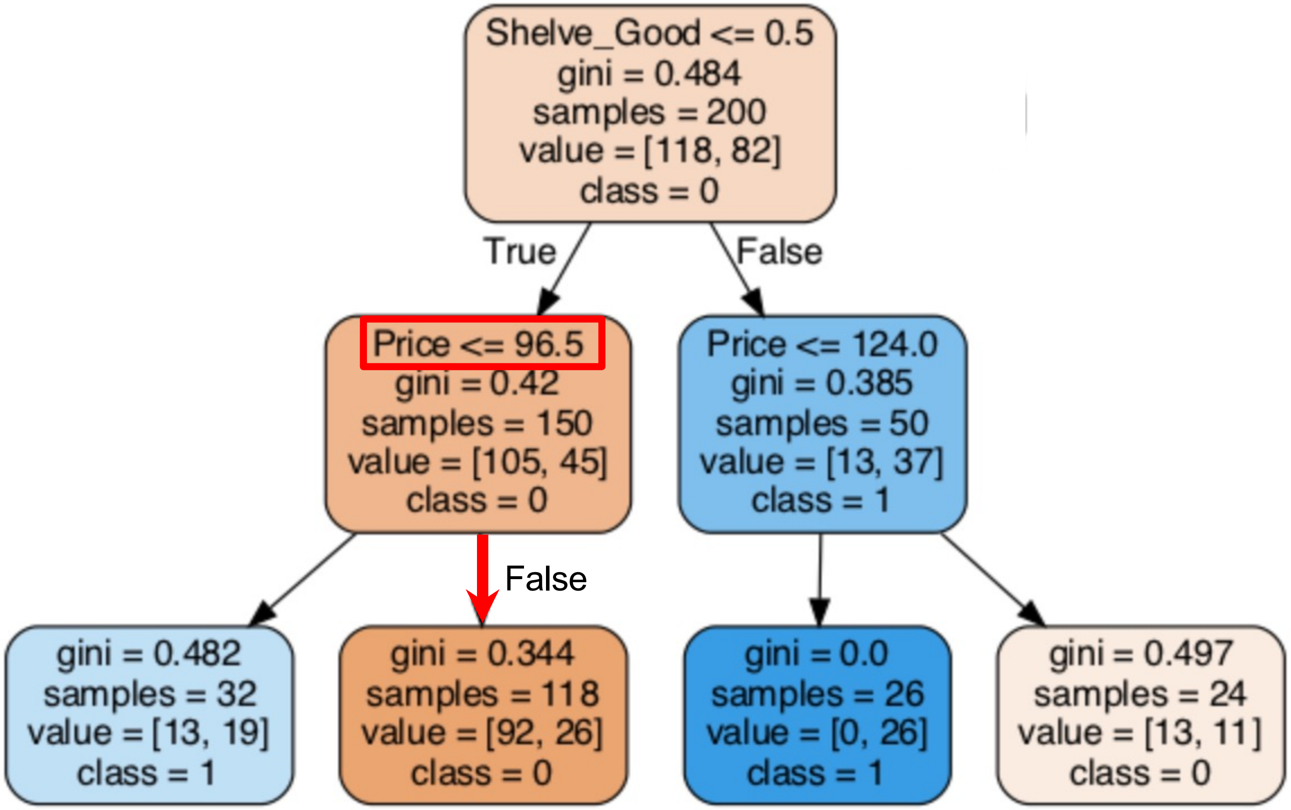



```
newval
```

	CompPrice	Income	Advertising	Population	Price	Age	Education	Shelve_Good	Shelve_Medium	Urban_Yes	US_Yes
	138	73	11	276	120	42	17	0	0	1	1

Classification Trees – Predict Sales (High 1, or Low 0) of a new store

.

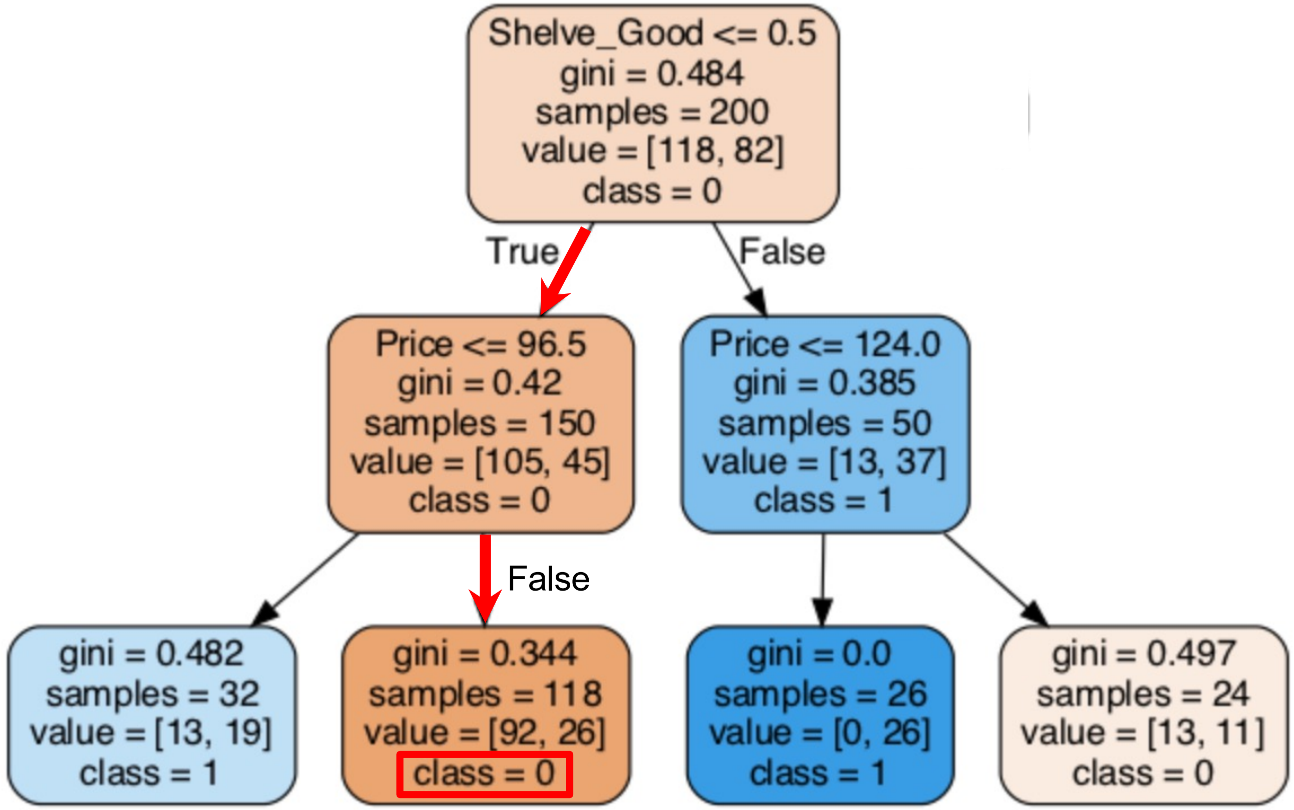



```
newval
```

	CompPrice	Income	Advertising	Population	Price	Age	Education	Shelve_Good	Shelve_Medium	Urban_Yes	US_Yes
	138	73	11	276	120	42	17	0	0	1	1

Classification Trees – Predict Sales (High 1, or Low 0) of a new store

.



This store is predicted as "low sales" →

CompPrice	Income	Advertising	Population	Price	Age	Education	Shelve_Good	Shelve_Medium	Urban_Yes	US_Yes
138	73	11	276	120	42	17	0	0	1	1

Test accuracy rate

.

max_depth = 2

```
model2 = DecisionTreeClassifier(max_depth = 2)
model2.fit(X_train, y_train)
model2.score(X_test, y_test)
```

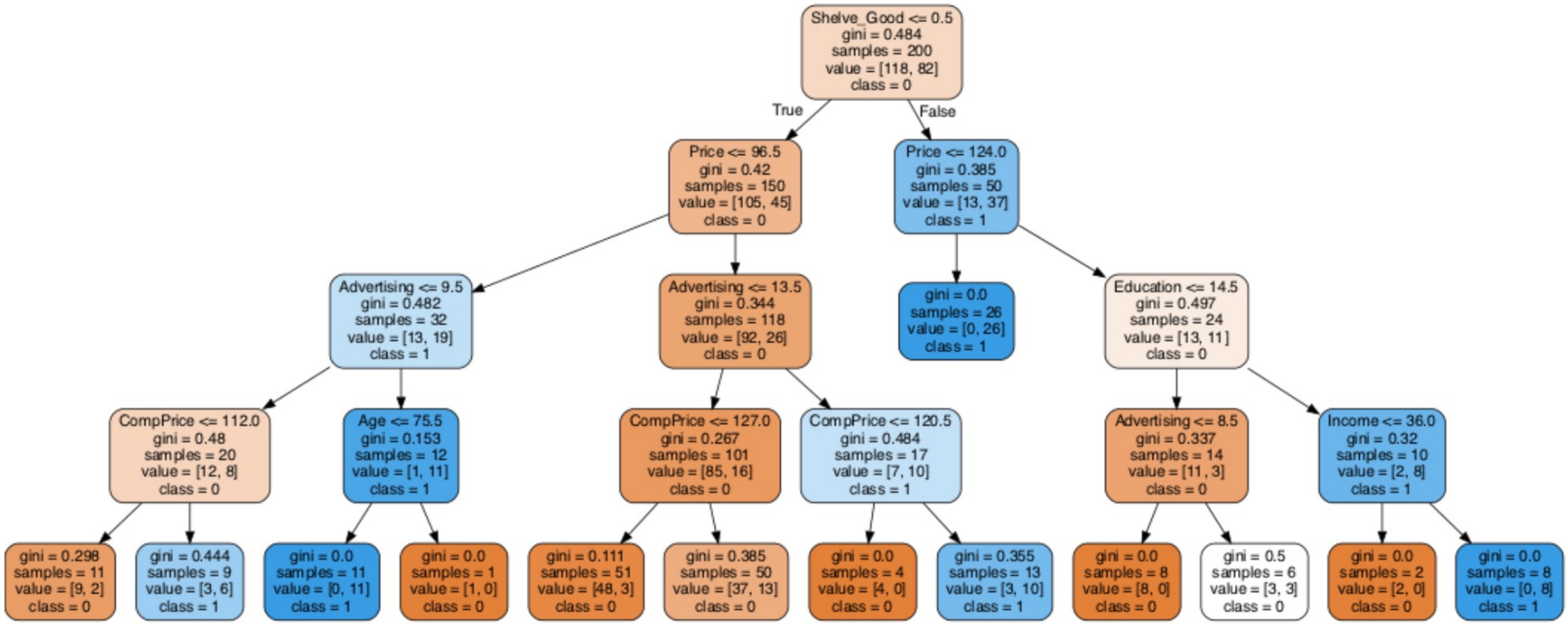
```
0.7
```

max_depth = 4

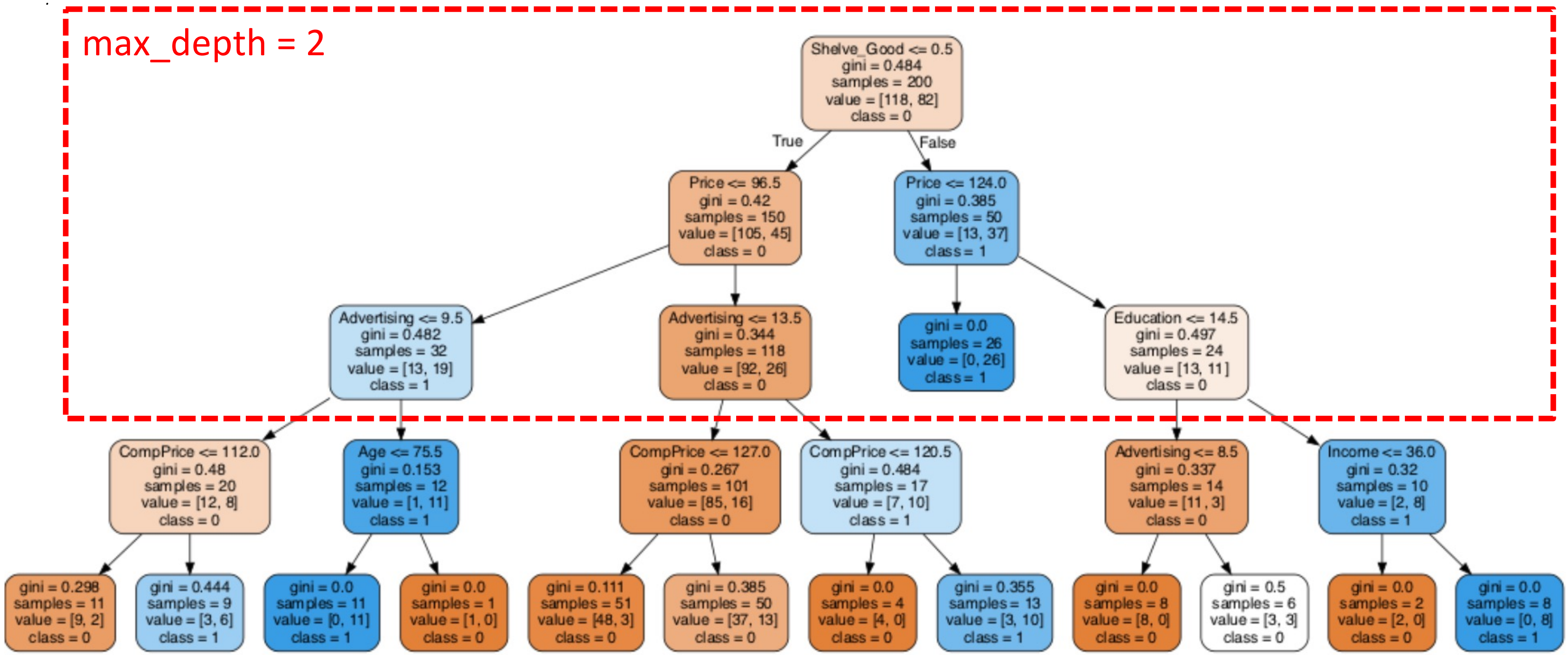
```
model4 = DecisionTreeClassifier(max_depth = 4)
model4.fit(X_train, y_train)
model4.score(X_test, y_test)
```

```
0.74
```

Classification Trees (max_depth = 4)

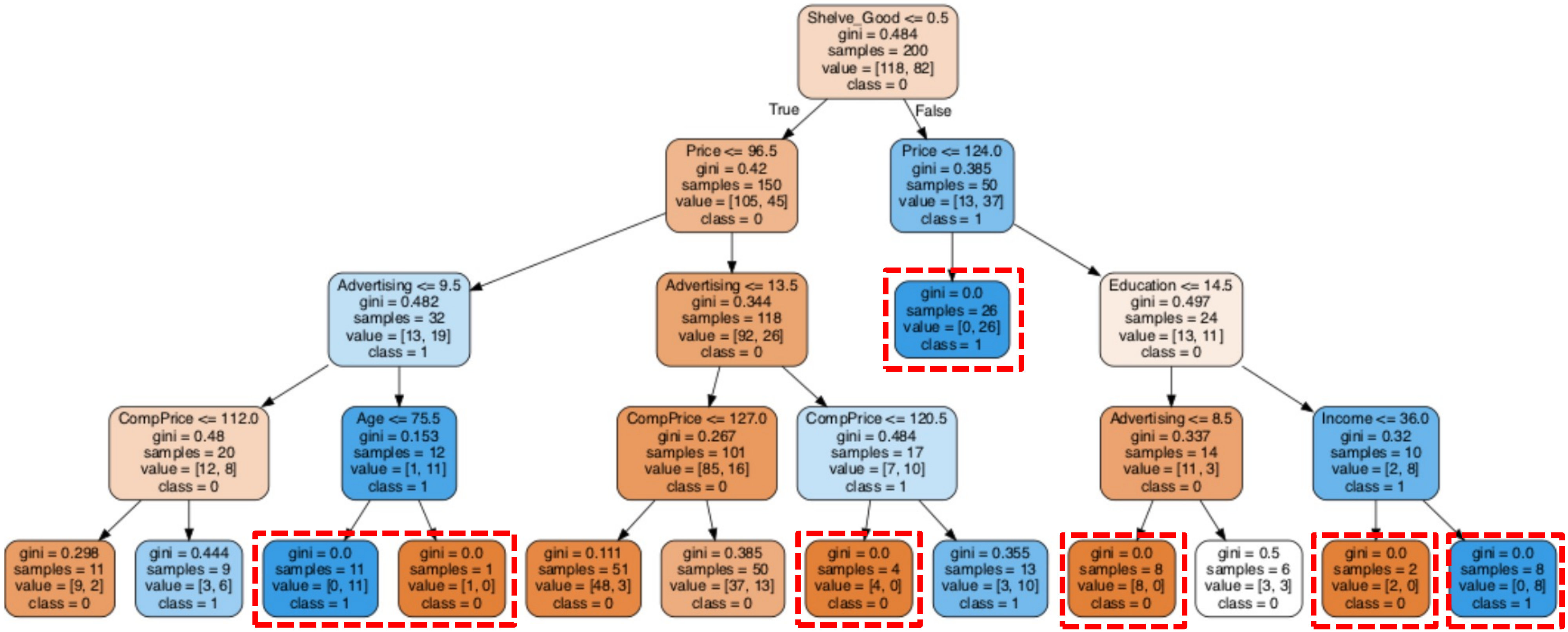


Classification Trees (max_depth = 4)



Classification Trees – Pure nodes

.



There are 7 pure nodes (all stores belong to same category) ↑ All 8 are low sales stores

Holdout Cross Validation

Test accuracy rate

max_depth = 2

```
model2 = DecisionTreeClassifier(max_depth = 2)
model2.fit(X_train, y_train)
model2.score(X_test, y_test)
```

```
0.7
```

max_depth = 4

```
model4 = DecisionTreeClassifier(max_depth = 4)
model4.fit(X_train, y_train)
model4.score(X_test, y_test)
```

```
0.74
```

Classification Trees – Select best value for max_depth

max_depth = 2

```
model2 = DecisionTreeClassifier(max_depth = 2)
model2.fit(X_train, y_train)
model2.score(X_test, y_test)
```

```
0.7
```

```
model = DecisionTreeClassifier(random_state=1)
accuracy = []
```

```
for i in range(2,22):
    model.set_params(max_depth = i)
    model.fit(X_train, y_train)
    acc = model.score(X_test, y_test)
    accuracy.append(acc)
```

```
print(accuracy)
```

```
[0.7, 0.705, 0.72, 0.75, 0.735, 0.75, 0.71, 0.76,
```

Holdout CV to find best value for max_depth

```
X_nontest,X_test,y_nontest,y_test = train_test_split(X,y,
                                                    test_size=0.40,
                                                    random_state=1)
```

```
X_train,X_validation,\
y_train,y_validation = train_test_split(X_nontest,y_nontest,
                                        test_size=0.5,random_state=1)
```

```
depths = range(2,22)
```

```
model = DecisionTreeClassifier(random_state=1)
accuracy = []

for i in range(2,22):
    model.set_params(max_depth = i)
    model.fit(X_train, y_train)
    accuracy_i = model.score(X_validation, y_validation)
    accuracy.append(accuracy_i)
```

Holdout CV to find best value for max_depth

```
df = pd.DataFrame(accuracy,columns = ['Val_Accuracy'])
df.index = depths
df.index.name = 'depth'
df[:11]
```

depth	Val_Accuracy
2	0.691667
3	0.716667
4	0.741667
5	0.716667
6	0.691667
7	0.691667
8	0.691667
9	0.675000
10	0.675000
11	0.675000
12	0.675000

```
max1 = df['Val_Accuracy'].max()
max1
```

```
0.7416666666666667
```

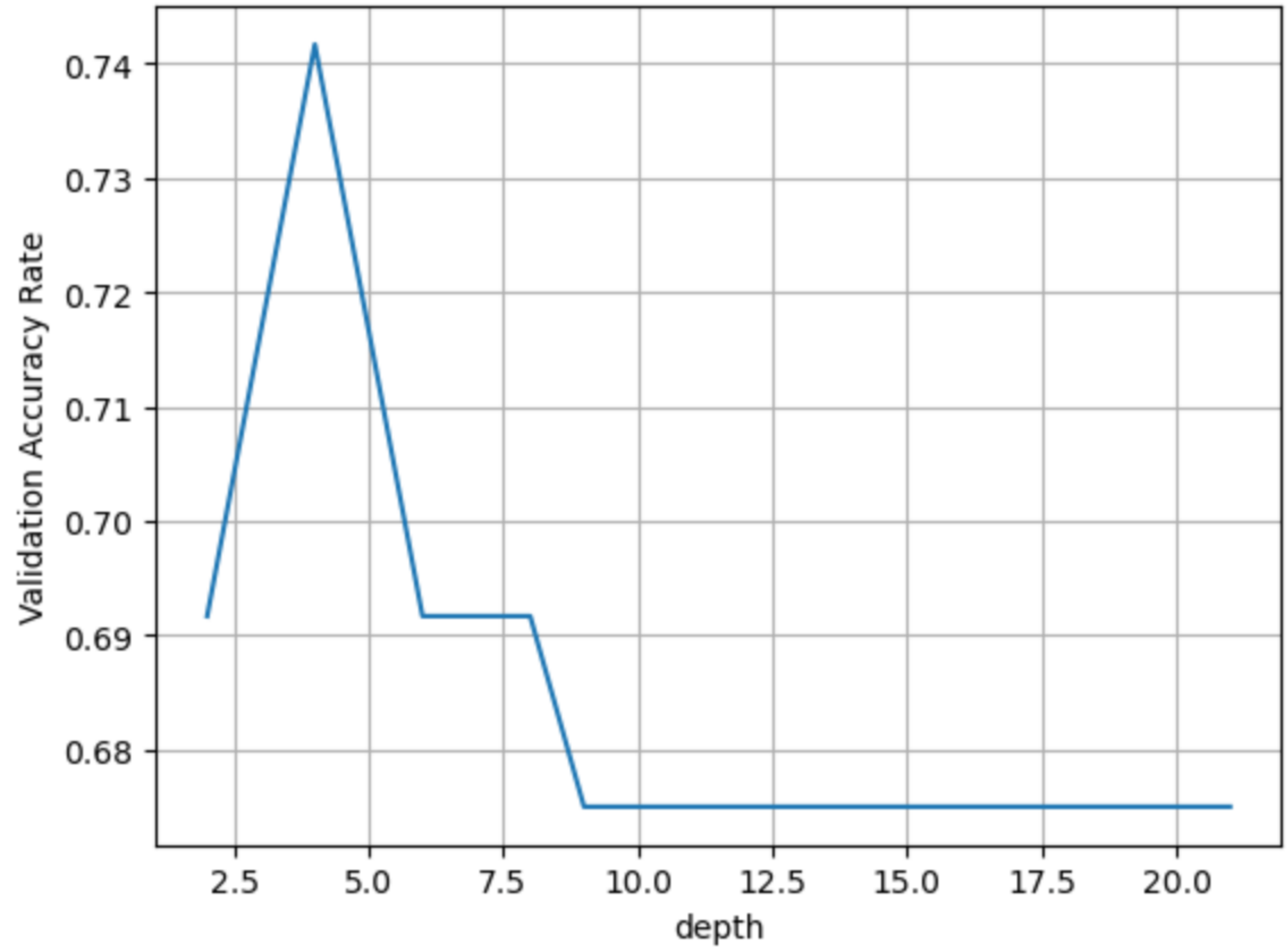
```
df[df.Val_Accuracy == max1]
```

depth	Val_Accuracy
4	0.741667

```
# best hyperparam value
# (maximizing validation accuracy rate)
best_depth = df.Val_Accuracy.idxmax()
best_depth
```

```
4
```

Holdout CV to find best value for max_depth



Holdout CV – Test Accuracy rate

```
# Test accuracy rate
```

```
model9 = DecisionTreeClassifier(max_depth = best_depth,
                                random_state=1)
model9.fit(X_nontest, y_nontest)
model9.score(X_test, y_test)
```

```
0.70625
```

EXAMPLE

Feature importance

Classification Trees – Feature Importance

```
model9.feature_importances_
```

```
array([0.08444228, 0.        , 0.10678597, 0.0138254 , 0.44880677,
       0.04877601, 0.03484001, 0.23483022, 0.        , 0.        ,
       0.02769334])
```

```
X.columns
```

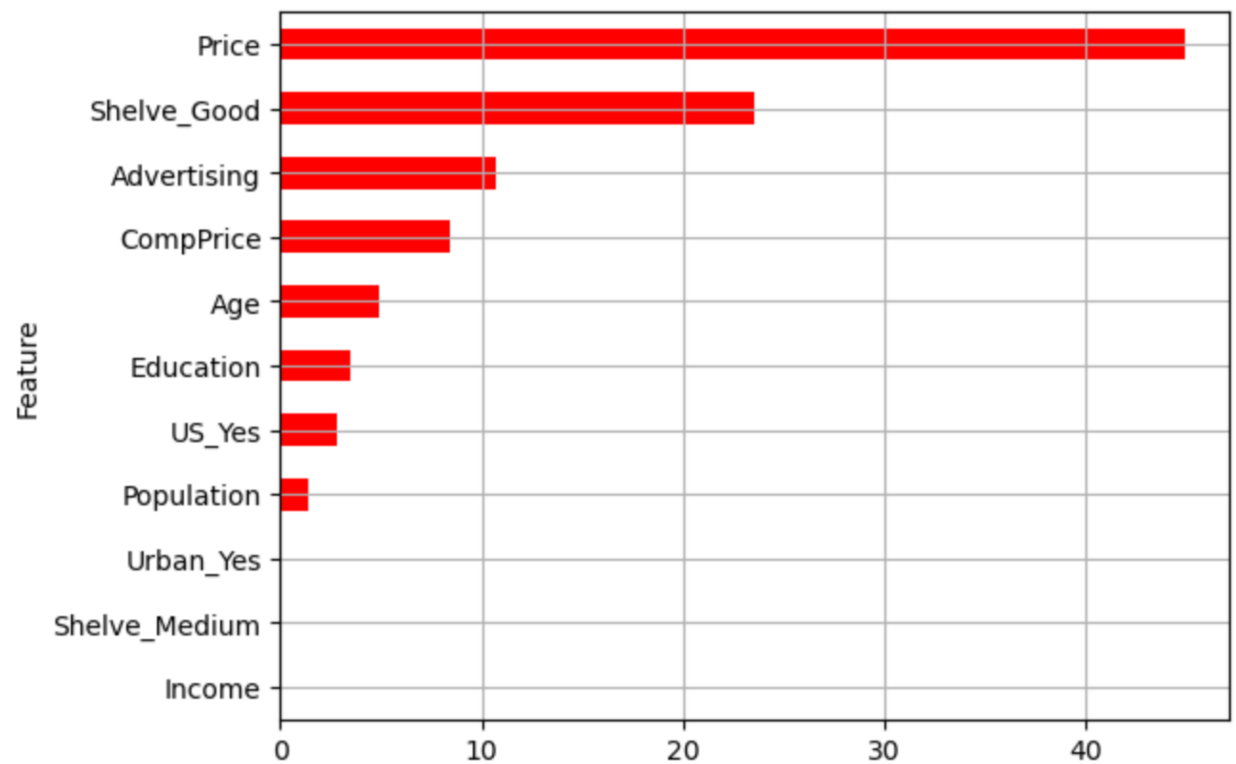
```
Index(['CompPrice', 'Income', 'Advertising', 'Population', 'Price', 'Age',
       'Education', 'Shelve_Good', 'Shelve_Medium', 'Urban_Yes', 'US_Yes'],
      dtype='object')
```

```
df9 = pd.DataFrame(100*model9.feature_importances_,
                   index = X.columns,
                   columns=['importance'])
df9 = df9.sort_values(by = 'importance',axis=0,
                      ascending=False)
```

Classification Trees – Feature Importance

	importance
Price	44.880677
Shelve_Good	23.483022
Advertising	10.678597
CompPrice	8.444228
Age	4.877601
Education	3.484001
US_Yes	2.769334
Population	1.382540
Income	0.000000
Shelve_Medium	0.000000
Urban_Yes	0.000000

```
df9 = df9.sort_values(by = 'importance',axis=0)
df9.plot(kind='barh',color='r',legend = False)
plt.xlabel('Importance')
plt.ylabel('Feature')
plt.grid()
```



k-Fold Cross Validation

K-Fold Cross validation – no parameter tuning

5-fold cross validation with max_depth = 7

```
kfold = StratifiedKFold(n_splits = 5,shuffle = True, random_state=1)
model = DecisionTreeClassifier(max_depth = 7,random_state=1)
test_accuracy_rates = cross_val_score(model,X,y,cv=kfold)
test_accuracy_rates
```

← loop over 5 folds

```
array([0.65  , 0.7875, 0.8375, 0.7125, 0.625 ])
```

```
# Test accuracy rate
test_accuracy_rates.mean()
```

```
0.7224999999999999
```

Kfold CV - Search best value for hyperparameter max_depth

5-fold cross validation to find best max_depth

```
X_nontest,X_test,y_nontest,y_test = train_test_split(X,y,
                                                      test_size=0.40,
                                                      random_state=1)
```

```
model = DecisionTreeClassifier(random_state=1)
parameters = {'max_depth':range(3,20)}
```

```
grid = GridSearchCV(model, parameters, cv=kfold)
grid.fit(X_nontest, y_nontest);
```

- loop over max_depth range
- loop over 5 folds

```
# Best depth
grid.best_params_
```

```
{'max_depth': 6}
```

```
# Best Validation Accuracy Rate
grid.best_score_
```

```
0.7666666666666667
```

Kfold CV – Test Accuracy rate

```
grid = GridSearchCV(model, parameters, cv=kfold)
grid.fit(X_nontest, y_nontest);
```

```
# Test Accuracy Rate
best_model = grid.best_estimator_
best_model.score(X_test, y_test)
```

```
0.775
```

```
# Confusion Matrix
best_model.fit(X_nontest, y_nontest)
yhat = best_model.predict(X_test)
```

```
df1 = pd.crosstab(y_test,yhat,
             rownames = ['y_test'],
             colnames = ['yhat'])
df1
```

yhat	0	1
y_test		
0	79	14
1	22	45

```
# Test Accuracy Rate from the Confusion matrix
np.diag(df1).sum()/df1.to_numpy().sum()
```

```
0.775
```

EXAMPLE

Feature importance

Classification Trees – Feature Importance

```
best_model.feature_importances_
```

```
array([0.16034346, 0.03560911, 0.11234705, 0.02766065, 0.35954997,
       0.07983719, 0.02311168, 0.15577843, 0.02739163, 0.        ,
       0.01837083])
```

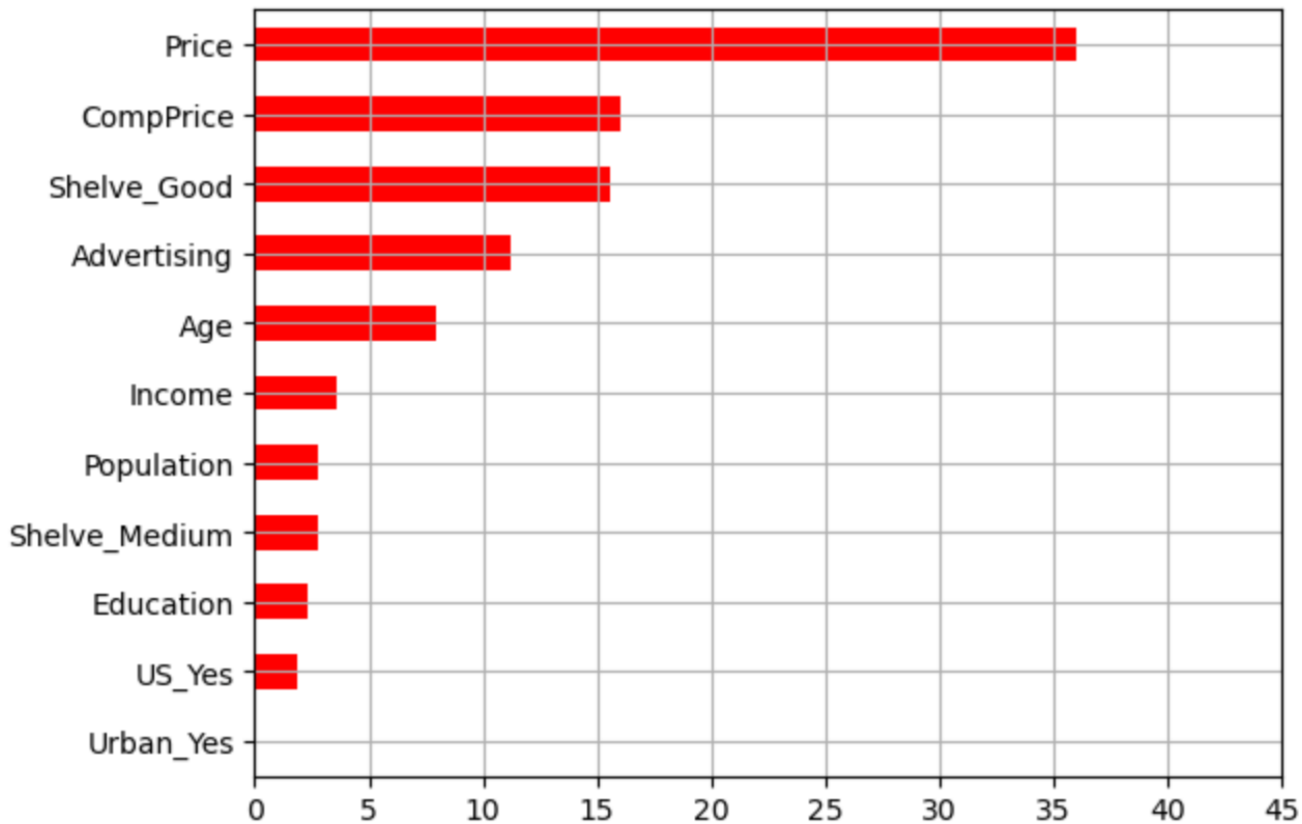
```
X.columns
```

```
Index(['CompPrice', 'Income', 'Advertising', 'Population', 'Price', 'Age',
       'Education', 'Shelve_Good', 'Shelve_Medium', 'Urban_Yes', 'US_Yes'],
      dtype='object')
```

```
df9 = pd.DataFrame(100*best_model.feature_importances_,
                   index = X.columns,
                   columns=['importance'])
df9 = df9.sort_values(by = 'importance',axis=0,
                      ascending=False)
```

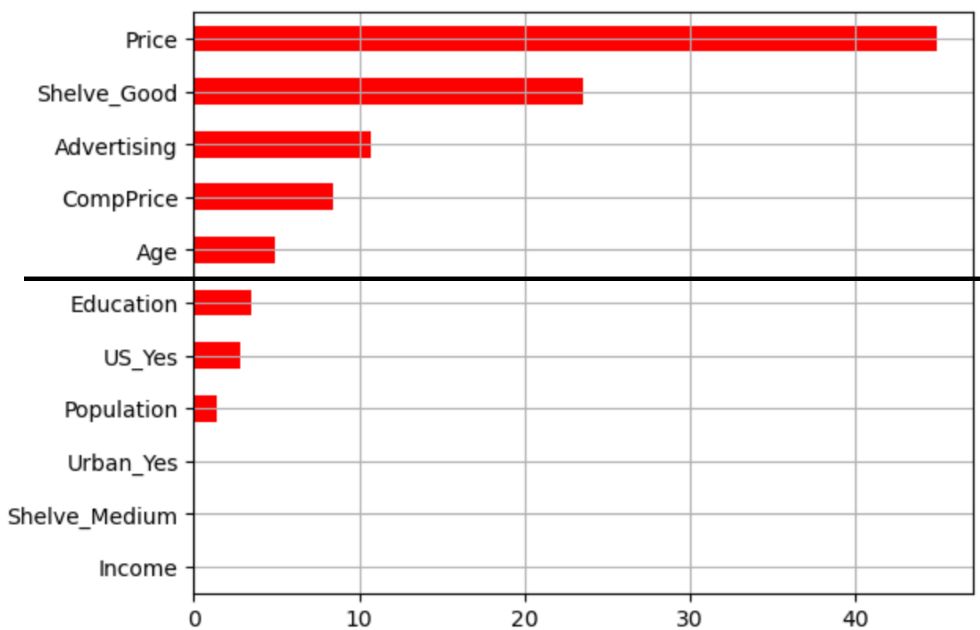
Classification Trees – Feature Importance

	importance
Price	35.954997
CompPrice	16.034346
Shelve_Good	15.577843
Advertising	11.234705
Age	7.983719
Income	3.560911
Population	2.766065
Shelve_Medium	2.739163
Education	2.311168
US_Yes	1.837083
Urban_Yes	0.000000



Classification Trees – Feature Importance Comparison

Holdout CV



k-Fold CV