Decision Trees

Introduction

Contents

What is Segmentation?

What is Segmentation?

- Imagine a scenario where we want to run a SMS marketing campaign to attract more customers in the next quarter
  - Some customers like to see high discount
  - Some customers want to see a large collection of items
  - Some customers are fans of particular brands
  - Some customers are Male some are Female
- Divide them based on their demographics, buying patterns and profile related attributes

What is Segmentation?

- One size doesn't fit all
- Divide the population in such a way that
  - Customers inside a group are homogeneous
  - Customers across groups are heterogeneous
- Is there any statistical way of dividing them correctly based on the data

Segmentation Business Problem

The Business Problem

Old Data

Gender	Marital Status	Ordered the product
M	Married	No
F	Unmarried	Yes
M	Married	No
M	Married	No
M	Married	No
M	Married	No
F	Unmarried	Yes
M	Unmarried	Yes
F	Married	No
M	Married	No
F	Married	No
M	Unmarried	No
F	Married	No
F	Unmarried	Yes

New Data

Gender	Marital Status	Product order
M	Married	??
F	Unmarried	??

The Business Problem

Old Data

Sr No	Gender	Marital Status	Ordered the product
1	M	Married	No
2	F	Unmarried	Yes
3	M	Married	No
4	M	Married	No
5	M	Married	No
6	M	Married	No
7	F	Unmarried	Yes
8	M	Unmarried	Yes
9	F	Married	No
10	M	Married	No
11	F	Married	No
12	M	Unmarried	No
13	F	Married	No
14	F	Unmarried	Yes

New Data

Gender	Marital Status	Product order
M	Married	??
F	Unmarried	??

The Decision Tree Philosophy

The Data

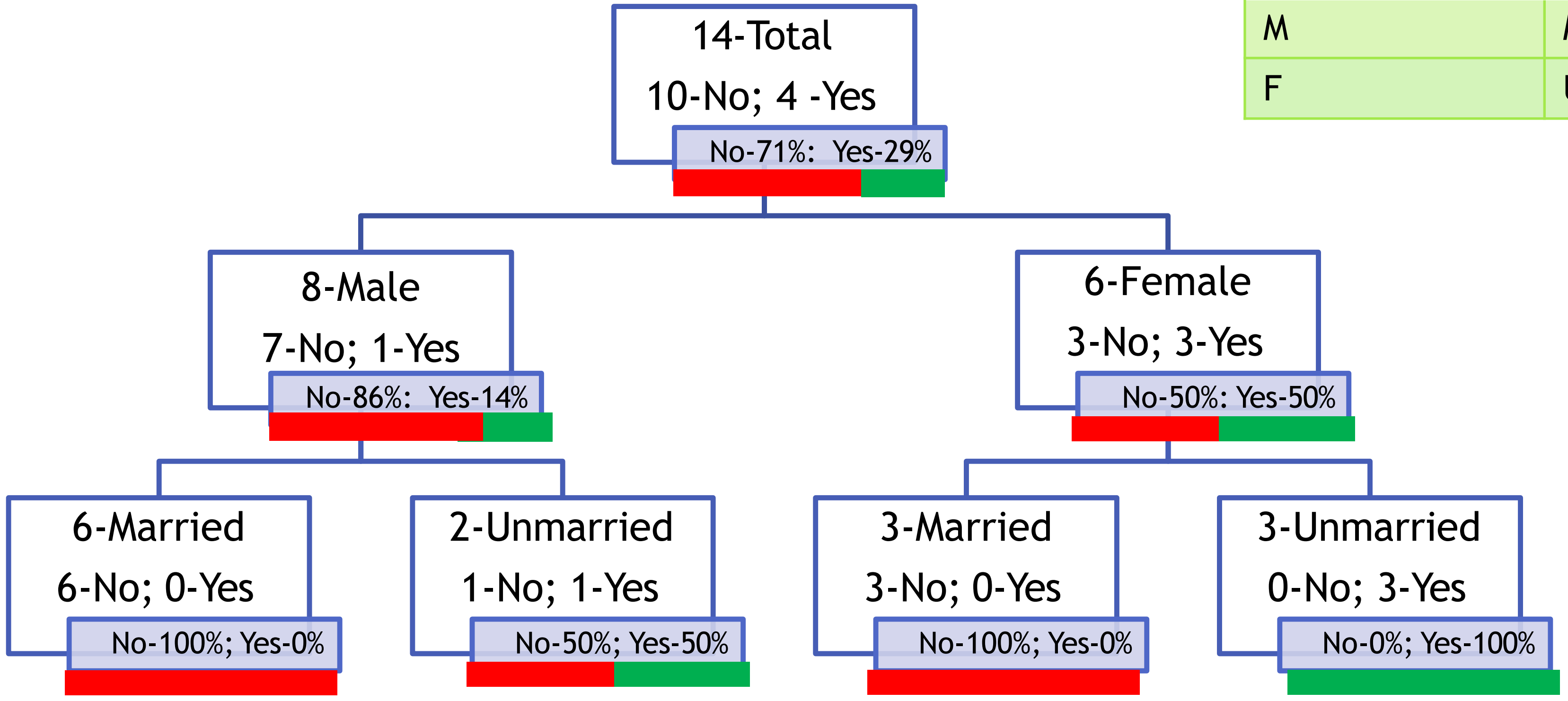
Old Data

Sr No	Gender	Marital Status	Ordered the product
1	M	Married	No
2	F	Unmarried	Yes
3	M	Married	No
4	M	Married	No
5	M	Married	No
6	M	Married	No
7	F	Unmarried	Yes
8	M	Unmarried	Yes
9	F	Married	No
10	M	Married	No
11	F	Married	No
12	M	Unmarried	No
13	F	Married	No
14	F	Unmarried	Yes

New Data

Gender	Marital Status	Product order
M	Married	??
F	Unmarried	??

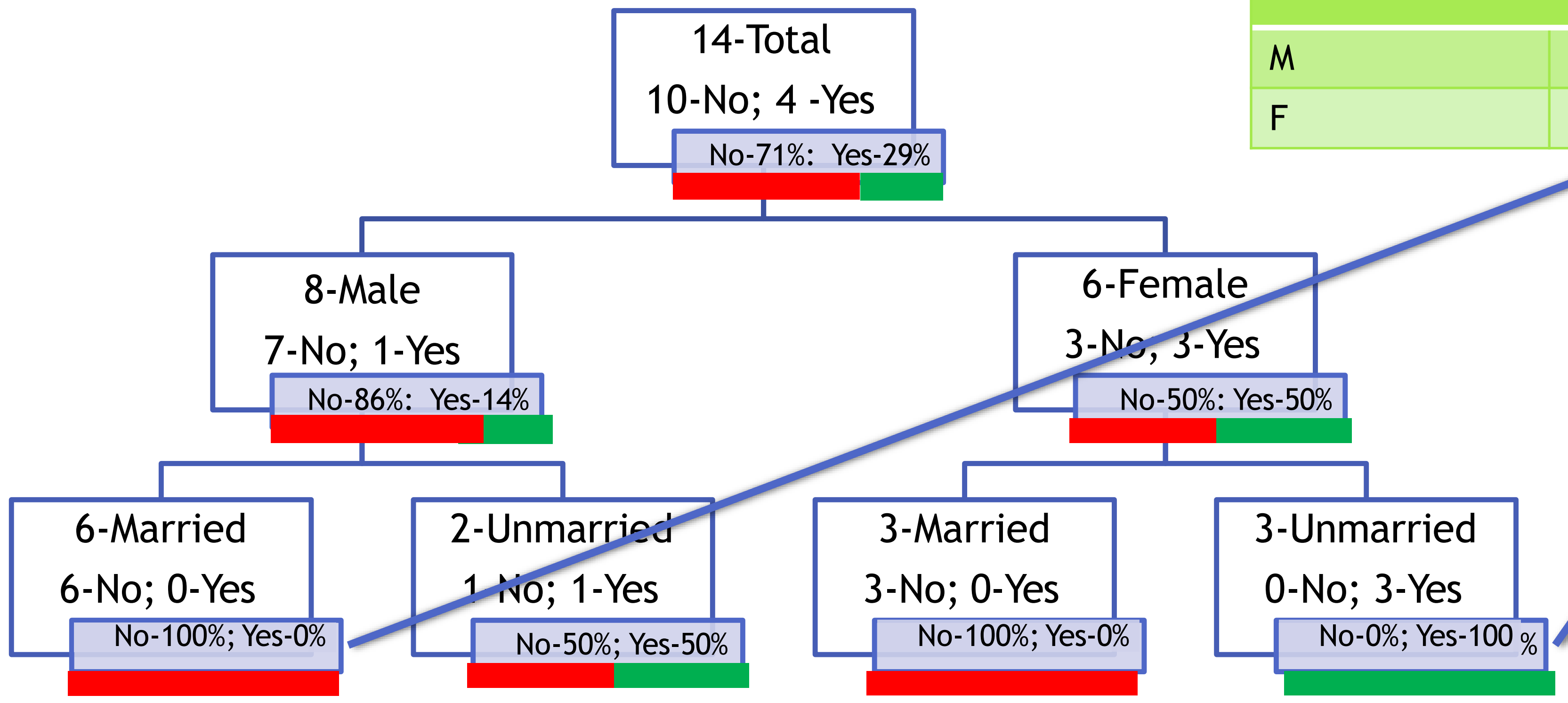
Re-Arranging the data



New Data

Gender	Marital Status	Product order
M	Married	??
F	Unmarried	??

Re-Arranging the data



New Data

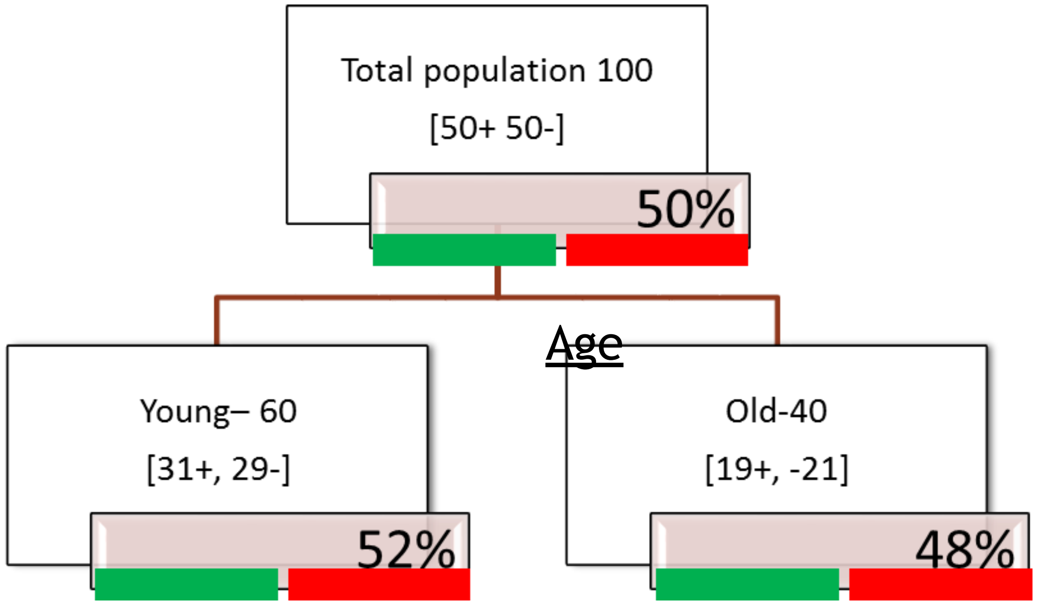
Gender	Marital Status	Product order
M	Married	??
F	Unmarried	??

The Decision Tree Approach

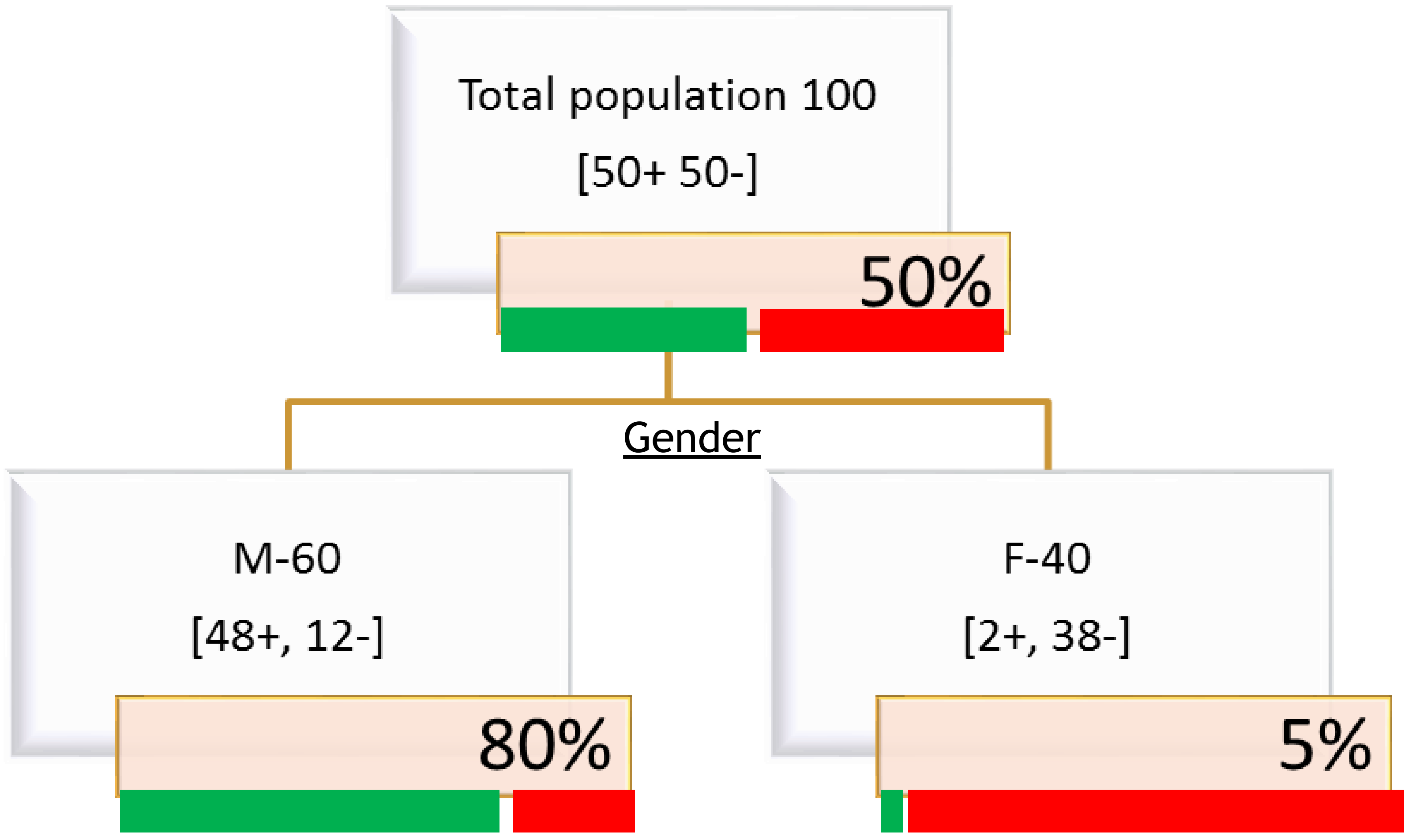
The Decision Tree Approach

- The aim is to decide the whole population or the data set into segments
- The segmentation need to be useful for business decision making.
- If one class is really dominating in a segments
  - Then it will be easy for us to classify the unknown items
  - Then its very easy for applying business strategy
- For example:
  - It takes no great skill to say that the customers have 50% chance to buy and 50% chance to not buy.
  - A good splitting criterion segments the customers with 90% -10% buying probability, say Gender="Female" customers have 5% buying probability and 95% not buying

Example Sales Segmentation Based on Age



Example Sales Segmentation Based on Gender



Main questions

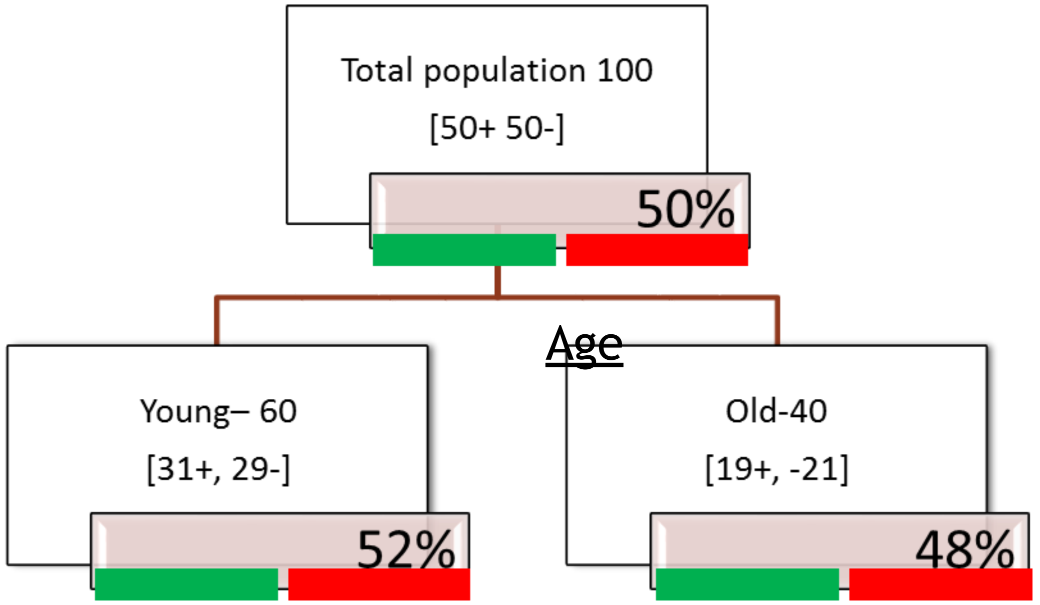
- Ok we are looking for pure segments
- Dataset has many attributes
- Which is the right attribute for pure segmentation?
- Can we start with any attribute?
- Which attribute to start? – The best separating attribute
- Customer Age can impact the sales, gender can impact sales , customer place and demographics can impact the sales. How to identify the best attribute and the split?

The Splitting Criterion

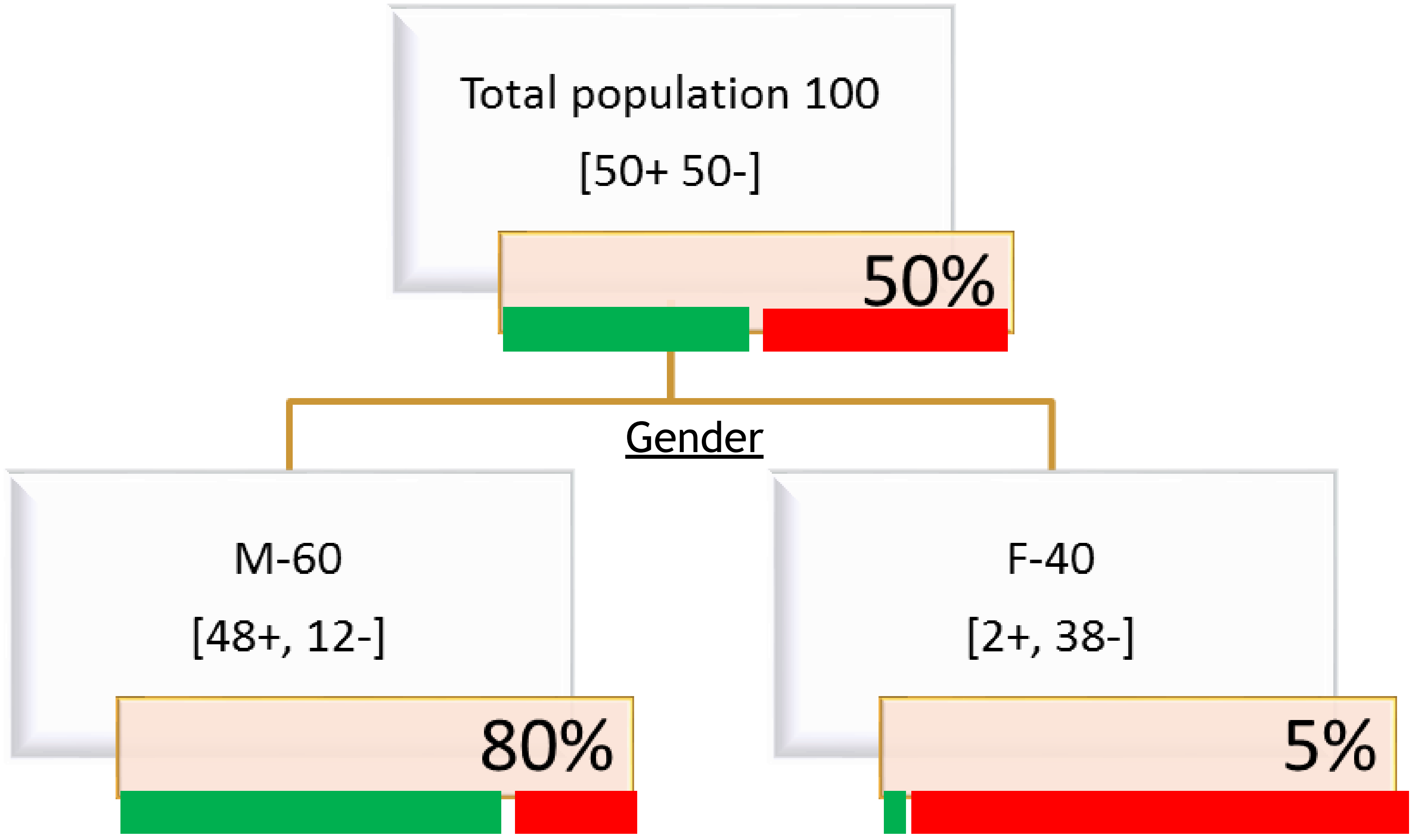
The Splitting Criterion

- The best split is
  - The split does the best job of separating the data into groups
    - Where a single class(either 0 or 1) predominates in each group

Example Sales Segmentation Based on Age

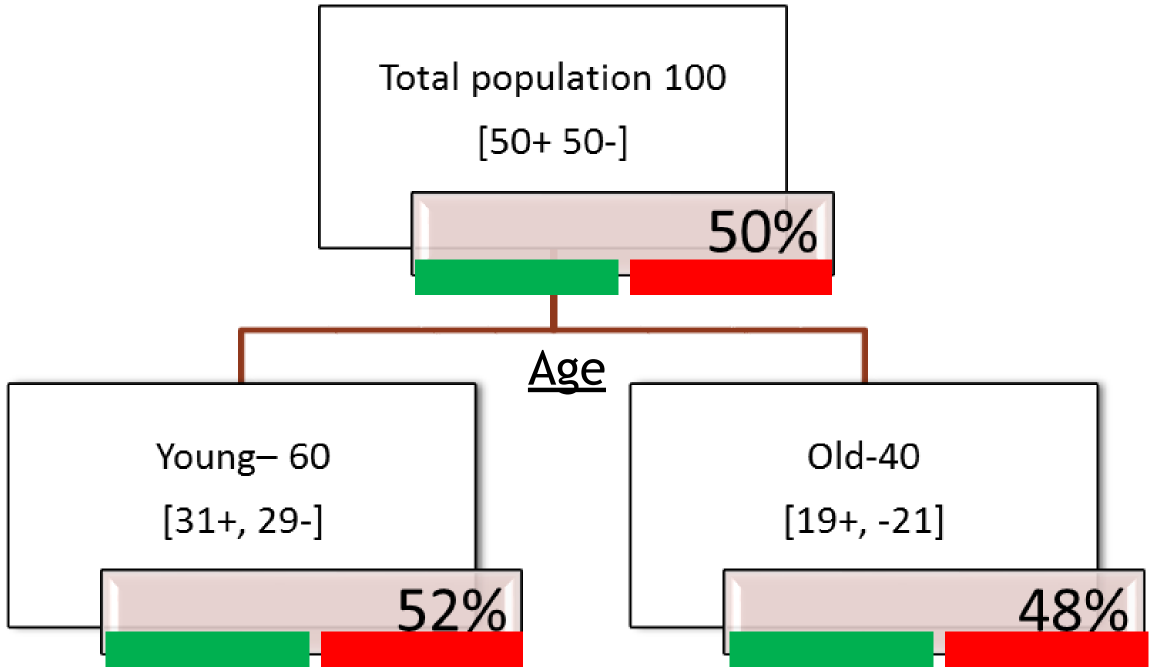


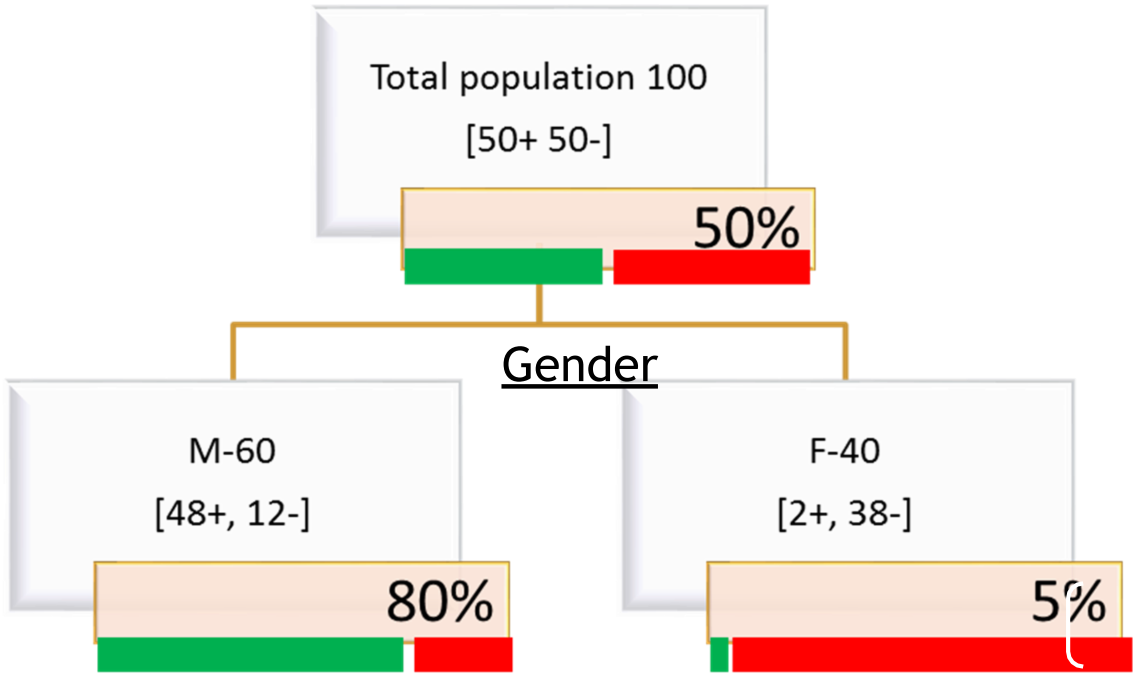
Example Sales Segmentation Based on Gender



Impurity (Diversity) Measures:

- We are looking for a impurity or diversity measure that will give high score for this Age variable(high impurity while segmenting), Low score for Gender variable(Low impurity while segmenting)



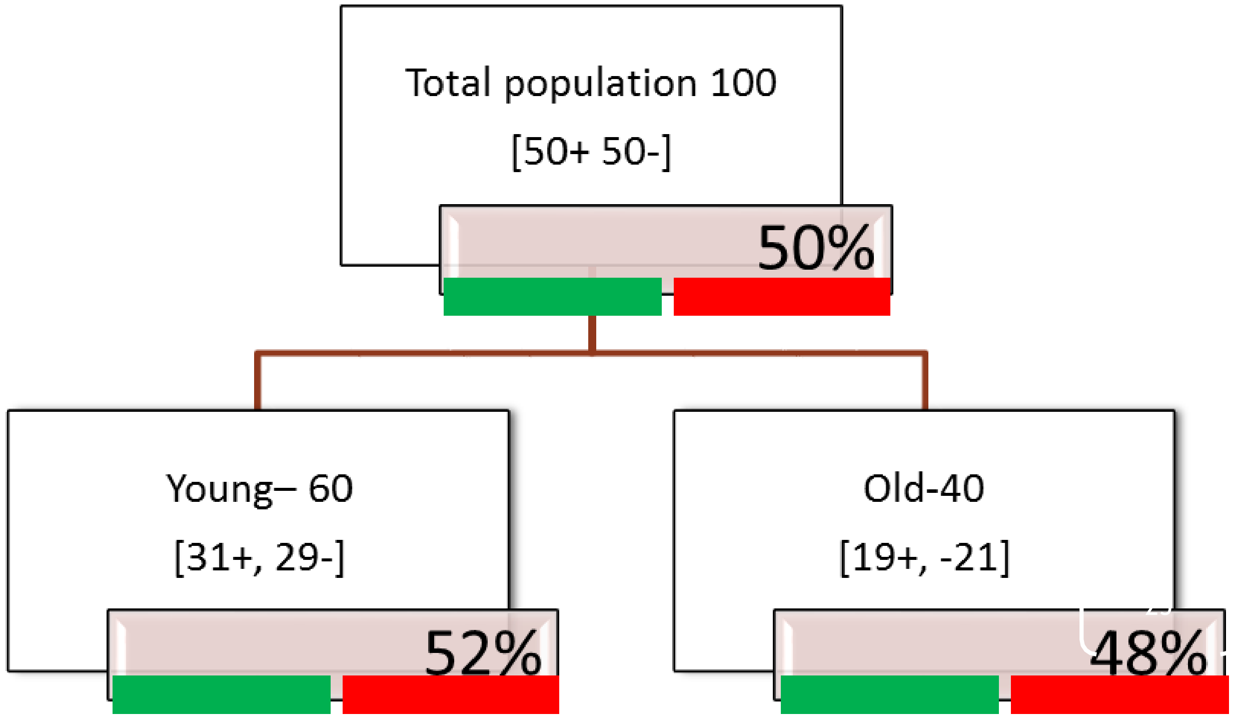


Impurity (Diversity) Measures:

- **Entropy:** Characterizes the impurity/diversity of segment
- Measure of uncertainty/Impurity
- Entropy measures the information amount in a message
- S is a segment of training examples, p_+ is the proportion of positive examples, p_- is the proportion of negative examples
- Entropy(S) = $-p_+ \log_2 p_+ - p_- \log_2 p_-$
  - Where p_+ is the probabailty of positive class and p_- is the probabailty of negative class
- Entropy is highest when the split has p of 0.5.
- Entropy is least when the split is pure .ie p of 1

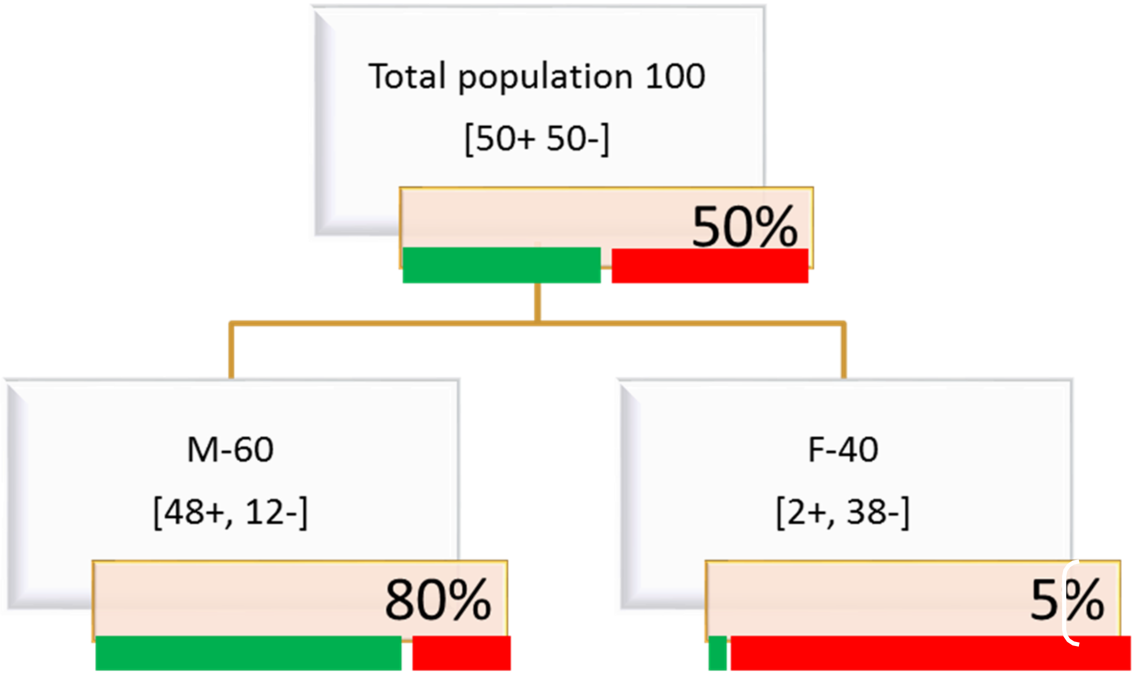
Entropy is highest when the split has p of 0.5

- Entropy(S) = $-p_+ \log_2 p_+ - p_- \log_2 p_-$
- Entropy is highest when the split has p of 0.5
- 50-50 class ratio in a segment is really impure, hence entropy is high
  - Entropy(S) = $-p_+ \log_2 p_+ - p_- \log_2 p_-$
  - Entropy(S) = $-0.5*\log_2(0.5) - 0.5*\log_2(0.5)$
  - Entropy(S) = 1



Entropy is least when the split is pure .ie p of 1

- Entropy(S) = $-p_+ \log_2 p_+ - p_- \log_2 p_-$
  - Entropy is least when the split is pure .ie p of 1
  - 100-0 class ratio in a segment is really pure, hence entropy is low
    - Entropy(S) = $-p_+ \log_2 p_+ - p_- \log_2 p_-$
    - Entropy(S) = $-1*\log_2(1) - 0*\log_2(0)$
    - Entropy(S) = 0



The less the entropy, the better the split

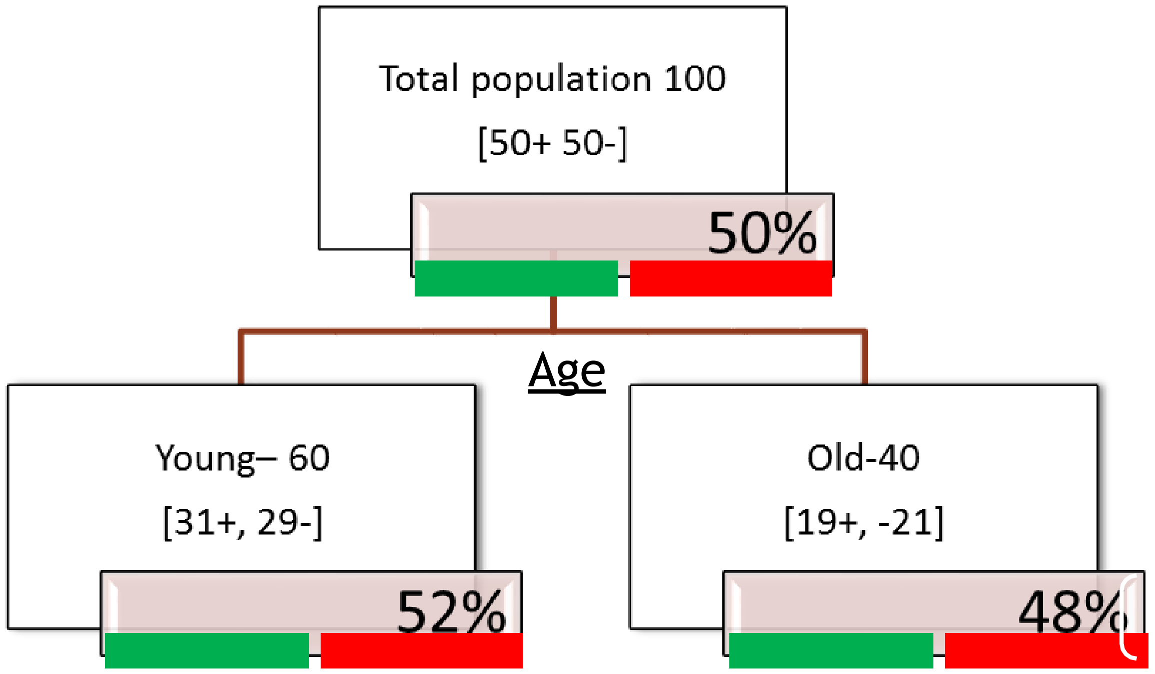
- The less the entropy, the better the split
- Entropy is formulated in such a way that, its value will be high for impure segments

Entropy Calculation - Example

Entropy Calculation

- Entropy at root
- Total population at root 100 [50+,50-]
- Entropy(S) = $-p_+ \log_2 p_+ - p_- \log_2 p_-$
- $-0.5 \log_2 (0.5) - 0.5 \log_2 (0.5)$
- $-(0.5)*(-1) - (0.5)*(-1)$
- 1
- 100% Impurity at root

Entropy(S) = -p+ log2 p+ - p- log2 p-

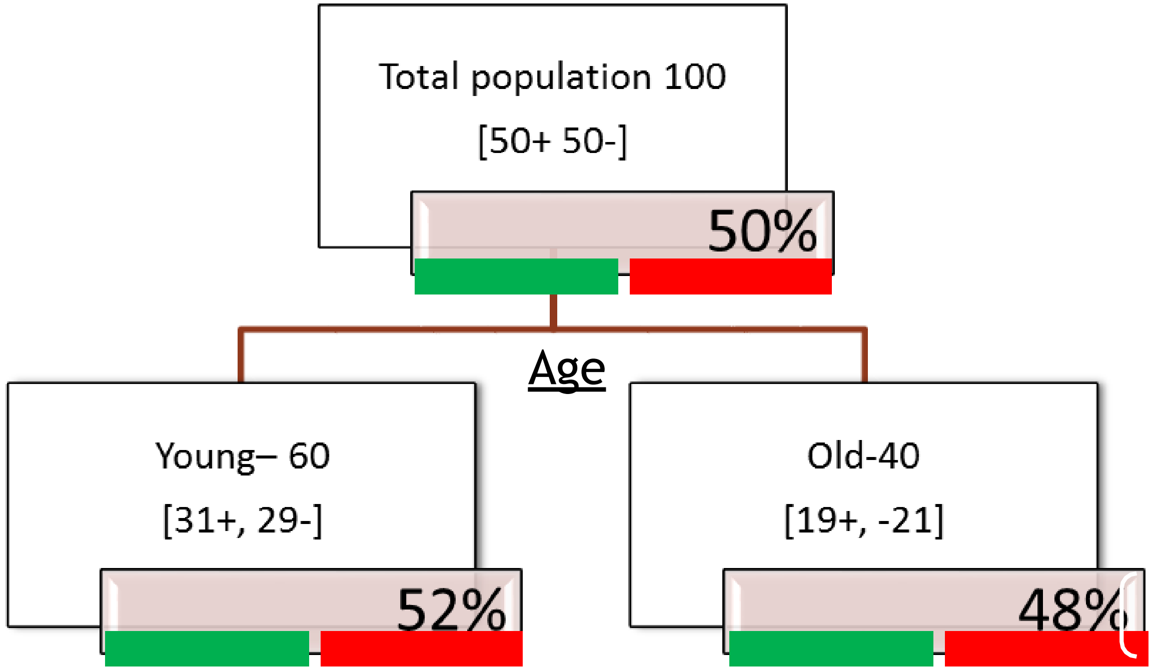


Entropy Calculation

- Gender Splits the population into two segments
- Segment-1 : Age="Young"
- Segment-2: Age="Old"

- Entropy at segment-1
- Age="Young" segment has 60 records [31+,29-]
- Entropy(S) = $-p_+ \log_2 p_+ - p_- \log_2 p_-$
- $-31/60 \log_2 31/60 - 29/60 \log_2 29/60$
- (-31/60)*log(31/60,2)-(29/60)*log(29/60,2)
- 0.9991984 (99% Impurity in this segment)

Entropy(S) = -p+ log2 p+ - p- log2 p-

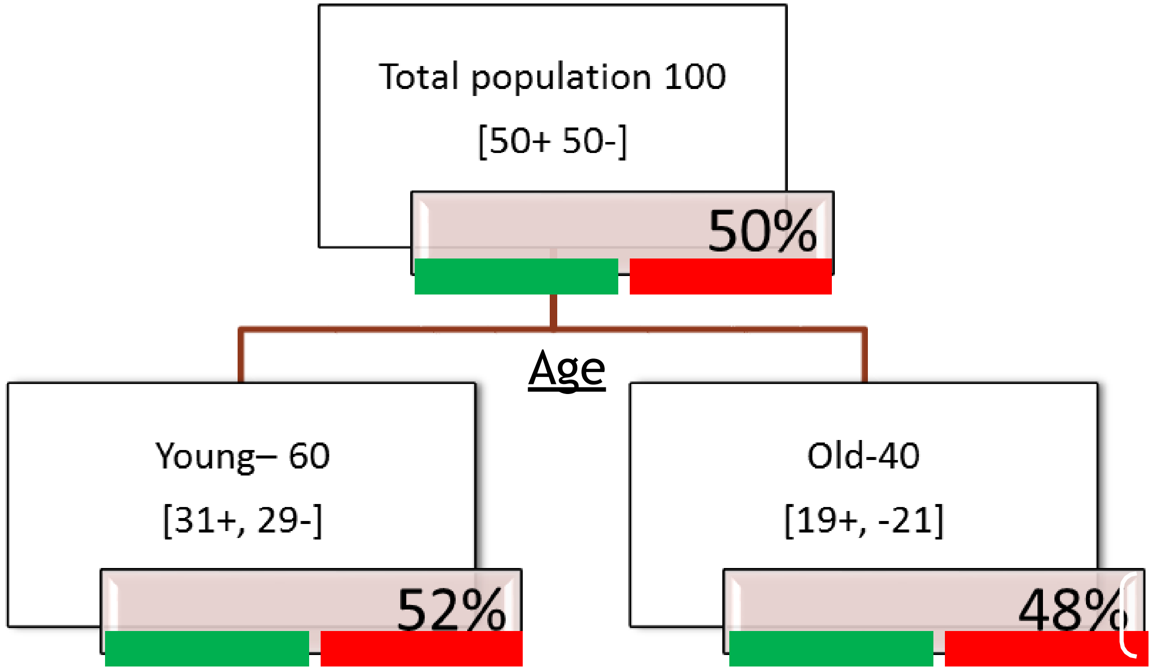


Entropy Calculation

- Gender Splits the population into two segments
- Segment-1 : Age=”Young”
- Segment-2: Age=”Old”

- Entropy at segment-2
- Age=”Old” segment has 40 records [19+,21-]
- Entropy(S) = $-p_+ \log_2 p_+ - p_- \log_2 p_-$
- $-19/40 \log_2 19/40 - 21/40 \log_2 21/40$
- (-19/40)*log(19/40,2)-(21/40)*log(21/40,2)
- 0.9981959(99% Impurity in this segment too)

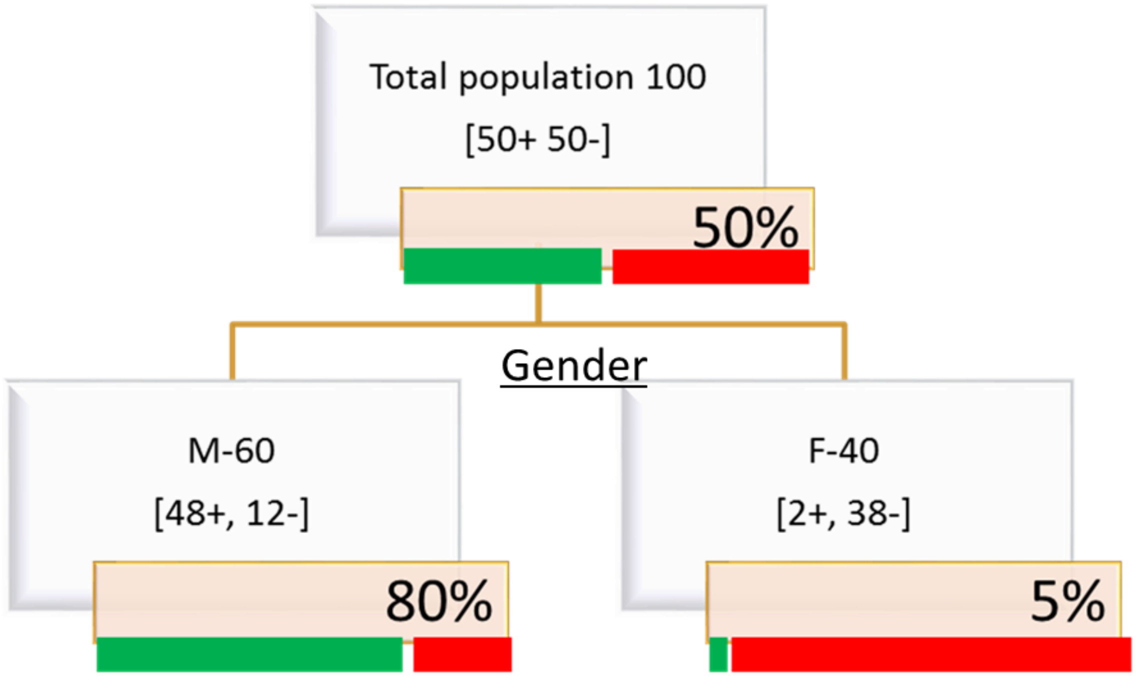
Entropy(S) = -p+ log2 p+ - p- log2



LAB: Entropy Calculation - use calculator or Excel

LAB Entropy Calculation

- Calculate entropy at the root for the given population
- Calculate the entropy for the two distinct gender segments



Code- Entropy Calculation

- Entropy at root 100%
- Male Segment : (-48/60)*log(48/60,2)-(12/60)*log(12/60,2)
  - 0.7219281
- FemaleSegment : (-2/40)*log(2/40,2)-(38/40)*log(38/40,2)
  - 0.286397

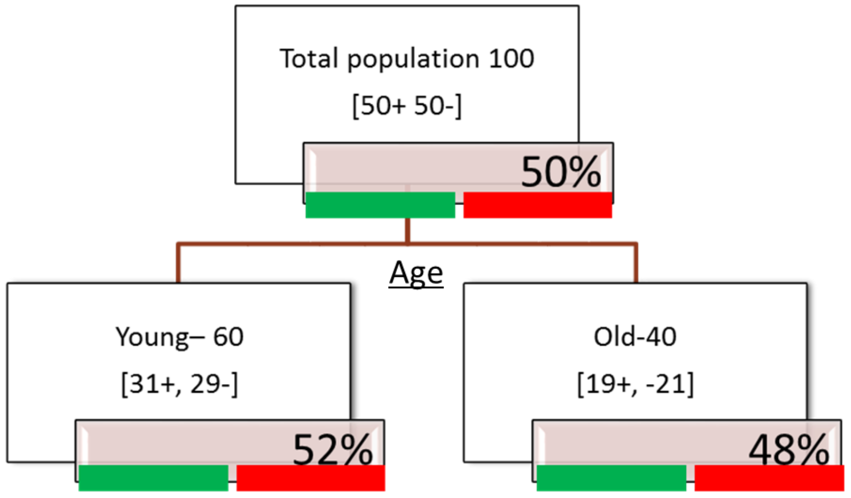
Information Gain

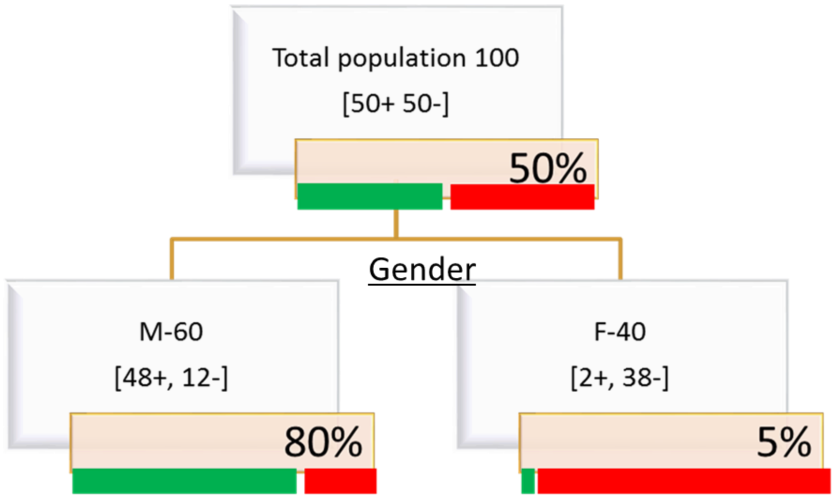
Information Gain

- Information Gain= entropyBeforeSplit – entropyAfterSplit
- Easy way to understand Information gain= (overall entropy at parent node) – (sum of weighted entropy at each child node)
- Attribute with maximum information is best split attribute

Information Gain- Calculation

Information Gain- Calculation





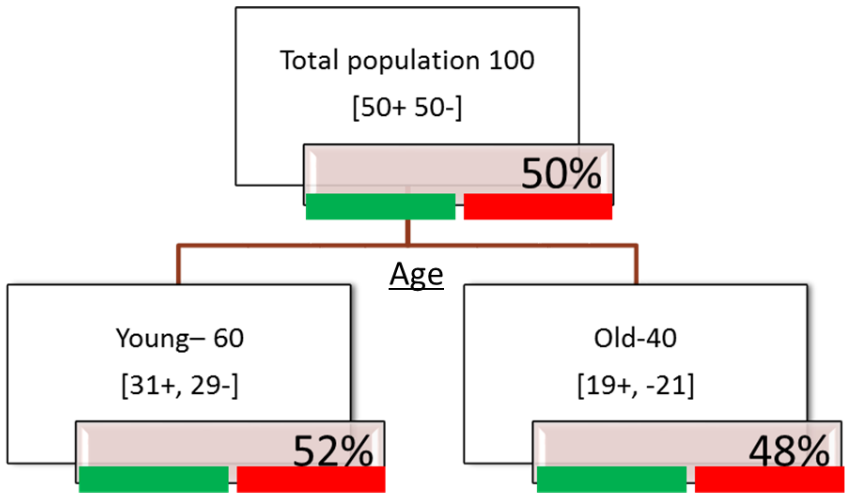
Information Gain= entropy Before Split – entropy After Split

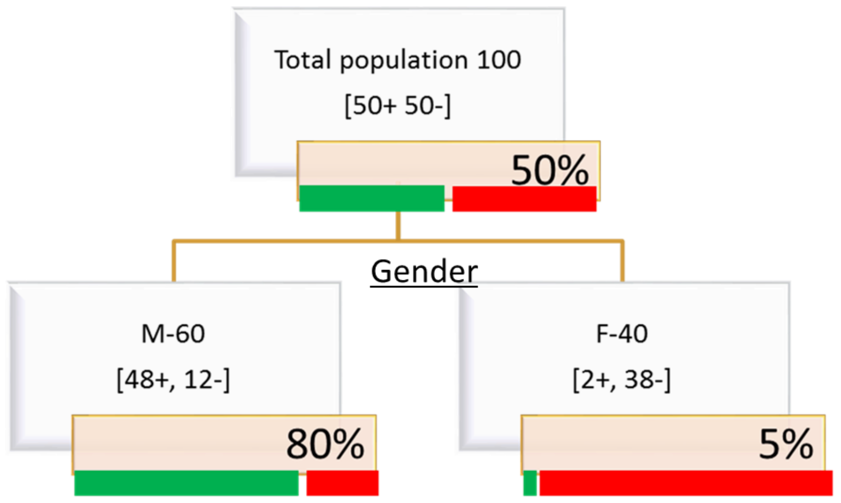
- Entropy Ovearll = 100% (Impurity)
- Entropy Young Segment = 99%
- Entropy Old Sgment = 99%
- Information Gain for Age =100-(0.6*99+0.4*99)=1

- Entropy Ovearll = 100% (Impurity)
- Entropy Male Segment = 72%
- Entropy Female Sgment = 29%
- Information Gain for Age =100-(0.6*72+0.4*29)=**45.2**

LAB: Information Gain- Calculation

- Calculate the information gain values for these two splits





Information Gain= entropy Before Split – entropy After Split

Other Purity (Diversity) Measures

- Chi-square measure of association
- Gini Index : Gini(T) = $1 - \Sigma p_j^2$
- Information Gain Ratio
- Misclassification error

The Decision tree Algorithm

The Decision tree Algorithm

- The major step is to identify the best split variables and best split criteria
- Once we have the split then we have to go to segment level and drill down further

The Decision tree Algorithm

Until stopped:

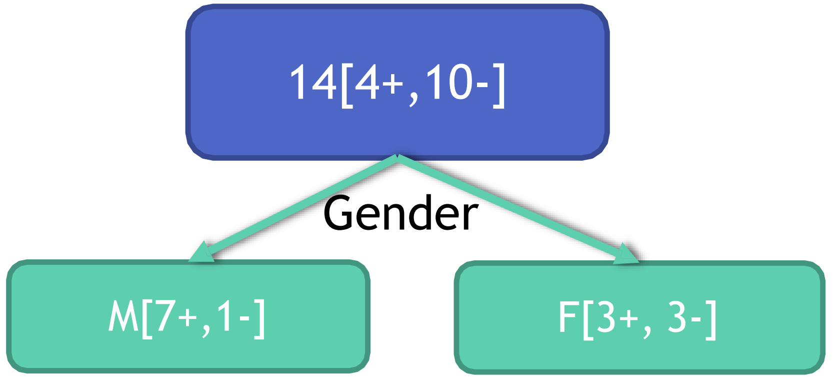
1. Select a leaf node
2. Find the best splitting attribute
3. Spilt the node using the attribute
4. Go to each child node and repeat step 2 & 3

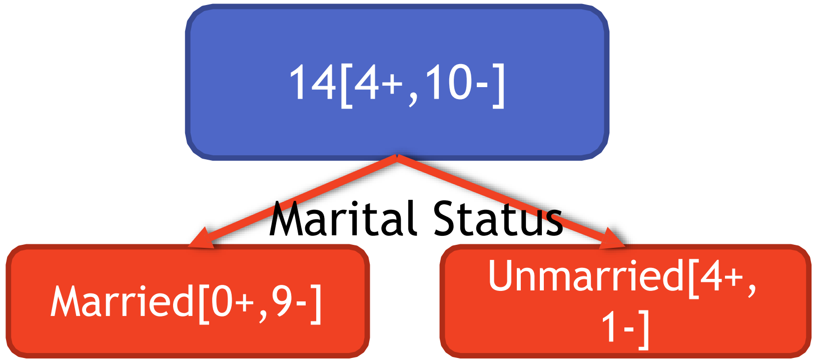
Stopping criteria:

- Each leaf-node contains examples of one type
- Algorithm ran out of attributes
- No further significant information gain

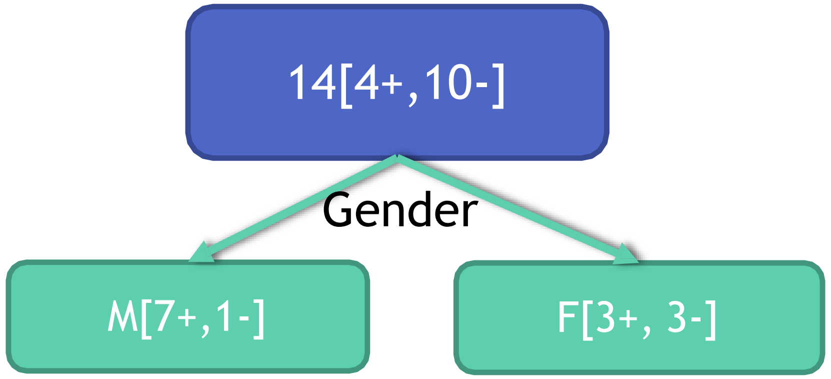
The Decision tree Algorithm- Demo

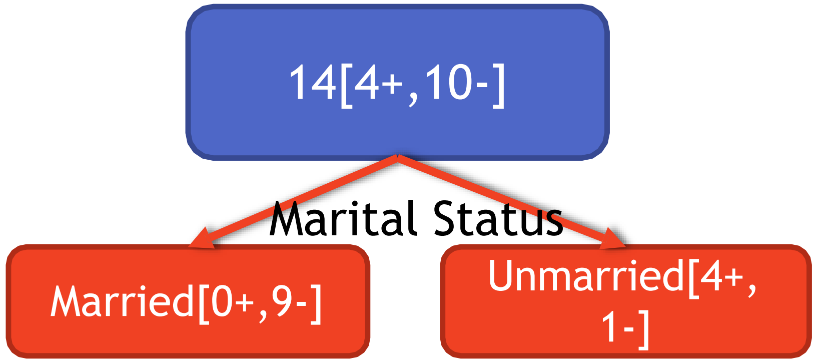
Sr No	Gender	Marital Status	Ordered the product
1	M	Married	No
2	F	Unmarried	Yes
3	M	Married	No
4	M	Married	No
5	M	Married	No
6	M	Married	No
7	F	Unmarried	Yes
8	M	Unmarried	Yes
9	F	Married	No
10	M	Married	No
11	F	Married	No
12	M	Unmarried	No
13	F	Married	No
14	F	Unmarried	Yes





The Decision tree Algorithm- Demo





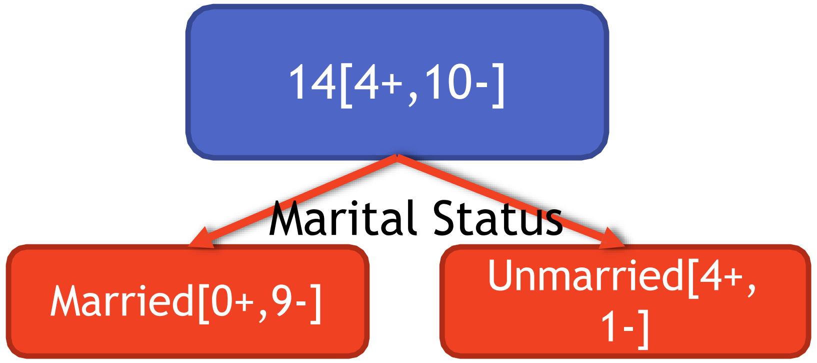
- Entropy([4+,10-]) Ovearll = 86.3% (Impurity)

- Entropy([7+,1-]) Male= 54.3%
- Entropy([3+,3-]) Female = 100%
- Information Gain for Gender=86.3-((8/14)*54.3+(6/14)*100) =12.4

- Entropy([0+,9-]) Married = 0%
- Entropy([4+,1-]) Un Married= 72.1%
- Information Gain for Marital Status=86.3-((9/14)*0+(5/14)*72.1)=**60.5**

The Decision tree Algorithm- Demo

- The information gain for Marital Status is high, so it has to be the first variable for segmentation



- Now we consider the segment “Married” and repeat the same process of looking for the best splitting variable for this sub segment

The Decision tree Algorithm

Until stopped:

1. Select a leaf node
2. Find the best splitting attribute
3. Spilt the node using the attribute
4. Go to each child node and repeat step 2 & 3

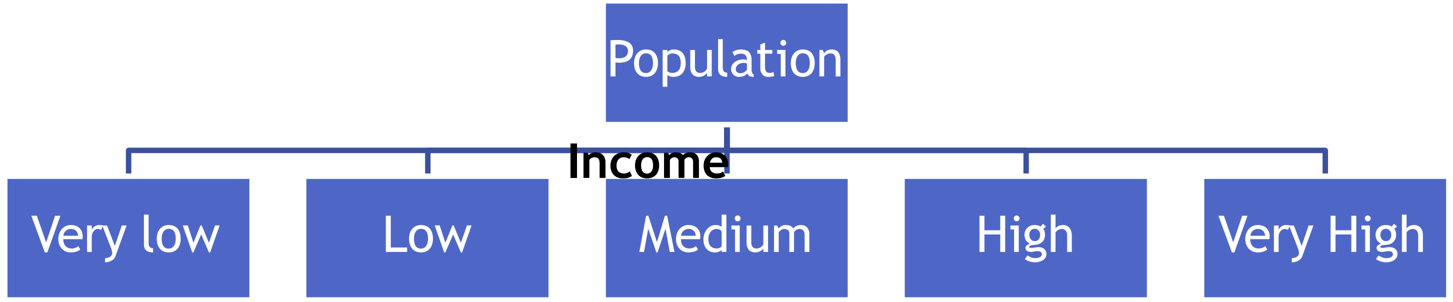
Stopping criteria:

- Each leaf-node contains examples of one type
- Algorithm ran out of attributes
- No further significant information gain

Many Splits for a single variable

Many Splits for a single variable

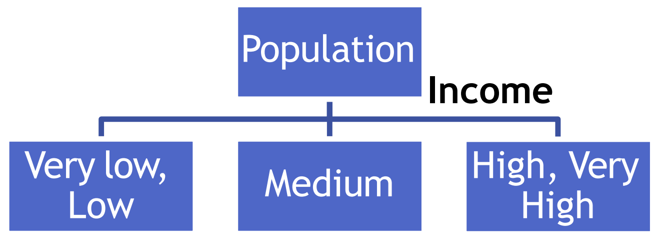
- Sometimes we may find multiple values taken by a variable
  - which will lead to multiple split options for a single variable
    - that will give us multiple information gain values for a single variable

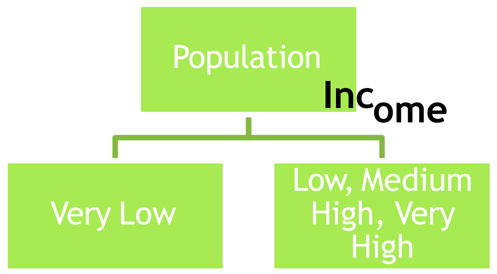


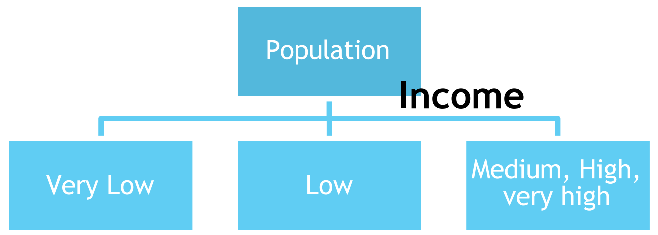
What is the information gain for income?

Many Splits for a single variable

- There are multiple options to calculate Information gain







- What is the information gain for income?
- For income, we will consider all possible scenarios and calculate the information gain for each scenario
- The best split is the one with highest information gain
- Within income, out of all the options, the split with best information gain is considered
- So, node partitioning for multi class attributes need to be included in the decision tree algorithm
- We need find best splitting attribute along with best split rule

The Decision tree Algorithm- Fullversion

Until stopped:

1. Select a leaf node
2. Select an attribute
   - Partition the node population and calculate information gain.
   - Find the split with maximum information gain for a this attribute
3. Repeat this for all attributes
   - Find the best splitting attribute along with best split rule
4. Spilt the node using the attribute

5.Go to each child node and repeat step 2 to 4

Stopping criteria:

- Each leaf-node contains examples of one type
- Algorithm ran out of attributes
- No further significant information gain

LAB: Decision Tree Building

LAB: Decision Tree Building

- Data:Ecom_Cust_Relationship_Management/Ecom_Cust_Survey.csv
- How many customers have participated in the survey?
- Overall most of the customers are satisfied or dis-satisfied?
- Can you segment the data and find the concentrated satisfied and dis-satisfied customer segments ?
- What are the major characteristics of satisfied customers?
- What are the major characteristics of dis-satisfied customers?

Notes: Decision Tree Building

- Decision Tree building in python uses sci-kit learn
- The package expects the data to be in a specific format
- We need to do lot of data preparation before building the actual tree
  - Converting all variables into numerical variables
  - Column names also need to be formatted
  - Create predictor variables matrix and dependent variables

Code: Decision Tree Building

```
df.dropna(inplace='True') # to remove all the missing values rows..
#Q1. How many customers have participated in the survey?
df.info()
#ANS: 11805

#Q.2 Overall most of the customers are satisfied or dis-satisfied?
#total number of customers
df.shape

#number of satisfied customers
freq=df['Overall_Satisfaction'].value_counts(sort=False)
print(freq)
```

Code: Decision Tree Building

```
df['Region'] =df['Region'].map( {'EAST': 1, 'WEST': 2, 'NORTH': 3, 'SOUTH':4} ).astype(int)
df['Customer_Type'] =df['Customer_Type'].map({'Prime': 1, 'Non_Prime': 0}).astype(int)

#We will also need to change the column names, as '.' and spaces are part of many basic functions in python

df.rename(columns={'Order Quantity':'Order_Quantity', 'Improvement Area' :'Improvement_Area'},
inplace=True)
df['Improvement_Area'] =df['Improvement_Area'].map({'Website UI':1, 'Packing &Shipping':2,
'Product Quality':3,}).astype(int)
df['Overall_Satisfaction'] =df['Overall_Satisfaction'].map( {'Dis Satisfied': 0, 'Satisfied': 1}
).astype(int)
```

Code: Decision Tree Building

```
from sklearn import tree

#Defining Features and lables, ignoring cust_num and target variable
features= list(df.columns[2:6])

X= df[features]
y = df['Overall_Satisfaction']

#Building Tree Model
clf =tree.DecisionTreeClassifier(max_depth=2)
clf.fit(X,y)
```

Code: Drawing Decision tree

- Unfortunately drawing a beautiful tree is not easy in python as it is in R, none the less we need a way out
- Have a latest version of jyputer and python installed, anaconda with python 3.5 and jupyter 4.2.3 is being used in this session
- We will need to install graphviz tool in our system and set the path in environment variables.
  - Visit http://www.graphviz.org/Download.php and find the optimal version for the computer.
  - Get the path for gvedit.exe in install directory(for me it was "C:\Program Files (x86)\Graphviz2.38\bin\")
  - goto start->computer->system properties->advanced settings->environment variables and add the path.
- We will need python package pydotplus(for older python versions pydot)
- use this command in your anaconda prompt: conda install -c conda-forge pydotplus
- if an error regarding version occure while installing the package go to https://anaconda.org/search?q=pydotplus
- this link will show the channel name of the suitable version suitable.
- and we can use use : conda install -c <channel name here> pydotplus

Code: Decision Tree Building

```
from IPython.display import Image
from sklearn.externals.six import StringIO

#First run below command on Anaconda Conslole and Install pydotplus manulaly otherwise import
pydotplus throws an error
#pip install pydotplus

#Before drawing the graph below command on anaconda console
#pip install graphviz

import pydotplus

dot_data = StringIO()

tree.export_graphviz(clf,
                     out_file =dot_data,
                     feature_names =features,
                     filled=True, rounded=True,
                     impurity=False)

graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
Image(graph.create_png())
```

Tree Validation

Classification Table & Accuracy

Predicted Classes

Actual Classes

	0(Positive)	1(Negative)
0(Positive)	True positive (TP) Actual condition is Positive, it is truly predicted as positive	False Negatives(FN) Actual condition is Positive, it is falsely predicted as negative
1(Negative)	False Positives(FP) Actual condition is Negative, it is falsely predicted as positive	True Negatives(TN) Actual condition is Negative, it is truly predicted as negative

- Accuracy=(TP+TN)/(TP+FP+FN+TN)
- Misclassification Rate=(FP+FN)/(TP+FP+FN+TN)

LAB: Tree Validation

LAB: Tree Validation

- Create the confusion matrix for the model
- Find the accuracy of the classification for the Ecom_Cust_Survey model

Code: Tree Validation

```
#Tree Validation
predict1 =clf.predict(X)
predict1

from sklearn.metrics import confusion_matrix ###for using confusion matrix###
cm=confusion_matrix(y, predict1)
print (cm)

total = sum(sum(cm))
#####from confusion matrix calculate accuracy
accuracy =(cm[0,0]+cm[1,1])/total
print(accuracy)
```

The Problem of Overfitting

LAB: The Problem of Overfitting

- Dataset: "Buyers Profiles/Train_data.csv"
- Import both test and training data
- Build a decision tree model on training data
- Find the accuracy on training data
- Find the predictions for test data
- What is the model prediction accuracy on test data?

Code: Data Mapping

```
##print train.info()
train.shape
test.shape

# Building the tree  model.
# the data have string  values weneed to  convert them into  numerica values
train['Gender'] =train['Gender'].map( {'Male': 1, 'Female': 0}  ).astype(int)
train['Bought'] =train['Bought'].map({'Yes':1,  'No':0}).astype(int)

test['Gender'] =test['Gender'].map( {'Male': 1, 'Female': 0} ).astype(int)
test['Bought'] =test['Bought'].map({'Yes':1,  'No':0}).astype(int)

##print train.info()
##print test.info()

from sklearn import tree

#Defining Features and lables

features = list(train.columns[:2])
```

Code: Building and Drawing the tree

```
X_train  = train[features]
y_train  = train['Bought']

X_test = test[features]
y_test = test['Bought']

#training Tree Model
clf = tree.DecisionTreeClassifier()
clf.fit(X_train,y_train)

from IPython.display import Image
from sklearn.externals.six import StringIO
import pydotplus
dot_data = StringIO()
tree.export_graphviz(clf,
                     out_file = dot_data,
                     feature_names = features,
                     filled=True, rounded=True,
                     impurity=False)
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
Image(graph.create_png())
```

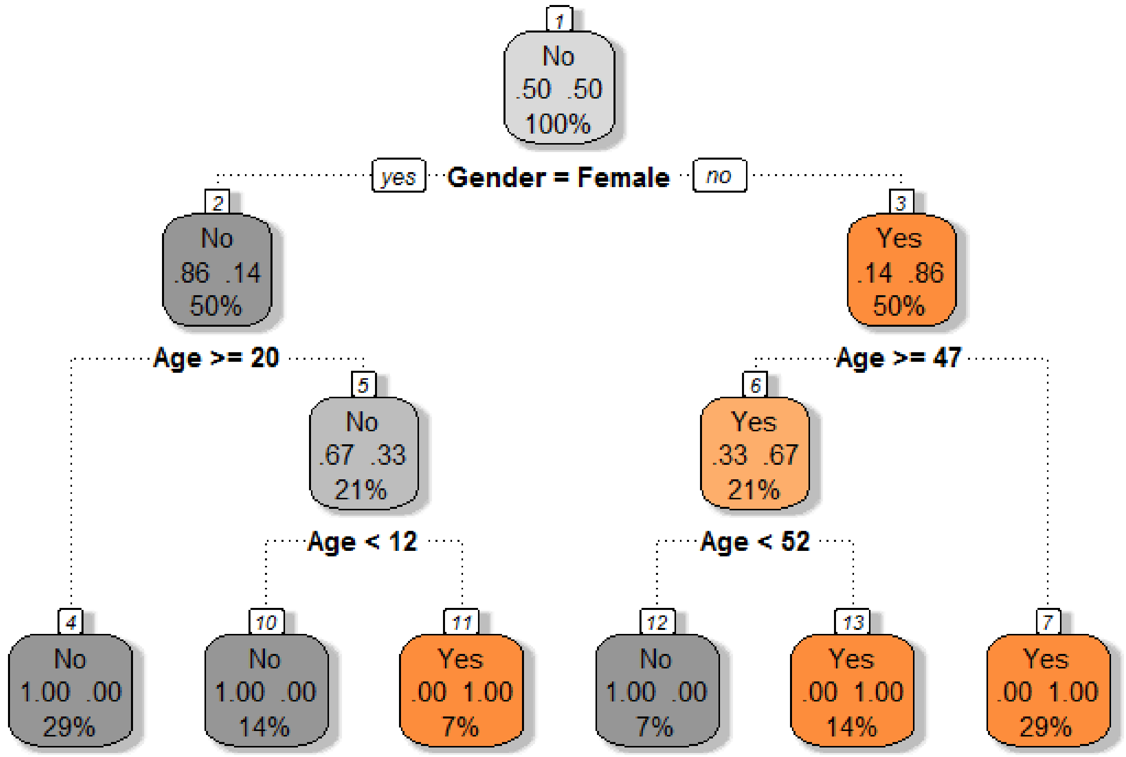
Code: Accuracy on training and test data

```
predict1 = clf.predict(X_train)
print(predict1)

predict2 = clf.predict(X_test)
print(predict2)

####Calculation of Accuracy and Confusion Matrix
#on the train data
from sklearn.metrics import confusion_matrix ###for using confusion matrix###
cm1=confusion_matrix(y_train,predict1)
total1 = sum(sum(cm1))
#####from confusion matrix calculate accuracy
accuracy1 =(cm1[0,0]+cm1[1,1])/total1
accuracy1

#On Test Data
cm2=confusion_matrix(y_test,predict2)
total2 = sum(sum(cm2))
#####from confusion matrix calculate accuracy
accuracy2 =(cm2[0,0]+cm2[1,1])/total2
accuracy2
```

The Problem of Overfitting

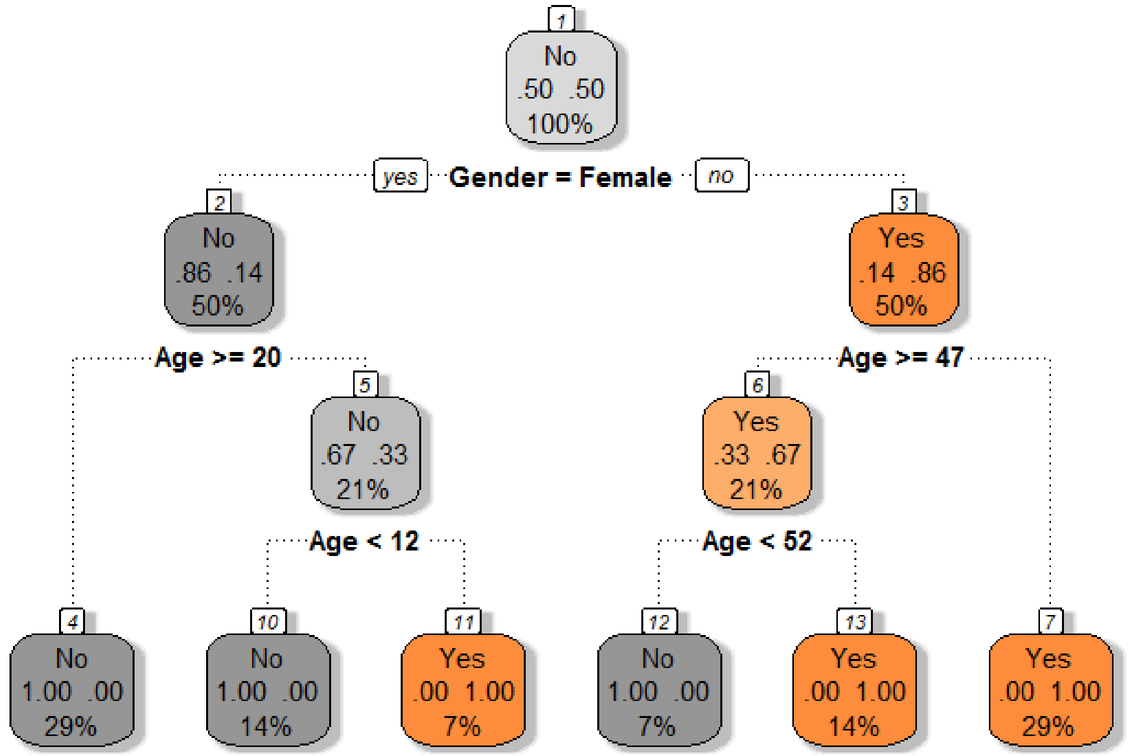
- Build a decision tree on Prune_Sample.csv

Age	Gender	Bought
29	Male	Yes
34	Male	Yes
13	Female	Yes
27	Female	No
10	Female	No
68	Male	Yes
15	Male	Yes
53	Male	Yes
51	Male	No
48	Female	No
63	Female	No
43	Male	Yes
8	Female	No
47	Female	No



The Final Tree with Rules

4) Gender=Female & Age>=20 No *
10) Gender=Female & Age< 20 & Age< 11.5 No *
11) Gender=Female & Age< 20 & Age>=11.5 Yes *
12) Gender=Male & Age>=47 & Age< 52 No *
13) Gender=Male & Age>=47 & Age>=52 Yes *
7) Gender=Male & Age< 47 Yes *



The Problem of Overfitting

Age	Gender	Bought
29	Male	Yes
34	Male	Yes
13	Female	Yes
27	Female	No
10	Female	No
68	Male	Yes
15	Male	Yes
53	Male	Yes
51	Male	No
48	Female	No
63	Female	No
43	Male	Yes
8	Female	No
47	Female	No

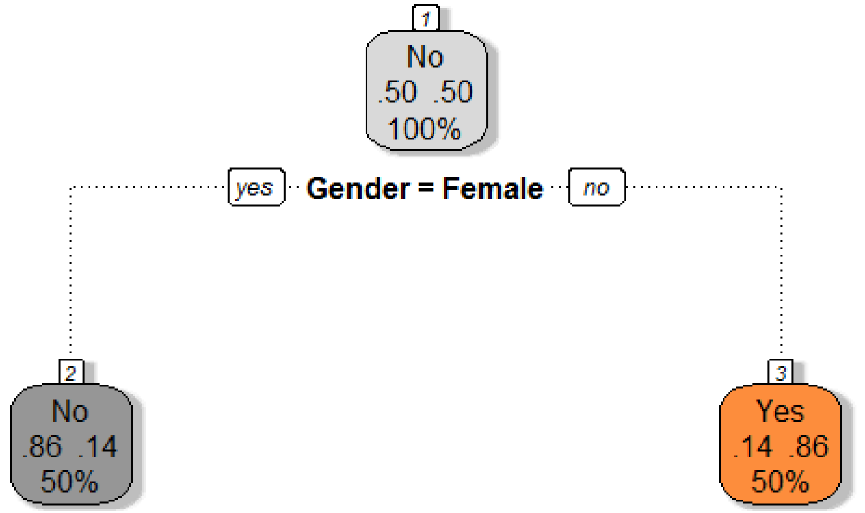
- If we further grow the tree we might even see each row of the input data table as the final rules
- The model will be really good on the training data but it will fail to validate on the test data
- Growing the tree beyond a certain level of complexity leads to overfitting
- A really big tree is very likely to suffer from overfitting.

Pruning

Pruning

Age	Gender	Bought
29	Male	Yes
34	Male	Yes
13	Female	Yes
27	Female	No
10	Female	No
68	Male	Yes
15	Male	Yes
53	Male	Yes
51	Male	No
48	Female	No
63	Female	No
43	Male	Yes
8	Female	No
47	Female	No

- Growing the tree beyond a certain level of complexity leads to overfitting
- In our data, age doesn't have any impact on the target variable.
- Growing the tree beyond Gender is not going to add any value. Need to cut it at Gender
- This process of trimming trees is called Pruning



Pruning to Avoid Overfitting

- Pruning helps us to avoid overfitting
- Generally it is preferred to have a simple model, it avoids overfitting issue
- Any additional split that does not add significant value is not worth while.
- We can avoid overfitting by changing the parameters like
  - max_leaf_nodes
  - min_samples_leaf
  - max_depth

Pruning Parameters

- max_leaf_nodes
  - Reduce the number of leaf nodes
- min_samples_leaf
  - Restrict the size of sample leaf
  - Minimum sample size in terminal nodes can be fixed to 30, 100, 300 or 5% of total
- max_depth
  - Reduce the depth of the tree to build a generalized tree
  - Set the depth of the tree to 3, 5, 10 depending after verification on test data

LAB: Pruning

- Rebuild the model for above data
- Prune the decision tree or rebuild it with optimal parameters
- Calculate the training and test error
- Check whether there is an issue of overfitting in the final model

Code: Pruning

```
dtree = tree.DecisionTreeClassifier(max_leaf_nodes = 10,
                                    min_samples_leaf = 5,
                                    max_depth= 5)
dtree.fit(X_train,y_train)

predict3 = dtree.predict(X_train)
predict4 = dtree.predict(X_test)

cm1=confusion_matrix(y_train,predict3)
cm1
total1 = sum(sum(cm1))
#####from confusion matrix calculate accuracy
accuracy1 =(cm1[0,0]+cm1[1,1])/total1
accuracy1


#On Test Data
cm2=confusion_matrix(y_test,predict4)
cm2
total2 = sum(sum(cm2))
#####from confusion matrix calculate accuracy
accuracy2 =(cm2[0,0]+cm2[1,1])/total2
accuracy2
```

LAB: Tree Building & Model Selection

LAB: Tree Building & Model Selection

- Import fiber bits data. This is internet service provider data. The idea is to predict the customer attrition based on some independent factors
- Build a decision tree model for fiber bits data
- Prune the tree if required
- Find out the final accuracy
- Is there any 100% active/inactive customer segment?

Decision Trees in Python

```
DecisionTreeClassifier(
    class_weight=None,
    criterion='gini’,
    max_depth=None,
    max_features=None,
    max_leaf_nodes=None,
    min_samples_leaf=1,
    min_samples_split=2,
    min_weight_fraction_leaf=0.0,
    presort=False,
    random_state=None,
    splitter='best')
```

Decision Trees parameters in Python

- criterion : default="gini"
  - The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gain.
- splitter : (default="best")
  - The strategy used to choose the split at each node.
  - "best" to choose the best split and "random" to choose the best random split.
- max_features : (default=None)
  - The number of features to consider when looking for the best split:
  - This parameter is to reduce the runtime
  - If "sqrt", then max_features=sqrt(n_features).
  - If "log2", then max_features=log2(n_features).
- max_depth : (default=None)
  - The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.
  - This is to avoid overfitting

Code: Data Importing and train test split

```
Fiber_df = pd.read_csv("D:\\Google Drive\\Training\\Datasets\\Fiberbits\\Fiberbits.csv", header=0)

Fiber_df.info()


#Defining Features and lables
features = list(Fiber_df.drop(['active_cust'],1).columns) #this code gives a list of column names
except 'active_cust'

X = np.array(Fiber_df[features])
y = np.array(Fiber_df['active_cust’])

X_train, X_test, y_train, y_test = cross_validation.train_test_split(X,y, train_size = 0.8)

clf1 = tree.DecisionTreeClassifier()
clf1.fit(X_train,y_train)
```

Code: Tree building and validation

```
clf1 = tree.DecisionTreeClassifier()
clf1.fit(X_train,y_train)


#If we want to see the predictive values we can do this by:
#predict1 = clf1.predict(y_train)
clf1.score(X_train,y_train)

clf1.score(X_test,y_test)
```

Code: Tree Pruning and Validation

```
clf2 = tree.DecisionTreeClassifier(criterion='gini',
                                   splitter='best',
                                   max_depth=20,
                                   min_samples_split=5,
                                   min_samples_leaf=5,
                                   max_leaf_nodes=10)
clf2.fit(X_train,y_train)

clf2.score(X_train,y_train)

clf2.score(X_test,y_test)
```

Conclusion

Conclusion

- Decision trees are powerful and very simple to represent and understand.
- One need to be careful with the size of the tree. Decision trees are more prone to overfitting than other algorithms
- Can be applied to any type of data, especially with categorical predictors
- One can use decision trees to perform a basic customer segmentation and build a different predictive model on the segments
- In python you will get overfitted models with default parameters. You need to adjust the parameters to avoid overfitting

Thank you